THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Mathématiques et Informatique
Unité de recherche : Institut National de Recherche en Informatique et en Automatique

Des algorithmes pour les bandits markoviens : indexabilité et apprentissage

Algorithms for Markovian bandits: Indexability and Learning

Présentée par :

Kimang KHUN

Direction de thèse :

Bruno GAUJAL Directeur de recherche, INRIA Centre Grenoble-Rhône-Alpes	Directeur de thèse
Nicolas GAST Chargé de recherche HDR, INRIA Centre Grenoble-Rhône-Alpes	Co-directeur de thèse

Rapporteurs :

KONSTANTIN AVRACHENKOV
Directeur de recherche, INRIA -SOPHIA ANTIPOLIS-MEDITERRANEE
ADITYA MAHAJAN
Professeur associé, McGill University

Thèse soutenue publiquement le **30 mars 2023**, devant le jury composé de :

BRUNO GAUJAL Directeur de recherche, INRIA CENTRE GRENOBLE-RHONE-ALPES	Directeur de thèse
KONSTANTIN AVRACHENKOV Directeur de recherche, INRIA -SOPHIA ANTIPOLIS-MEDITERRANEE	Rapporteur
ADITYA MAHAJAN Professeur associé, McGill University	Rapporteur
AURELIEN GARIVIER Professeur des Universités, ENS DE LYON	Président
ANA BUSIC Chargé de recherche, INRIA CENTRE DE PARIS	Examinatrice
FRANCK IUTZELER Maître de conférences HDR, UNIVERSITE GRENOBLE ALPES	Examinateur

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisors, Nicolas GAST and Bruno GAUJAL, for their excellent guidance and their attention toward me. I have learnt a lot during my thesis thanks to them. They taught me to be precised in my speech and to think clearly and independently. Without them, I would not have come this far.

Secondly, I wish to thank Annie SIMON and Arnaud LEGRAND for their empathy, their advice on some of my daily life problems in France, and their effort to facilitate the research life in Polaris team. I am grateful to Institut national de recherche en sciences et techonologies du numérique (Inria) for financing my thesis, the graduate program Artificial Intelligence and Advanced Visual Computing of École Polytechnique for granting me the scholarship of Inria for this thesis, and Laboratoire d'Informatique de Grenoble (LIG) for hosting me.

I also want to thank the members of the jury for kindly accepting to be part of this committee. In particular, thanks to the two reviewers, Aditya MAHAJAN and Konstantin AVRACHENKOV, for providing pertinent comments on my manuscript which improved the final version.

My special thanks to my colleagues in Polaris and Datamove teams, to my officemates, Chen YAN and Till KLETTI, for all nice conversations on various topics. My deep thanks to Sotheara LEANG, Rottana LY, Sreynoch SOUNG, Kimhong CHAO, Bonpagna KANN, Pungpanhavoan TEP, Puthineath LAY, Sonita TE, Leangsiv HAN, Sophal THEAR, Mengchhoung ANG, and Nakanyseth VUTH for all fun moments during lunch and during our trips. Thanks to all members of Association des Polytechniciens Khmers (AXK) for their encouragement and friendship. I also want to thank Leangsiv HAN and Mengchhoung ANG for their helps before my departure to Cambodia.

I also enjoyed the friendship of several great people at LIG: Jibril FREJ, Sebastian ALLMEIER, Louis-Sébastien REBUFFI, Victor BOONE, Romain CRAVIC, and Fiorella ALBASINI. I am glad that I have met you.

Finally, I am indebted to my parents, Sothea MONG and Sim VENG, for their belief in me, their endless support, and their unconditional love, to my brothers, Kim Khuy KHUN and Kim Eng KHUN, for their life experiences, their advices, and their helps, to my sisters, Sivlang KHUN and Sivluy KHUN, for their belief in me, their love, and their concerns.

Abstract

A Markovian bandit is a sequential decision problem in which the decision maker has to activate a set of bandit's arms at each time, and the active arms evolve in a Markovian manner. There are two types of Markovian bandits: (i) *rested* bandits where the arms that are not activated (i.e., are passive) remain frozen, and (ii) *restless* bandits where the passive arms evolve in a Markovian manner. In general, Markovian bandits suffer from the curse of dimensionality that often makes the exact solution computationally intractable. So, one has to resort to tractable heuristics such as index policies. Two celebrated indices are the Gittins index for rested bandits and the Whittle index for restless bandits.

This thesis focuses on two questions (1) index computation when all model parameters are known and (2) learning algorithms when the parameters are unknown.

For index computation, we point out the ambiguities in the classical indexability definition and propose a definition that assures the uniqueness of the Whittle index when this latter exists. We then develop an algorithm for testing the indexability and computing the Whittle indices of a restless arm. The theoretical complexity of our algorithm is $\mathcal{O}(S^{2.5286})$, where S is the number of arm's states.

For learning in rested bandits, we propose modifications of PSRL and UCBVI algorithms that we call MB-PSRL and MB-UCBVI. We show that they can leverage Gittins index policy to have a regret guarantee and a runtime scalable in the number of arms. Furthermore, we show that MB-UCRL2, a modification of UCRL2, also has a regret guarantee scalable in the number of arms. However, MB-UCRL2 has a runtime exponential in the number of arms. When learning in restless bandits, the regret guarantee depends heavily on the structure of the bandit. We study how the structure of arms translates into the structure of the bandit. We exhibit a subclass of restless bandits that are not learnable. We also show that it is difficult to construct a subclass of restless bandits with a desirable learning structure by only making assumptions about arms.

Résumé

Un bandit markovien est un problème de décision séquentielle dans lequel un sous-ensemble de bras doivent être activés à chaque instant, et les bras évoluent de manière markovienne. Il y a deux catégories de bandits markoviens. Si les bras qui ne sont pas activés restent figés, on entre alors dans la catégorie des bandits markoviens *avec repos*. S'ils évoluent de manière markovienne, on parle alors de bandit markovien *sans repos*. En général, les bandits markoviens souffrent de la malédiction de la dimension qui rend souvent la solution exacte prohibitive en terme de calculs. Il faut donc recourir à des heuristiques telles que les politiques d'indice. Deux indices célèbres sont l'indice de Gittins pour les bandits avec repos et l'indice de Whittle pour les bandits sans repos.

Cette thèse se concentre sur deux questions : (1) le calcul d'indices lorsque tous les paramètres du modèle sont connus et (2) les algorithmes d'apprentissage lorsque les paramètres sont inconnus.

Pour le calcul de l'indice, nous relevons les ambiguïtés de la définition classique de l'indexabilité et proposons une définition qui assure l'unicité de l'indice de Whittle quand ce dernier existe. Nous développons ensuite un algorithme testant l'indexabilité et calculant les indices de Whittle. La complexité théorique de notre algorithme est $\mathcal{O}(S^{2.5286})$, où S est le nombre d'états du bras.

Pour l'apprentissage dans les bandits avec repos, nous montrons que MB-PSRL et MB-UCBVI, des versions modifiées des algorithmes PSRL et UCBVI, peuvent tirer parti de la politique d'indice de Gittins pour avoir une garantie de regret et un temps d'exécution qui passent à l'échelle avec le nombre de bras. De plus, nous montrons que MB-UCRL2, une version modifiée de UCRL2, possède également une garantie de regret qui passe à l'échelle. Cependant, MB-UCRL2 a un temps d'exécution exponentiel dans le nombre de bras. Lors de l'apprentissage dans les bandits sans repos, la garantie de regret dépend fortement de la structure du bandit. Ainsi, nous étudions comment la structure des bras se traduit dans la structure du bandit. Nous exposons une sous-classe de bandits sans repos qui ne sont pas apprenables. Nous montrons également qu'il est difficile de construire des hypothèses sur les bras qui rendent les bandits sans repos apprenables efficacement.

Contents

Introduction 1

1.1 Topic of the thesis

Markov decision processes (MDPs) are powerful models for solving stochastic optimization problems. They suffer, however, from what is called the *curse of dimensionality*, which basically says that the state size of a Markov process is exponential in the number of system components. This implies that the complexity of computing an optimal policy is generally exponential in the number of system components. The same holds for general-purpose reinforcement learning (RL) algorithms: they all have a regret and a runtime exponential in the number of components, so they also suffer from the same curse.

Very few MDPs are known to escape from this curse of dimensionality. One of the most famous examples is the infinite horizon discounted rested bandit problem which is a special case of Markovian bandit problem. In Markovian bandit problem, a decision maker faces n MDPs (the n components, which we will call the n arms in the rest of the thesis) and chooses $m \leq n$ arms to activate at each decision epoch. Markovian bandits have been applied to many resource allocations and scheduling problems such as wireless communication [Rag+08; LZ10; ALT19], web crawling [Niñ14; AB22], congestion control [Avr+13; APZ18], queueing systems [GKO09; AAR09; ABG09; AAR11; LAV15; BP17; SHS18], and clinical trials [VBW15].

Markovian bandits are well-structured MDPs. They form a subclass of multi-armed bandit problems in which each arm has an internal state that evolves in a Markovian manner as a function of the decision maker's actions. In such a problem, the decision maker observes the state of all arms at each decision time and chooses which to activate. When the state of an arm evolves only when this arm is chosen, one falls into the category of *rested* Markovian bandits. When the state of an arm can also evolve when the arm is not chosen, the problem is called a *restless* bandit problem. It has been shown over the years that index policy, a strategy that requires computation load linearly in number of arms, performs exceptionally well in Markovian bandit problems [GM02; Ans+03; GRK06; Avr+13; AM19b]. Moreover, there is a subclass of Markovian bandits in which index policy is shown to be optimal [Git79].

Two celebrated index definitions are *Gittins index* [Git79] for rested bandits and *Whittle index* [Whi88] for restless bandits. Yet, as mentioned in [Whi96, Chapter 14], the existence of Whittle index is guaranteed only for restless bandits that satisfy a so-called indexability property. To the best of our knowledge, there are very few efficient general-purpose algorithms to test indexability in restless bandits. Meanwhile, few RL algorithms that leverage index policy in learning with Markovian bandits despite its desirable computational complexity compared to dynamic programming solutions. These raise two grand questions in this thesis:

- **How to efficiently test indexability and compute Whittle index?**
- **Can index policy be a pillar for RL algorithms when learning in Markovian bandits?**

1.2 Contributions

Inspired by the challenging questions above, this thesis has several goals that can be regrouped into two main parts: (1) efficiently test indexability and compute the Whittle index for Markovian bandits and (2) learn with efficiency in Markovian bandits.

Firstly, we design an efficient single algorithm that tests indexability and computes Whittle index for both discounted and average reward restless bandits and Gittins index for discounted rested bandits.

Secondly, we show that some RL algorithms can be tailored to discounted rested bandits to have a regret bounded sublinearly in the number of arms and a runtime linear in the number of arms.

Thirdly, we study the implication of the arms' structure in the structure of bandit and regret when learning in a restless bandit with average reward criterion. We show that it is difficult to construct a subclass of restless bandits with desirable learning structure by only making assumptions about arms.

To make these contributions explicit, we divide this thesis into three parts.

1.2.1 Part I: Background

In this part, we recall the existing problem setups and results in the literature. This part serves as the basis for understanding our contributions and positioning them among the vast literature of RL and Markovian bandits.

We present the formalism of Markov decision process (MDP) in Chapter 2: we give the notations, optimization criteria, and the existing theoretical results in MDP. This chapter is the central pillar of all the chapters that follow.

Chapter 3 consists of a summary of existing learning setups in RL, regret definition that is used as a performance metric of learning algorithms, minimax regret lower bound, and several existing results in RL with generic tabular MDPs.

We present the formalism of Markovian bandit in Chapter 4. We give the notations, existing setups and optimization criteria in Markovian bandit literature. We finish this chapter by outlining all the questions discussed in the following chapters.

1.2.2 Part II: Indexability

In this part, we present our contributions to the computational side of Markovian bandit literature.

In Chapter 5, we point out the possible ambiguities in the classical definition of indexability in restless Markovian bandits by providing a few simple counter-examples. This leads us to introduce a new notion of Bellman optimality in the MDP with long-run average reward criterion. This new notion is then used in our definition of indexability, which is thoroughly detailed in the chapter. Then, we give the corresponding Whittle index definition and illustrative examples. Finally, we complete the chapter by studying the properties of Bellman optimality that are useful for indexability tests and index computation.

In Chapter 6, we present a new algorithm to test the indexability presented in the previous chapter and compute Whittle index of any finite state restless arm. It is a single algorithm that tests indexability and computes Whittle index in either discounted or average reward restless bandits, and the Gittins index in discounted rested bandits. Moreover, to the best of our knowledge, this algorithm is the first to achieve subcubic theoretical computational complexity. Indeed, if the considered arm has S states, then the best variant of our algorithm performs $\mathcal{O}(S^{2.5286})$ arithmetic operations[1]. This is made possible by the sporadic use of the fastest matrix multiplication method of [CW87] and the Sherman-Morrison formula. Thanks to the current implementation of matrix multiplication in python, our algorithm is implemented to run in subcubic time in python programming language. We also present a few numerical experiments that witness the subcubic achievement of our algorithm. The code of all experiments is available at `https://gitlab.inria.fr/markovianbandit/efficient-whittle-index-computation`. All variants of our implementation are available in the form of an open-source python package installable by a simple command line: `pip install markovianbandit-pkg`. Finally, we believe that our algorithm has room for improvement, such as exploiting the sparse structure of the arms in Markovian bandits. We leave this question to future work.

1.2.3 Part III: Learning in Markovian bandits

In this part, we present our contributions to the learning side of Markovian bandit literature.

In Chapter 7, we consider an episodic RL problem in which the unknown environment is a rested Markovian bandit having n arms and S states per arm, and the episode length is geometrically distributed. So, the bandit has S^n states in total. Given that Gittins index policy is optimal and computationally efficient when the bandit is known [Git79], we compare the optimism in face of uncertainty (OFU) principle method with posterior sampling in terms of runtime and learning performance encoded by regret. To do so, we adapt UCRL2 [JOA10] and UCBVI [AOM17], two different algorithms from the OFU family, and PSRL [ORV13], an algorithm from the posterior sampling family, to rested bandit with discount. The adapted versions are respectively called MB-UCRL2, MB-UCBVI, and MB-PSRL, where "MB" stands for Markovian bandit. We show that the three MB-* algorithms have a regret bounded by $\tilde{\mathcal{O}}(S\sqrt{nK})$, where K is the number of episodes. This is an exponential improvement in terms of the number of arms n. We also derive a Bayesian minimax regret lower bound for learning algorithms in discounted rested bandit. That is, any algorithm learning in discounted rested bandit suffers a regret that is at least $\Omega(\sqrt{SnK})$. For the computational aspect, we show that UCRL2 and its variants that use extended value iteration [JOA10] cannot leverage Gittins index policy to achieve efficient policy computation. This is because such algorithms

[1]multiplications and additions of real numbers, regardless of their values.

rely on the confidence bonus on arms' transition; which does not allow them to guarantee the OFU principle when working on each arm independently of the other arms. Lastly, we perform several numerical experiments that advocate the good behavior of MB-PSRL. These experiments are reproducible by following this link `https://gitlab.inria.fr/kkhun/learning-in-rested-markovian-bandit`. While acknowledging that posterior sampling has a weaker regret guarantee, we conclude that this approach has the upper hand in problem adaptability when compared to OFU method. This is vital when working with problems having a special structure such as weakly coupled MDPs.

Chapter 8 considers the RL problems in which the unknown environment is a restless Markovian bandit with long-run average reward criterion. We study how the structure of arms translates into the structure of the restless bandit. In particular, we show that no RL algorithms can perform uniformly well in restless bandits whose arms are unichain and have a bounded span of local bias function. This inspires us to study the restless bandit whose arms all have no local transient state. We provide an example in which a restless bandit is multichain even though its arms are ergodic. Also, an ergodic restless bandit can have an arbitrarily large mixing time, although its arms are ergodic and have a bounded mixing time. We also provide a piece of positive result showing that if all arms are ergodic, then the restless bandit is communicating. Moreover, if all arms have an ergodicity coefficient smaller than 1, then the corresponding restless bandit also has an ergodicity coefficient smaller than 1. Finally, we discuss a few issues such as which policy to compare to in the regret definition when learning general restless bandits. All of our arguments imply that defining a subclass of restless bandits with desirable properties for learning is essential but complex.

1.3 Organization of the thesis

We provide the thesis structure in Figure 1.1.

To facilitate the comprehension of this thesis, we suggest the following flow of reading: One would want to start with Chapters 2 and 4, then Part II. Next, one should jump back to Chapter 3 before diving into Part III and finishing the conclusion.

We highlight that Chapters 2 and 4 are required to understand our contributions in Part II, and Part I is required to understand our contributions in Part III.

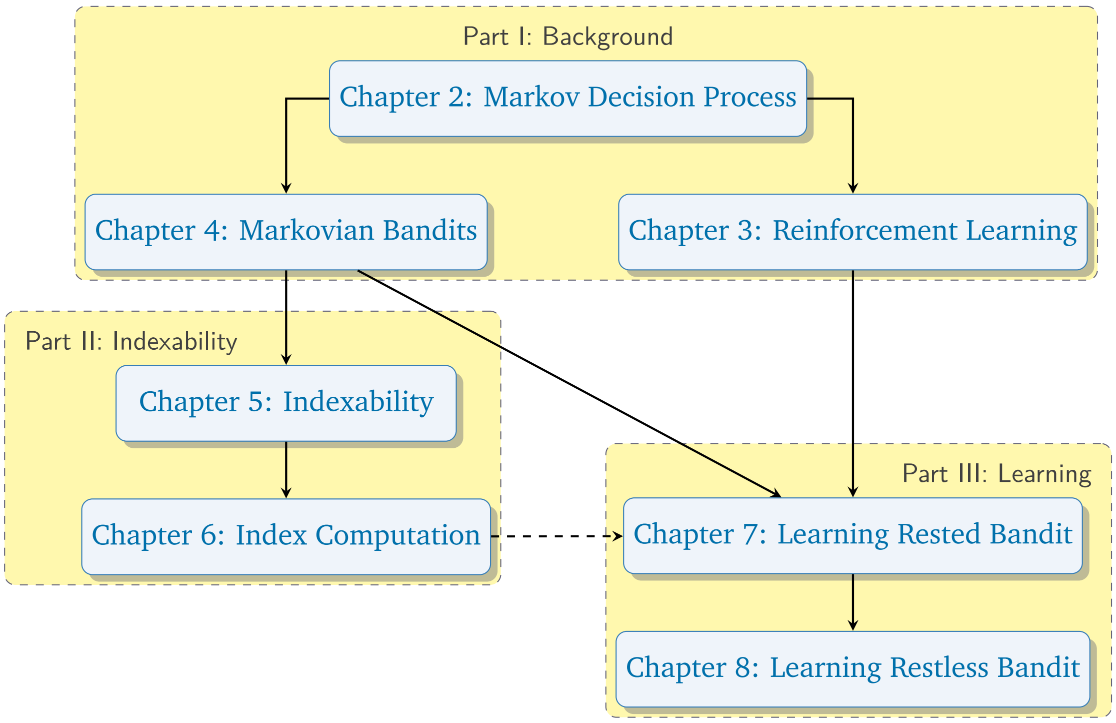

Part I: Background
Chapter 2: Markov Decision Process
Chapter 4: Markovian Bandits
Chapter 3: Reinforcement Learning
Part II: Indexability
Chapter 5: Indexability
Chapter 6: Index Computation
Part III: Learning
Chapter 7: Learning Rested Bandit
Chapter 8: Learning Restless Bandit

Figure 1.1.: Thesis organization

Part I

Background on Markov decision process, reinforcement learning, and Markovian bandit

Markov Decision Process 2

In this chapter, we define the notion of Markov decision process (MDP), a generic model to solve Markovian bandit problem that we will describe in Chapter 4. MDP is also used to mathematically describe the environment in the Reinforcement Learning (RL) framework. An MDP models a discrete-time decision problem where the decision maker executes available "action" over time steps and receives an immediate incentive known as "reward" for each time step. In such a problem, the decision maker seeks to maximize the expected cumulative rewards by identifying a sequence of actions that produce such an effect.

In Section 2.1, we lay out the notation of MDP's parameters and the dynamic of the decision process. Then, in Section 2.2, we briefly present the finite-horizon setting. Similarly, we present the classical settings for the infinite horizon in Sections 2.3 and 2.4.

2.1 Definitions and notations

In this section, we give the *formalism* of Markov decision process. We essentially follow the notations of [Put14].

2.1.1 State, action, reward, and state transition

A Markov decision process M is defined as a 4-tuple $M := \langle \mathcal{S}, \mathcal{A}, r, p \rangle$. $\mathcal{S}$ and $\mathcal{A} := \cup_{s\in\mathcal{S}}\mathcal{A}_s$ denote the *state* and *action* space of the MDP. When the MDP is in state $s \in \mathcal{S}$, the decision maker can execute one of the available actions in $\mathcal{A}_s$. As a result of executing action $a \in \mathcal{A}_s$ in state s, the MDP incurs a random *reward* with the *expected value* $r(s, a)$ and then transitions to a new state $s' \in \mathcal{S}$ with probability $p(s' \mid s, a) \in [0, 1]$, where $\sum_{s'\in\mathcal{S}} p(s' \mid s, a) = 1$. The name *Markov* comes from the fact that the random reward and the next state depend only on state s and action a and are independent of anything else. This thesis considers the MDPs with *finite* state and action spaces, $|\mathcal{S}| =: S \in \mathbb{N}$, and $|\mathcal{A}| =: A \in \mathbb{N}$.

2.1.2 Sequential decision problem and policy

In this thesis, the decision maker executes actions at *discrete* time steps. We denote a decision time by $t \in \mathbb{N}^+$ where $\mathbb{N}^+ := \mathbb{N} \setminus \{0\}$ is the set of positive natural numbers. At time step $t \geq 1$, the MDP is in state s_t, and the decision maker executes an action a_t. The MDP incurs a (random) reward denoted by r_t and transitions to the next state denoted by s_{t+1}. This mechanism is repeated, and one obtains a sequence of the form $\{s_1, a_1, r_1, s_2, \dots, s_t, a_t, r_t, s_{t+1}, \dots\}$ that is called *history* (also known as a "trajectory").

A *deterministic decision rule* π is a mapping function from state space $\mathcal{S}$ to action space $\mathcal{A}$ where for each $s \in \mathcal{S}$, $\pi(s) \in \mathcal{A}_s$. A *policy* is a sequence of deterministic decision rules $\{\pi_1, \pi_2, \dots\}$ such that at time step $t \geq 1$, the decision maker follows the policy by executing the action $\pi_t(s_t) \in \mathcal{A}$. With a slight abuse of notation, we also denote a policy by $\pi = \{\pi_1, \pi_2, \dots\}$. We denote the set of all policies by Π. We say that a policy is *stationary* when the decision rules are invariant over time step, $\pi_t = \pi$ for all $t \geq 1$ and $\pi : \mathcal{S} \mapsto \mathcal{A}$.

The state of the MDP at time step 1 is called the *initial state* and is generally drawn from a probability distribution ρ such that $\sum_{s \in \mathcal{S}} \rho(s) = 1$. Given the initial state s_1, a policy $\pi \in \Pi$ incurs a sequence $\{s_1, a_1, r_1, \dots, s_t, a_t, r_t, \dots\}$ which is a *stochastic process* with a well-defined probability distribution [Put14, Section 2.1.6]. We will denote by $\mathbb{P}^\pi(\cdot \mid s_1)$ the *probability measure* associated with this stochastic process and denote by $\mathbb{E}^\pi[\cdot \mid s_1]$ the corresponding *expectation*.

In the following sections, we specify the objective function of the decision problem.

2.2 Finite horizon problem

In the finite-horizon setting, the decision maker can collect rewards from an MDP M over a *fixed* number of time steps $H \in \mathbb{N}^+$ called the *horizon*. We call the expected cumulative reward when following a policy π as the *value function*. That is, the value of state s when starting at time step $1 \leq h \leq H$ and following policy π is given by $w^\pi_{h:H}(s) := \mathbb{E}^\pi\left[\sum_{t=h}^H r_t \mid s_h = s\right]$. Formally, the decision maker wants to find a policy that satisfies

$$\sup_{\pi \in \Pi} \sum_{s \in \mathcal{S}} \rho(s) w^\pi_{1:H}(s). \tag{2.1}$$

From [Put14, Chapter 4], there always exists an *optimal* policy $\pi^* = \{\pi^*_1, \pi^*_2, \dots, \pi^*_H\}$ that maximizes Equation (2.1) for any ρ and such that for all $1 \leq h \leq H$, $\pi^*_h : \mathcal{S} \mapsto \mathcal{A}$

is a deterministic decision rule and independent of the initial state distribution. For each state s, we denote the *optimal* expected cumulative reward from s over time steps h to H by $w^*_{h:H}(s) := \max_{\pi\in\Pi} w^\pi_{h:H}(s)$. The maximum value of Equation (2.1) is given by $\sum_{s\in\mathcal{S}} \rho(s) w^*_{1:H}(s)$.

From [Put14, Chapter 4], the Bellman *optimality* equations in this setting are written: for $1 \leq h \leq H-1$ and $s \in \mathcal{S}$,

$$w^*_{h:H}(s) = \max_{a\in\mathcal{A}_s} \Big(r(s,a) + \sum_{s'\in\mathcal{S}} p(s' \mid s,a) w^*_{h+1:H}(s') \Big) \tag{2.2}$$

and $w^*_{H:H}(s) = \max_{a\in\mathcal{A}_s} r(s,a)$. So an optimal policy π^* can be constructed using backward induction on Equation (2.2): for each state s

- $\pi^*_H(s) = \arg\max_{a\in\mathcal{A}_s} r(s,a)$;
- for h from $H-1$ to 1, $\pi^*_h(s) = \arg\max_{a\in\mathcal{A}_s} \Big(r(s,a) + \sum_{s'\in\mathcal{S}} p(s' \mid s,a) w^*_{h+1:H}(s') \Big)$

where the ties are broken arbitrarily.

Finally, from [Put14, Chapter 4] the value function also satisfies the Bellman *evaluation* equations: given a policy π, for $1 \leq h \leq H$ and $s \in \mathcal{S}$,

$$w^\pi_{h:H}(s) = r\Big(s, \pi_h(s)\Big) + \sum_{s'\in\mathcal{S}} p\Big(s' \mid s, \pi_h(s)\Big) w^\pi_{h+1:H}(s') \tag{2.3}$$

with $w^\pi_{H+1:H}(s) = 0$.

2.3 Infinite horizon discounted problem

In some problems, the decision maker can collect rewards over an infinite number of time steps, and the immediate reward incurred by the MDP is more critical than those rewards in the future time steps. To capture this aspect, one can introduce a discount factor denoted by γ where $\gamma \in [0,1)$. If $\gamma = 0$, the decision maker is solely interested in the immediate reward from the current state of the MDP. The value of state s when following a policy π is defined by $v^\pi_\gamma(s) := \mathbb{E}^\pi \Big[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t \mid s_1 = s \Big]$. It captures the cumulative discounted reward one would expect when starting from state s and following policy π. The decision maker wants to find a policy that satisfies

$$\sup_{\pi\in\Pi} \sum_{s\in\mathcal{S}} \rho(s) v^\pi_\gamma(s). \tag{2.4}$$

From [Put14, Chapter 6], there always exists a stationary optimal policy $\pi^* : \mathcal{S} \mapsto \mathcal{A}$ that maximizes Equation (2.4) for any initial distribution ρ. For any s, since $0 \leq \gamma < 1$, $v_\gamma^\pi(s)$ is a geometric series that converges. So, $v_\gamma^\pi(s)$ exists and is well-defined for any state s and any policy $\pi \in \Pi$. The value function of policy π satisfies the following Bellman evaluation equation: for each $s \in \mathcal{S}$

$$v_\gamma^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, \pi(s)) v_\gamma^\pi(s'). \tag{2.5}$$

The *optimal* value function is defined by $v_\gamma^*(s) := \max_{\pi \in \Pi} v_\gamma^\pi(s)$ for all $s \in \mathcal{S}$. By [Put14, Theorem 6.2.5], the optimal value function $\boldsymbol{v}_\gamma^* \in \mathbb{R}^S$ is unique: any optimal policies induce the same value function. Moreover, $\boldsymbol{v}_\gamma^*$ satisfies the Bellman optimality equation: for each $s \in \mathcal{S}$

$$v_\gamma^*(s) = \max_{a \in \mathcal{A}_s} \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v_\gamma^*(s') \Big). \tag{2.6}$$

So, the maximum value of Equation (2.4) is given by $\sum_{s \in \mathcal{S}} \rho(s) v_\gamma^*(s)$.

It is well-known that an optimal policy can be constructed as the following: for all state s, $\pi^*(s) = \arg\max_{a \in \mathcal{A}_s} \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v_\gamma^*(s') \Big)$, where the ties are broken arbitrarily. This requires the optimal value function $\boldsymbol{v}_\gamma^*$, which is approximated by iterative algorithms such as value iteration:

- initialize $v_\gamma^0(s) = 0$ for all $s \in \mathcal{S}$
- iterate over k: $v_\gamma^{k+1}(s) = \max_{a \in \mathcal{A}_s} \Big(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) v_\gamma^k(s') \Big)$ for all $s \in \mathcal{S}$, then update $k = k+1$ and repeat until $\left\| \boldsymbol{v}_\gamma^{k+1} - \boldsymbol{v}_\gamma^k \right\| \leq (1-\gamma)\varepsilon/\gamma$

where $\|\cdot\|$ can be the maximum norm or span semi-norm on $\mathbb{R}^S$, and ε is the accuracy of the approximation. From [Put14, Chapter 6], this iterative schema always converges: $\lim_{k \to \infty} \boldsymbol{v}_\gamma^k = \boldsymbol{v}_\gamma^*$. Precisely, this schema stops after a finite number of iterations K, where $\left\| \boldsymbol{v}_\gamma^K - \boldsymbol{v}_\gamma^* \right\| \leq \varepsilon$ (see [Put14, Theorem 6.3.1]).

This setting with a discount factor γ can be seen as a finite-horizon setting whose horizon is randomly sampled from a *geometric distribution* with parameter $(1-\gamma)$. Precisely, the decision maker can stop interacting with the MDP with probability $(1-\gamma)$ at each time step. So, the value of state s under stationary policy π is the expected sum of rewards over the random horizon:

$$v_\gamma^\pi(s) := \mathbb{E}^\pi \left[\sum_{t=1}^{+\infty} \gamma^{t-1} r_t \mid s_1{=}s \right] = \mathbb{E} \left[\mathbb{E}^\pi \left[\sum_{t=1}^{H} r_t \mid H, s_1{=}s \right] \right] = \mathbb{E}\left[W_{1:H}^\pi(s) \right], \tag{2.7}$$

where the expectation of the fourth term integrates over the randomness of the horizon H whose expected value is $1/(1-\gamma)$.

2.4 Infinite horizon average reward criterion

In some problems, the decision maker is more interested in the long-term average reward per decision time than the expected cumulative discounted reward. Therefore, we focus on this criterion in this section.

2.4.1 Gain and bias

Formally, the decision maker wants to identify a policy that satisfies

$$\sup_{\pi\in\Pi}\sum_{s\in\mathcal{S}}\rho(s)\liminf_{T\to+\infty}\frac{1}{T}\mathbb{E}^{\pi}\left[\sum_{t=1}^{T}r_t \mid s_1=s\right]. \tag{2.8}$$

Since the state space is finite, the infimum limit of Equation (2.8) equals the supremum limit for any stationary policy π (see [Put14, Chapter 8]). Hence, the limit of Equation (2.8) exists and is called the *long-run* average reward or *gain* of state s under policy π. Concretely, it is defined by

$$g^{\pi}(s) := \lim_{T\to+\infty}\frac{1}{T}\mathbb{E}^{\pi}\left[\sum_{t=1}^{T}r_t \mid s_1=s\right]. \tag{2.9}$$

This notion generalizes both the finite horizon and the discounted settings when $H\to+\infty$ or $\gamma\to 1$ because it is shown (see [Put14, Sections 8.2.1 and 8.2.2]) that

$$\lim_{H\to+\infty}\frac{1}{H}w^{\pi}_{1:H}(s)=g^{\pi}(s) \text{ and } \lim_{\gamma\to1}(1-\gamma)v^{\pi}_{\gamma}(s)=g^{\pi}(s). \tag{2.10}$$

In consequence, if π and π' are two stationary policies such that $g^{\pi}(s)<g^{\pi'}(s)$, then for H big enough and γ close enough to 1, $w^{\pi}_{1:H}(s)<w^{\pi'}_{1:H}(s)$ and $v^{\pi}_{\gamma}(s)<v^{\pi'}_{\gamma}(s)$.

The gain of a stationary policy captures the average reward obtained in the *steady regime* (or *asymptotic regime*). Another quantity associated with a stationary policy π is its *bias function*, defined by: for each $s\in\mathcal{S}$

$$h^{\pi}(s) := \underset{T\to+\infty}{\text{C-}\lim}\,\mathbb{E}^{\pi}\left[\sum_{t=1}^{T}r_t-g^{\pi}(s_t) \mid s_1=s\right]. \tag{2.11}$$

The *Cesaro-limit* denoted by C-lim is used because it is well-defined for any stationary policies, albeit the “classical” limit may not exist for a stationary policy that induces a stochastic process governed by a *periodic* Markov chain. The bias function captures the expected total difference between the reward and the average reward in the steady regime. The difference of bias values $h^\pi(s) - h^\pi(s')$ captures the (dis-)advantage of starting at state s rather than s' when following policy π. We denote by $sp(\boldsymbol{h}^\pi) := \max_{s\in\mathcal{S}} h^\pi(s) - \min_{s\in\mathcal{S}} h^\pi(s)$ the range or *span* of the bias function of policy π.

Any stationary policy π induces a Markov reward process (MRP), where the reward function and state transition are encoded by vector $\boldsymbol{r}^\pi$ and stochastic matrix $\boldsymbol{P}^\pi$ such that for any $s, s' \in \mathcal{S}$, $r^\pi(s) := r(s, \pi(s))$ and $P^\pi(s,s') := p(s' \mid s, \pi(s))$. The gain and bias can be computed using Bellman evaluation equations given in Proposition 2.1.

Proposition 2.1 ([Put14, Theorem 8.2.6])
For any stationary policy π, the gain $\boldsymbol{g}^\pi$ and bias $\boldsymbol{h}^\pi$ satisfy the following system of Bellman evaluation equations: for any $s \in \mathcal{S}$

$$g(s) - \sum_{s'\in\mathcal{S}} P^\pi(s,s')g(s') = 0 \tag{2.12}$$

$$g(s) - r^\pi(s) + h(s) - \sum_{s'\in\mathcal{S}} P^\pi(s,s')h(s') = 0. \tag{2.13}$$

Moreover, suppose that $\boldsymbol{g}$ and $\boldsymbol{h}$ satisfy (2.12) *and* (2.13)*. Then, $\boldsymbol{g}^\pi = \boldsymbol{g}$ and $\boldsymbol{h}^\pi = \boldsymbol{h} + \boldsymbol{u}$ where $u(s) = \sum_{s'\in\mathcal{S}} P^\pi(s,s')u(s')$ for all s.*

Equations (2.12) and (2.13) uniquely define $\boldsymbol{g}$ and determine $\boldsymbol{h}$ up to an element of the null space of $(\boldsymbol{I} - \boldsymbol{P}^\pi)$ where $\boldsymbol{I}$ is the identity matrix of size $S \times S$. If $\boldsymbol{P}^\pi$ is unichain[1], then we say that policy π is unichain, and its gain is state independent: $g^\pi(s) = g^\pi(s')$ for any $s, s' \in \mathcal{S}$. So, for any policy π that is unichain, we just denote its gain by g^π. Moreover, if π is unichain, its bias vector $\boldsymbol{h}^\pi$ is defined up to a constant vector (see [Put14, Chapter 8]).

2.4.2 Classification of Markov decision processes

Since Equation (2.8) compares policies based on their average reward in the steady regime, we need to take into account the chain structure induced by the policies. In an MDP with a finite state space, the states are either *transient* or *recurrent*. A state

[1]we provide more explanation about unichain in (ii) of Definition 2.2.

is *transient* if it is never visited in the steady regime. For more formal definition, the reader may refer to [Put14, Appendix A] or [LPW17]. We classify the MDPs according to the following definition.

Definition 2.2 (Classification of MDPs)
We say that an MDP is

(i) ***ergodic*** *if the Markov chain induced by any stationary policy has a single recurrent class that coincides the state space (i.e., all states are visited infinitely often with probability* 1 *independently of initial state);*

(ii) ***unichain*** *if the Markov chain induced by any stationary policy is* unichain, *i.e., it has a single recurrent class plus a –possibly empty– set of transient states;*

(iii) ***multichain*** *if it is not unichain;*

(iv) ***communicating*** *if for every pair of states* $(s, s') \in \mathcal{S}$*, there exists a stationary policy under which* s' *is accessible from* s *in a finite number of time steps with non-zero probability*

(v) ***weakly communicating*** *if the state space can be partitioned into two subsets* $\mathcal{S}^{\mathcal{C}}$ *and* $\mathcal{S}^{\mathcal{T}}$ *(with* $\mathcal{S}^{\mathcal{T}}$ *possibly empty), such that for every pair of states* $(s, s') \in \mathcal{S}^{\mathcal{C}}$*, there exists a stationary policy under which* s' *is accessible from* s *in finite time with non-zero probability, and all states in* $\mathcal{S}^{\mathcal{T}}$ *are transient under all stationary policies.*

With this definition, ergodic MDPs are special cases of unichain MDPs, and weakly communicating MDPs generalize communicating MDPs. Moreover, ergodic MDPs are communicating, and unichain MDPs are weakly communicating. Figure 2.1 summarizes this relation.

Testing if a Markov chain is unichain can be done in $\mathcal{O}(S^2)$ using Tarjan's strongly connected component algorithm. Testing if an MDP is unichain is, however, NP-hard [Tsi07].

2.4.3 Gain optimality

Similarly to the discounted setting, in MDPs with finite state and action spaces, there always exists an optimal stationary policy π^* that satisfies Equation (2.8) for any initial distribution ρ (see [Put14, Theorem 9.1.8]). Such an optimal policy π^*

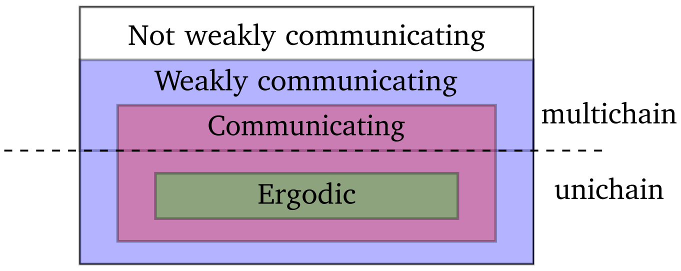


Figure 2.1.: MDP space: the portion of rectangle below the dashed line represents the set of unichain MDPs, and the portion above the dashed line represents the set of multichain MDPs. The blue rectangle is the set of weakly communicating MDPs. The red rectangle on top of the blue one is the set of communicating MDPs. Finally, the green rectangle on top of the red one is the set of ergodic MDPs.

induces the *optimal gain* denoted by $\boldsymbol{g}^*$, and the value of Equation (2.8) is given by $\sum_{s\in\mathcal{S}} \rho(s) g^*(s)$. From [Put14, Chapter 9], the optimal gain $\boldsymbol{g}^*$ satisfies the following system of *modified optimality* equations: for each $s \in \mathcal{S}$,

$$g(s) = \max_{a\in\mathcal{A}_s} \sum_{s'\in\mathcal{S}} p(s' \mid s,a) g(s') \tag{2.14}$$

$$g(s) + h(s) = \max_{a\in\mathcal{A}_s} \Big(r(s,a) + \sum_{s'\in\mathcal{S}} p(s' \mid s,a) h(s') \Big). \tag{2.15}$$

[Put14, Chapter 9] also provides the *optimality* equations for $\boldsymbol{g}^*$, but we do not use them in this thesis. So, we will just call (2.14) and (2.15) the *Bellman optimality* equations.

The optimal gain $\boldsymbol{g}^*$ is uniquely defined by (2.14) and (2.15). Any policies achieving $\boldsymbol{g}^*$ are said to be *gain optimal* (or average reward optimal): for any $\pi \in \Pi$, $g^\pi(s) \le g^*(s)$ for all $s \in \mathcal{S}$. By [Put14, Theorem 8.3.2], if an MDP is weakly communicating, then the optimal gain is state independent: $g^*(s) = g^*(s')$ for any $s, s' \in \mathcal{S}$. So, in weakly communicating MDPs, we just denote the optimal gain by g^*. Note that the fact that the optimal gain is constant over states does not imply that all gain optimal policies are unichain. Table 2.1 describes the gain of stationary policies in different classes of MDPs. Finally, we denote $\boldsymbol{h}^* \in \mathbb{R}^S$ the vector that satisfies (2.15) with $\boldsymbol{g}^*$.

Model class	Optimal gain	Gain of a stationary policy
Ergodic	Constant	Constant
Unichain	Constant	Constant
Communicating	Constant	Possibly nonconstant
Weakly Communicating	Constant	Possibly nonconstant
Multichain	Possibly nonconstant	Possibly nonconstant

Table 2.1.: [Put14, Table 8.3.1]: Relationship between MDP class and gain

Equations (2.14) and (2.15) do not uniquely defined $\boldsymbol{h}^*$. The properties of solution space for $\boldsymbol{h}^*$ given $\boldsymbol{g}^*$ is studied in [SF78].

Reinforcement Learning

3

In the previous chapter, we used the formalism of MDPs to describe how a decision maker interacts with its environment. Depending on the chosen optimality criterion, we explained how an optimal policy could be obtained when the parameters of the MDP are fully known. In this chapter, we consider the case when the MDP's parameters are **unknown** and need to be learned by the decision maker via trial and error. Also, we will use the term "learner" instead of[1] "decision maker".

Section 3.1 describes the existing settings in the reinforcement learning (RL) framework. Then, we present the learning setting considered in this thesis and the performance measure, namely the regret. In Section 3.4, we give the regret lower bound for any learning algorithms. We finish the chapter by presenting two learning approaches known as optimism in face of uncertainty principle and posterior sampling in Section 3.5. We discuss several algorithms with regret guarantee at the end of this chapter.

The reader can skip this chapter and go directly to Chapter 4 and Part II and return to this chapter before diving into Part III.

3.1 A brief summary of existing learning setups

3.1.1 Models and paradigms of learning

Fundamentally, a RL problem consists of a learner interacting with an environment modeled by an MDP $M = \langle \mathcal{S}, \mathcal{A}, r, p \rangle$. The state and action spaces $\mathcal{S}$ and $\mathcal{A}$ are known to the learner, but the expected value of reward r and state transition probability p are **unknown**. So, we say that the MDP M is unknown to the learner. The goal of the learner is to identify an optimal policy π^* by interacting with the unknown MDP.

In the literature of RL, there are mainly two distinguishable learning models:

[1]We use the term "decision maker" when the parameters are known and "learner" when the parameters are unknown.

(i) *Generative model*: the learner chooses the state of the unknown MDP M at the desire

(ii) *Navigating model*: the learner has no control over the state of M.

With the generative model, the learner picks any state-action pair (s, a), and the MDP M incurs a random reward u, whose expected value is $r(s, a)$, and the next state $s' \in \mathcal{S}$ with probability $p(s' \mid s, a)$. The learner collects the sample $\{u, s'\}$ related to the pair (s, a), and the process is repeated. This mechanism is often used in *offline learning paradigm* (see e.g., [LGR12; Lev+20]), in which the learner starts by collecting samples until a budget resource fixed by the setting is exhausted (it can be a time resource, sample resource or approximation error). Then, the learner plans a policy based on the collected samples and follows the policy. No more policy update is made thereafter.

With the navigating model, the learner observes the current state s of the MDP M and executes an action a. The MDP M incurs a random reward with the expected value $r(s, a)$ and transitions to state s' with probability $p(s' \mid s, a)$. The learner cannot force the MDP to restart in any state. If the learner wants the MDP to be in a certain state, it has to go along the trajectory that brings the MDP from its current state to the desired state. In some settings like the finite horizon or infinite horizon with discount, the MDP M can restart, but the learner has no control over the restart, such as when and in which state to restart. This mechanism is used in *online learning paradigm* in which the learner keeps updating its policy based on the observations it collects via the interaction with the MDP M (see e.g., [JOA10; ORV13; AOM17; Ouy+17; ZB19]). The policy update can be done in every decision time –known as the real-time update–, periodically, or when some conditions are met –known as the *episodic update*.

3.1.2 Classification of learning algorithms

In either paradigm described above, there are two types of learning

- *model-free*: the learner tries to infer the value functions in the MDP;
- *model-based*: the learner tries to infer the unknown parameters r and p of the MDP.

The typical algorithm for model-free methods is **Q-learning** [Wat89], which tries to infer the optimal state-action value function via stochastic approximation. Model-free method is very appealing when the MDP has a large or continuous state and action spaces (see e.g., [Mni+15; BDM17; Dab+18]). However, model-free methods are generally slower than model-based ones in learning an optimal policy.

Model-based algorithms infer the value of r and p. They input the estimates of r and p into a planning method described in the previous chapter, such as backward induction, policy or value iterations, and follow the policy output by the planning method (see e.g., [JOA10; ORV13; AOM17]). The main difference within model-based algorithms is the way r and p are estimated at each planning phase. Model-based algorithms are costly for MDPs with large or continuous state and action spaces, but when applicable, they are usually faster than model-free algorithms.

Last but not least, for both model-free and model-based methods, statistical inference provides two perspectives on the unknown MDP:

- *frequentist perspective*, in which all quantities related to the unknown MDP, such as r, p, value function, etc, are seen as **unknown deterministic** quantities,
- *Bayesian perspective*, in which all quantities related to the unknown MDP, such as r, p, value function, etc, are seen as a realization of **random variables**.

In frequentist school, the planning is decided before the learning begins, and the observations collected during the learning are used to control **the probability that the plan is correct** (see e.g., [JOA10; AOM17; Jin+18; Shi+22]). In Bayesian school, the observations collected during the learning are used to **update the posterior of the random variables**. The planning is then done based on the posterior belief (see e.g., [ORV13; Ouy+17; BDM17; Dab+18]).

3.2 Learning setup studied in this thesis

In this thesis, the learner interacts with an MDP $M = \langle \mathcal{S}, \mathcal{A}, r, p \rangle$, where the state space $\mathcal{S}$ of size S and action space $\mathcal{A}$ of size A are known, but the expected value of reward r and state transition probability p are unknown. Also, for any state-action pair (s, a), the expected reward is bounded[2] $r(s, a) \in [0, 1]$. At time step $t \geq 1$, the MDP is in the state s_t, and the learner executes an action a_t. The MDP incurs a random reward denoted by r_t and transitions to the next state denoted by s_{t+1}. This mechanism is repeated, and the learner observes a sequence of the form $\{s_1, a_1, r_1, s_2, \ldots, s_t, a_t, r_t, s_{t+1}, \ldots\}$, which is called *observations* (also known as a "trajectory"). At time step t, the learner has in its disposition the observations

[2]in general, the expected reward is bounded in $[0, r_{max}]$, where $r_{max} \geq 1$, but the interval can be simply scaled down to $[0, 1]$.

up to time t denoted by $o_t := \{s_1, a_1, r_1, s_2, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t\}$. The learner's objective is to maximize the expected cumulative reward $\mathbb{E}\left[\sum_{t\geq 1} r_t\right]$ incurred by the MDP by identifying an optimal policy π^* as early as possible. When the context is clear, we will use the term "algorithm" to mean the learner.

This thesis focuses on **model-based** algorithms with **episodic policy update** in online learning. We present the fundamental structure of episodic learning algorithms in Algorithm 1. Basically, the algorithm takes the state space $\mathcal{S}$ and action space $\mathcal{A}$ of the MDP M as input. The initial state s_1 of M is drawn from some categorical distribution over the state space $\mathcal{S}$. Then, state s_1 is revealed to the learning algorithm. The algorithm computes a policy denoted by π^1, and episode 1 begins. During episode $k \geq 1$, the algorithm executes action $a_t = \pi^k(s_t)$ and collects observations $\{r_t, s_{t+1}\}$. This process repeats until some specific conditions depending on the setting are met. The episode k then terminates. Then, the algorithm computes a new policy π^{k+1} based on its observations, and episode $k+1$ begins.

Algorithm 1: Episodic learning algorithms

```
Input  : The MDP $M$ with state space $\mathcal{S}$ and action space $\mathcal{A}$
1 Set $t = 1$ and observe initial state $s_1$
2 for episodes $k = 1, 2, \ldots$ do
3     Set $t^k = t$
4     Compute a new policy $\pi^k$
5     while terminal condition is not met do
6         Execute action $a_t = \pi^k(s_t)$
7         Observe $r_t$ and next state $s_{t+1}$
8         $t \leftarrow t + 1$.
```

3.3 Regret definition

In online learning, there are at least two performance metrics: (1) probably approximately correct (PAC) bounds on the sample complexity (see e.g., [BT02; KS02; Kak03; DB15; JA18; Wan+20]) and (2) regret (see e.g., [JOA10; ORV13; AOM17; Jin+18; ZB19; ZJ19]). The recent works often provide performance guarantees in terms of both metrics (see e.g., [HZG21a; ZJD21]).

In this thesis, we measure the learner's performance using a notion of *regret* that compares the expected cumulative reward of an optimal policy π^* to the cumulative reward of the learner.

If a learner $\mathcal{L}$ interacts with the unknown MDP M over $T \in \mathbb{N}^+$ time steps, we denote by $\mathrm{Regret}(\mathcal{L}, M, T)$ the regret suffered by the learner. So, maximizing the expected cumulative reward is equivalent to minimizing the expected regret, $\mathbb{E}\left[\mathrm{Regret}(\mathcal{L}, M, T)\right]$.

The formal definition of regret depends on the setting of the learning problem. In this section, we recall the definition in two settings: the finite horizon and infinite horizon average reward criterion.

3.3.1 Finite horizon setting

In this setting, the learner interacts with the unknown MDP M over $K \in \mathbb{N}^+$ episodes, each episode lasts in $H \in \mathbb{N}^+$ time steps (so the total time steps is $T = KH$), and the state of M is reset according to a distribution ρ. The terminal condition in Line 5 of Algorithm 1 is simply $t - t^k = H$. In such a setting, H is called *horizon*, and it is bounded in $\mathbb{N}^+$. At the beginning of each episode $k \geq 1$, the initial state of the MDP $s_{t^k} \sim \rho$ is revealed to the learner who computes a policy π^k based on its collected observations and follows the policy π^k during the episode. Following the definition in Section 2.2, we denote by $w^{\pi^k}_{1:H}(s) := \mathbb{E}^{\pi^k}[\sum_{t=1}^{H} r_t \mid s_1 = s]$ the expected cumulative reward from state s over time steps 1 to H when following policy π. The regret is defined by the following.

Definition 3.1
If a learner $\mathcal{L}$ interacts with an unknown MDP M over K episodes, each episode ends in $H \in \mathbb{N}^+$ time steps, then its regret is

$$\mathrm{Regret}(\mathcal{L}, M, K) := \sum_{k=1}^{K} \sum_{s \in \mathcal{S}} \rho(s)[w^{*}_{1:H}(s) - w^{\pi^k}_{1:H}(s)]. \tag{3.1}$$

We say that a learner is "good" if its expected regret is sublinear in the number of episodes K, i.e., $\mathbb{E}\left[\mathrm{Regret}(\mathcal{L}, M, K)\right] = o(K)$ when $K \to +\infty$.

3.3.2 Infinite horizon average reward criterion

For this setting, there is no reset on the MDP state, but the unknown MDP M is assumed to be *weakly communicating*. This assumption is required so that the optimal gain in M is state independent. The learner interacts with M over T time steps and collects a sequence of rewards $\{r_t\}_{1 \leq t \leq T}$. The learner specifies the terminal condition at Line 5 of Algorithm 1 depending on how often it wants to update its policy. The regret is defined by the following.

Definition 3.2
If a learner $\mathcal{L}$ interacts with an unknown MDP M which is weakly communicating, then its regret after T time steps is

$$\text{Regret}(\mathcal{L}, M, T) := Tg^* - \sum_{t=1}^{T} r_t, \tag{3.2}$$

where g^* is the optimal gain of M (see its definition in Section 2.4).

So, a learner is “good” if its expected regret is sublinear in the total number of time steps T, i.e., $\mathbb{E}\Big[\text{Regret}(\mathcal{L}, M, T)\Big] = o(T)$ when $T \to +\infty$.

3.4 Minimax regret lower bound

We compare the performance of good learners by how fast their regret tends to zero. [JOA10] has proven that no learners can achieve a regret that is smaller than a lower bound in all MDPs. We will give this lower bound below, but first, we need to introduce the notion of the diameter of an MDP.

Definition 3.3
The diameter of an MDP is defined by

$$D := \max_{s,s' \in \mathcal{S}} \min_{\pi: \mathcal{S} \mapsto \mathcal{A}} \mathbb{E}^{\pi}[\tau(s') \mid s_1 = s] - 1, \tag{3.3}$$

where $\tau(s') := \inf\{t \geq 2 : s_t = s'\}$ is the first time step when s' is reached.

So, the diameter of an MDP is the length of the **longest shortest path** in the MDP, where the distance between any pair of vertices is 1. In other words, it is the expected number of vertices along the shortest path between two states that are the **most distant** from each other. From Definition 2.2 and [Put14, Proposition 8.3.1], it is clear that the diameter D is finite if and only if the MDP is communicating.

Given any learner $\mathcal{L}$, there always exists an unknown MDP M that slows down the learner $\mathcal{L}$ as the following.

Proposition 3.4 ([JOA10, Theorem 5])
In infinite horizon setting, for any learner $\mathcal{L}$, any integers S, $A \geq 10$, $D \geq 20\log_A(S)$, and $T \geq DSA$, there is an MDP $M = \langle \mathcal{S}, \mathcal{A}, r, p\rangle$ whose diameter is D such that for any initial state s_1, the expected regret of $\mathcal{L}$ after T time steps is lower bounded as

$$\mathbb{E}\left[\mathrm{Regret}(\mathcal{L}, M, T)\right] \geq 0.015\sqrt{DSAT}. \tag{3.4}$$

This proposition means that no matter how "good" a learner is, it is always possible to construct a worst-case MDP M having S states, A actions and a diameter D such that the learner suffers a regret $\Omega(\sqrt{DSAT})$ after T time steps in M. This bound on worst-case regret is often referred to as "minimax" bound.

The minimax bound differs from the bounds given in [OPT18; BK97], which are problem-dependent and asymptotic. Minimax bounds usually scale as $\sqrt{T}$ while problem-dependent bounds scale logarithmically[3] with T. We refer to [OPT18; BK97] for more detail about these problem-dependent bounds.

Minimax bound is often expressed in terms of the span (or range) of the optimal bias function as well. In fact, the specific worst-case MDP constructed by [JOA10] to prove the lower bound in Proposition 3.4 satisfies $D = 2sp(\boldsymbol{h}^*)$, where $\boldsymbol{h}^*$ is the optimal bias function (see Section 2.4 for its definition). If H is an upper bound on $sp(\boldsymbol{h}^*)$, then the minimax bound is also expressed as $\Omega(\sqrt{HSAT})$. In recent work of [ZJ19], the proposed algorithm, albeit no efficient implementation is given, achieves a regret upper bounded by $\tilde{O}(\sqrt{HSAT})$ (see [ZJ19, Theorem 1]). This result suggests that the minimax bound in Proposition 3.4 cannot be improved.

For the finite horizon setting, the diameter of the MDP or the upper bound on the span of the optimal bias function is replaced with the horizon of an episode. So, the minimax bound can be immediately derived from Proposition 3.4: for any learner, it is always possible to construct a worst-case MDP with S states and A actions such that after K episodes, each of horizon[4] H, the learner suffers a regret[5] $\Omega(H\sqrt{SAK})$.

[3] In the bandit literature, "problem-dependent" bounds are said to be distribution-dependent, as opposed to minimax bounds which are said to be distribution-free [GMS19]

[4] For stochastic bandit, we have $H = 1, S = 1$ and $K = T$. So, the minimax bound is $\Omega(\sqrt{AT})$ as given in [BC12]

[5] In time-inhomogeneous finite horizon setting, the minimax bound is $\Omega(H\sqrt{HSAK})$ (see, e.g., [Jin+18; Dom+21]).

3.5 Algorithms with regret guarantee

To get enough information for deriving π^*, the learner needs to *explore* the dynamic of the MDP as much as possible. However, too much exploration equalizes the learner's performance with one that blindly chooses action at random. So, the regret is linear in the total number of time steps. A good learner should then *exploit* the gathered information as soon as possible. Unfortunately, untimely exploitation leads to suboptimal policy; thus, a regret that is linear in T. This is the famous "exploration versus exploitation dilemma" in RL problem.

To manage the exploration-exploitation dilemma, the two perspectives from statistical inference mentioned in Section 3.1.2 are adopted. In a frequentist perspective, the learner $\mathcal{L}$ that maximizes the expected cumulative reward over T time steps equivalently minimizes its expected regret $\mathbb{E}\Big[\mathrm{Regret}(\mathcal{L}, M, T)\Big]$. To achieve this goal, a common strategy in the frequentist paradigm is to apply the *optimism in face of uncertainty* (OFU) principle: the learner maintains a confidence set for the unknown MDP M and executes an optimal policy of the "best" MDP in the confidence set (it is the best in terms of gain, or value function, etc.), e.g., [JOA10; FCG10; BT12; AOM17; Fru+17; Jin+18; Fru+18; FPL18; ZB19; ZJ19; BMT20; Ort20].

In a Bayesian perspective, we say that the unknown MDP M is drawn according to a probability distribution (prior or posterior) ϕ. The learner minimizes a similar notion of regret known as the *Bayesian regret* (or Bayes risk) defined by

$$\mathrm{BayesRegret}(\mathcal{L}, \phi, T) := \mathbb{E}\Big[\mathbb{E}\Big[\mathrm{Regret}(\mathcal{L}, M, T) \mid M\Big]\Big], \tag{3.5}$$

where ϕ is the prior distribution of the unknown MDP M. In the Bayesian paradigm, an efficient exploration-exploitation trade-off can be done by posterior sampling introduced by [Tho33]: the learner keeps a posterior distribution over possible MDPs (precisely, the support of the prior distribution) and executes an optimal policy of a sampled MDP, see e.g., [ORV13; GM15; Ouy+17].

We summarize several existing results about minimizing the regret in Table 3.1 for finite horizon setting and Table 3.2 for infinite horizon average reward criterion. The results in Table 3.1 are a few because the minimax lower bound in the finite horizon setting is achieved relatively early (ignoring the logarithmic terms). However, there are a lot of works for finite horizon models such as [Jin+18; Dom+21; Li+21] in which the reward and state transition depend on the time step (also known as time-inhomogeneous, or non-stationary MDP) or [ZB19; ZJD21] in which the total reward over H time steps is bounded by 1.

The quest to the minimax regret lower bound for the infinite horizon setting is longer compared to the finite horizon setting. With the average reward criterion, the structure of the unknown MDP is vital. So, the works in Table 3.2 make different assumptions on the MDP depending on the nature of the proposed algorithms. In addition, policy computation is also a critical aspect of the infinite horizon setting. Some algorithms in Table 3.2 only have a theoretical regret guarantee and cannot be implemented efficiently. Precisely, either the implementation of those algorithms takes an unreasonably long time to run or there is no possible implementation. So, we say that they are intractable. However, the algorithms that use value iteration, extended value iteration, or modified extended value iteration are implementable, and their computational complexity is $\mathcal{O}(S^2A)$ [JOA10]. Finally, Bayesian algorithms like PSRL and TSDE use planning methods described in Chapter 2 for policy computation. It might seem that these algorithms are computationally "efficient", but they are "easily implementable" if the conjugate prior has a closed-form expression. Otherwise, a sophisticated implementation might be required for the sampling method such as Markov Chain Monte Carlo [And+03], Sequential Monte Carlo [DJ+09], and Variational Inference [BKM17].

Algorithm	Regret
PSRL [ORV13]	$\tilde{\mathcal{O}}(HS\sqrt{AT})$
UCBVI-BF [AOM17]	$\tilde{\mathcal{O}}(\sqrt{HSAT})$
EULER [ZB19]	$\tilde{\mathcal{O}}(\sqrt{HSAT})$
Lower bound	$\Omega(\sqrt{HSAT})$ [JOA10]

Table 3.1.: The quest to the minimax regret lower bound for model-based RL algorithms in finite horizon setting with S states, A actions, and $T = KH$ steps. K is the total number of episodes and H is horizon of each episode.

3.6 Overview of regret analysis

As mentioned in Section 3.2, we consider **model-based** algorithms with **episodic policy update**. Algorithm 1 shows how episodic algorithms work during learning. For both optimism and posterior sampling methods, computing a new policy at Line 4 of Algorithm 1 is composed of two steps

(i) compute an estimate $M^k = \langle \mathcal{S}, \mathcal{A}, r^k, p^k \rangle$ of the unknown MDP M;

(ii) compute an optimal policy π^k of M^k.

Algorithm	Regret	Assump. on M	Policy comp.
UCRL2 [JOA10]	$\tilde{O}(DS\sqrt{AT})$	comm.	EVI
REGAL [BT12]	$\tilde{O}(HS\sqrt{AT})$	weakly comm.	Intractable
TSDE [Ouy+17]	$\tilde{O}(HS\sqrt{AT})$	weakly comm.	VI
SCAL [Fru+18]	$\tilde{O}(HS\sqrt{AT})$	weakly comm.	modified EVI
OSP [Ort20]	$\tilde{O}(\sqrt{t_{mix}SAT})$	ergodic	Intractable
UCRL2B [FPL20]	$\tilde{O}(\sqrt{\Gamma DSAT})$	comm.	EVI
EBF [ZJ19]	$\tilde{O}(\sqrt{HSAT})$	weakly comm.	Intractable
Lower bound	$\Omega(\sqrt{DSAT})$ [JOA10]		

Table 3.2.: The quest to the minimax regret lower bound for model-based RL algorithms in infinite horizon average reward model with S states, A actions, and T steps. D is the diameter of the MDP, $H \geq sp(\boldsymbol{h}^*)$ is the upper bound on the span of the optimal bias function, t_{mix} is the mixing time of the MDP and Γ is the upper bound on the number of the next possible states. EVI stands for Extended value iteration [JOA10], VI for value iteration, assump. for assumption, comm. for communicating, and comp. for computation. Intractable here means that there is no efficient implementation.

We will say that M^k is the imagined version of the unknown M for episode k. Since an optimal policy can be computed in a deterministic manner given the MDP, the "high-level" difference between both approaches lies in Step (i) at which the imagined MDP M^k is constructed: chosen "optimistically" by the OFU method or chosen randomly by posterior sampling.

In the following, we provide an overview of regret analysis in the finite horizon setting. The analysis will involve the value function associated with different MDPs. So, we extend the notation in Chapter 2 as the following

- the optimal expected cumulative reward from the unknown MDP M between time steps h and H is denoted by $w^{\pi^*}_{M,h:H}$, where π^* is an optimal policy in M;
- the optimal expected cumulative reward from the imagined MDP M^k between time steps h and H is denoted by $w^{\pi^k}_{M^k,h:H}$, where π^k is an optimal policy in M^k.

By Definition 3.1, the regret of a learner $\mathcal{L}$ after K episodes in an unknown MDP M is

$$\text{Regret}(\mathcal{L}, M, K) = \sum_{k=1}^{K} \sum_{s \in \mathcal{S}} \rho(s)[w^{\pi^*}_{M,1:H}(s) - w^{\pi^k}_{M,1:H}(s)].$$

The term $\boldsymbol{w}_M^{\pi^*} - \boldsymbol{w}_M^{\pi^k}$ can be rewritten as

$$\boldsymbol{w}_M^{\pi^*} - \boldsymbol{w}_M^{\pi^k} = \underbrace{(\boldsymbol{w}_M^{\pi^*} - \boldsymbol{w}_{M^k}^{\pi^k})}_{=:\Delta^k_{model}} + \underbrace{(\boldsymbol{w}_{M^k}^{\pi^k} - \boldsymbol{w}_M^{\pi^k})}_{=:\Delta^k_{conc}} \tag{3.6}$$

The first term Δ^k_{model} is the difference between the real but **unknown** optimal value function and the **imagined** optimal value function. It is also understood as an error due to model misspecification. Since M and π^* are unknown, this first term is hard to be analyzed. However, we will see in the following that the term is **non-positive** for the OFU methods or **zero in expectation** for posterior sampling.

The second term Δ^k_{conc} can be interpreted as the "dissimilarity" between the **unknown** MDP M and the **imagined** MDP M^k along the trajectory induced by the policy π^k. In other words, it is the discordance between the value from M^k and the value from M for policy π^k. Since M^k and π^k are chosen by the learner, Δ^k_{conc} can be analyzed. OFU and posterior sampling methods use concentration argument to bound this term Δ^k_{conc}.

3.7 Sketch of proof on regret bound of three algorithms

In this section, we revisit three algorithms: UCRL2 [JOA10] and UCBVI [AOM17], which rely on the OFU method, and PSRL [ORV13], which is a posterior sampling algorithm. We give the sketch of proof for their regret upper bound in the finite horizon setting, although UCRL2 was originally designed for the infinite horizon average reward criterion. We will provide detailed proof in Chapter 7 when we adapt the three algorithms to Markovian bandit problems.

3.7.1 Upper confidence bound reinforcement learning (UCRL2) [JOA10].

As mentioned above, UCRL2 was originally designed for the infinite horizon setting. Here, we adapt UCRL2 to the finite horizon setting. When updating its policy, UCRL2 constructs confidence sets $\mathbb{B}_r$ and $\mathbb{B}_p$ for r and p based on high probability confidence bounds. We denote by $\mathbb{M}$ the set of plausible MDPs whose reward function lives in $\mathbb{B}_r$ and state transition lives in $\mathbb{B}_p$. UCRL2 chooses the MDP that incurs the highest optimal value among the MDPs in $\mathbb{M}$ and computes an optimal policy of the chosen MDP. That is, at Line 4 of Algorithm 1, UCRL2 computes a new policy π^k such that

$$\boldsymbol{w}_{M^k}^{\pi^k} \geq \max_{\pi \in \Pi} \max_{M' \in \mathbb{M}^k} \boldsymbol{w}_{M'}^{\pi}, \tag{3.7}$$

where

$$\mathbb{M}^k := \Big\{ \langle \mathcal{S}, \mathcal{A}, r', p' \rangle : \text{ for all } (s,a), r'(s,a) \in \mathbb{B}_r^k(s,a) \text{ and } \boldsymbol{p}'(\cdot \mid s,a) \in \mathbb{B}_p^k(s,a) \Big\}$$

is the set of plausible MDPs compatible with the confidence sets for episode k. To do so, UCRL2 uses extended value iteration (EVI). This is because EVI is designed such that (3.7) holds. We refer to [JOA10] for the detail about EVI and its convergence proof.

Let t^k be the time step when the episode k begins and $t^1 := 1$ (see Algorithm 1). The observations up to time t^k is denoted by $o_{t^k}{:=}\{s_1, a_1, r_1, \dots, s_{t^k-1}, a_{t^k-1}, r_{t^k-1}, s_{t^k}\}$. For all state-action pair (s,a), let $N^k(s,a)$ be the number of times up to t^k that the algorithm executes action a when the MDP is in state s. At time step t^k, UCRL2 uses the observations o_{t^k} to compute $\hat{r}^k$ and $\hat{p}^k$, the empirical means of r and p. The confidence sets for reward and transition are then computed: for all state-action pair (s,a),

$$\begin{aligned}
\mathbb{B}_r^k(s,a) &:= \Big\{ u \in [0,1] : \Big| u - \hat{r}^k(s,a) \Big| \leq \beta_r^k(s,a) \Big\}, \\
\mathbb{B}_p^k(s,a) &:= \Big\{ \boldsymbol{q} \in [0,1]^S : \Big\| \boldsymbol{q} - \hat{\boldsymbol{p}}^k(\cdot \mid s,a) \Big\|_{\ell_1} \leq \beta_p^k(s,a) \text{ and } \|\boldsymbol{q}\|_{\ell_1} = 1 \Big\},
\end{aligned}$$

where $\beta_r^k(s,a) = \sqrt{\dfrac{c_r \ln(SAt^k)}{2\max\big(1, N^k(s,a)\big)}}$ and $\beta_p^k(s,a) = \sqrt{\dfrac{c_p S \ln(At^k)}{\max\big(1, N^k(s,a)\big)}}$ are the confidence bonuses, and c_r and c_p are constants to be chosen accordingly before the learning.

The set of plausible MDPs $\mathbb{M}^k$ is constructed to contain the unknown M with high probability. Hence, by (3.7), $\boldsymbol{w}_{M^k}^{\pi^k} \geq \boldsymbol{w}_M^{\pi^*}$ with high probability. In consequence, Equation (3.6) implies that the regret of UCRL2 is smaller than $\sum_k \sum_s \rho(s)[\Delta_{conc}^k(s)]$ with high probability. Using Bellman evaluation equation, the term $\Delta_{conc}^k(s)$ can be rewritten as

$$\begin{aligned}
w_{M^k,1:H}^{\pi^k}(s_1) - w_{M,1:H}^{\pi^k}(s_1) = r^k(s_1,a_1) &+ \sum_{s' \in \mathcal{S}} p^k(s' \mid s_1, a_1) w_{M^k,2:H}^{\pi^k}(s') \\
&- r(s_1,a_1) - \sum_{s' \in \mathcal{S}} p(s' \mid s_1, a_1) w_{M,2:H}^{\pi^k}(s'),
\end{aligned} \tag{3.8}$$

where $a_1 = \pi_1^k(s_1)$ and $a_t = \pi_t^k(s_t)$ for any $1 \leq t \leq H$. The term $\boldsymbol{p}^k \boldsymbol{w}_{M^k}^{\pi^k} - \boldsymbol{p}\boldsymbol{w}_M^{\pi^k}$ equals $(\boldsymbol{p}^k - \boldsymbol{p})\boldsymbol{w}_{M^k}^{\pi^k} + \boldsymbol{p}(\boldsymbol{w}_{M^k}^{\pi^k} - \boldsymbol{w}_M^{\pi^k})$. Since for any $h \leq H$ and $s \in \mathcal{S}$, $w_{M^k,h:H}^{\pi^k}(s) \in [0, H]$ is not deterministic, the term $(\boldsymbol{p}^k - \boldsymbol{p})\boldsymbol{w}_{M^k}^{\pi^k}$ is bounded using Hölder's inequality. So, we get

$$\Delta_{conc}^k(s_1) \leq \sum_{t=1}^{H} \left| r^k(s_t, a_t) - r(s_t, a_t) \right| + H \left\| \boldsymbol{p}^k(\cdot \mid s_t, a_t) - \boldsymbol{p}(\cdot \mid s_t, a_t) \right\|_{\ell_1} + d_t, \quad (3.9)$$

where $d_{t-1} = \boldsymbol{p}(\cdot \mid s_{t-1}, a_{t-1})\left(\boldsymbol{w}_{M^k,t:H}^{\pi^k} - \boldsymbol{w}_{M,t:H}^{\pi^k}\right) - \left(w_{M^k,t:H}^{\pi^k}(s_t) - w_{M,t:H}^{\pi^k}(s_t)\right)$. So, $\{d_t\}_{t \geq 1}$ is a martingale difference sequence, each term upper bounded by H. The sum $\sum_{t \geq 1} d_t$ can be bounded using Azuma-Hoeffding's inequality. Moreover, if the unknown M belongs to the plausible set $\mathbb{M}^k$, then $\left| r^k - r \right|$ and $\left\| \boldsymbol{p}^k - \boldsymbol{p} \right\|_{\ell_1}$ can be bounded by the confidence bonuses β_r^k and β_p^k. So, the regret of UCRL2 is bounded.

For the sake of completeness, we recall the result of [JOA10] for the infinite horizon setting below.

Proposition 3.5 ([JOA10, Theorem 2])
For any communicating MDP M with S states, A actions and diameter D, with probability at least $1 - \delta$, it holds that for any $T > 1$, the regret of UCRL2 is bounded by

$$\mathrm{Regret}(\mathrm{UCRL2}, M, T) \leq 34DS\sqrt{AT \ln\left(\frac{T}{\delta}\right)}.$$

Compared to Proposition 3.4, the upper bound in Proposition 3.5 is loose by a factor $\sqrt{DS}$ ignoring the logarithmic term.

3.7.2 Upper confidence bound value iteration (UCBVI) [AOM17].

UCBVI is an optimistic algorithm designed for the finite horizon setting. The algorithm keeps track of the empirical mean $\hat{p}$ of p and only constructs confidence set $\mathbb{B}_r$ for r based on a high probability confidence bound. Differently from UCRL2, which chooses only one copy of the expected reward, UCBVI executes Line 4 of Algorithm 1 by choosing H copies in the following manner:

- set $w_{H+1:H,M^k}^{\pi^k}(s) = 0$ for all s,
- for h from H to 1

- choose $r'(s,a) \in \mathbb{B}_r^k(s,a)$ for all (s,a) such that for all s,

$$w_{h:H,M^k}^{\pi^k}(s) = \min\left\{H, \max_{a\in\mathcal{A}}\left(r'(s,a) + \sum_{s'\in\mathcal{S}} \hat{p}^k(s' \mid s,a) w_{h+1:H,M^k}^{\pi^k}(s')\right)\right\} \tag{3.10}$$

- set $\pi_h^k(s) = a_h$ where a_h is one of the actions that achieve the maximum of (3.10).

The confidence sets for reward are defined by: for all state-action pair (s,a) and all time step $1 \le h \le H$,

$$\mathbb{B}_r^k(s,a) := \left\{u \in [0,H] : \left|u - \hat{r}^k(s,a)\right| \le \beta_r^k(s,a)\right\}$$

where $\beta_r^k(s,a) = H\sqrt{\frac{c_r \ln(SAt^k)}{2\max\left(1, N^k(s,a)\right)}}$ is the confidence bonus, and c_r is a constant to be chosen accordingly before the learning.

It is shown in [AOM17] that $\boldsymbol{w}_M^{\pi^*} - \boldsymbol{w}_{M^k}^{\pi^k} \le \boldsymbol{0}$ holds with high probability. By Equation (3.6), the regret of UCBVI is bounded by $\sum_k \sum_s \rho(s)[\Delta_{conc}^k(s)]$ with high probability. The theoretical novelty in UCBVI is to efficiently deal with the term $(\hat{\boldsymbol{p}}^k - \boldsymbol{p})\boldsymbol{w}_{M^k}^{\pi^k}$ when rewriting the right term of (3.8). Indeed, $(\hat{\boldsymbol{p}}^k - \boldsymbol{p})\boldsymbol{w}_{M^k}^{\pi^k}$ is also rewritten as $(\hat{\boldsymbol{p}}^k - \boldsymbol{p})(\boldsymbol{w}_{M^k}^{\pi^k} - \boldsymbol{w}_M^{\pi^*}) + \boldsymbol{w}_M^{\pi^*}(\hat{\boldsymbol{p}}^k - \boldsymbol{p})$. Since $\boldsymbol{w}_M^{\pi^*}$ is deterministic, the term $\boldsymbol{w}_M^{\pi^*}(\hat{\boldsymbol{p}}^k - \boldsymbol{p})$ can be bounded using Chernoff-Hoeffding's inequality on individual component $w_M^{\pi^*}(s')\left(\hat{p}^k(s') - p(s')\right)$. Moreover, thanks to the optimism, $w_{M^k}^{\pi^k}(s') - w_M^{\pi^*}(s')$ is non-negative for any $s' \in \mathcal{S}$ with high probability. So, the empirical Bernstein's inequality can be used to upper bound each individual component $\left(\hat{p}^k(s') - p(s')\right)\left(w_{M^k}^{\pi^k}(s') - w_M^{\pi^*}(s')\right)$.

UCBVI enjoys the following worst-case regret guarantee.

Proposition 3.6 ([AOM17, Theorem 1])
In the finite horizon setting with horizon H, for any unknown MDP M with S states and A actions, with probability at least $1-\delta$, it holds that the regret of UCBVI is bounded by

$$\text{Regret}(\text{UCBVI}, M, K) \le 20H^{3/2}\sqrt{SAKL} + 250H^2S^2AL^2$$

where $L := \ln\left(\frac{5SAKH^2}{\delta}\right)$.

So, for $T \geq HS^3A$ and $SA \geq H$, this bound is of order $\tilde{\mathcal{O}}(H\sqrt{SAT})$ where $T = KH$ and $\tilde{\mathcal{O}}(\cdot)$ hides the logarithmic terms. Compared to the lower bound $\Omega(\sqrt{HSAT})$, the bound given in Proposition 3.6 is loose by a factor $\sqrt{H}$ ignoring the logarithmic terms. The advanced version UCBVI-BF, which enjoys a regret bound $\tilde{\mathcal{O}}(\sqrt{HSAT})$, is also given in [AOM17].

3.7.3 Posterior sampling for reinforcement learning (PSRL) [ORV13].

PSRL designed for the finite horizon setting is a posterior sampling algorithm which starts by choosing prior distributions ϕ_r and ϕ_p, such that the unknown r is assumed to be drawn according to ϕ_r, and the unknown p according to ϕ_p. To ease the exposition, we will say that an MDP $M' = \langle \mathcal{S}, \mathcal{A}, r', p' \rangle$ is sampled according to ϕ when r' is sampled according to ϕ_r and p' is sampled according to ϕ_p. When updating its policy, PSRL uses the collected observations and Bayes' theorem to derive the posterior distribution of the unknown MDP. After that, an MDP is sampled according to the posterior, and PSRL computes an optimal policy of the sampled MDP. Precisely, at time step t^k, PSRL uses the observations o_{t^k} to update the posterior $\phi(\cdot \mid o_{t^k})$. Then, Line 4 of Algorithm 1 is performed by sampling an MDP M^k from $\phi(\cdot \mid o_{t^k})$ and computing an optimal policy π^k in M^k. The policy π^k is then used during the episode k to collect more observations.

The performance of PSRL is measured by the Bayesian regret given in (3.5). Broadly speaking, we bound the Bayesian regret by bounding the regret with some deterministic terms, and the expected value of those terms are then themselves. To do so, we will first show that the expected value of Δ^k_{model} defined in (3.6) is zero. Then, the term Δ^k_{conc} is bounded in the same manner as UCRL2 does.

Lemma 3.7
For any $k \geq 1$, let t^k be the time step that episode k starts and o_{t^k} be the observations collected right before t^k. Assume that the unknown MDP M is drawn according to the prior ϕ and that M^k is sampled according to the posterior $\phi(\cdot \mid o_{t^k})$. Then, for any o_{t^k}-measurable function f, one has

$$\mathbb{E}[f(M)] = \mathbb{E}[f(M^k)]. \tag{3.11}$$

Proof. At the start of each episode k, M and M^k are identically distributed conditioned on o_{t^k}. In consequence, if f is o_{t^k}-measurable function, one has:

$$\mathbb{E}[f(M) \mid o_{t^k}] = \mathbb{E}[f(M^k) \mid o_{t^k}].$$

Equation (3.11) then follows from the tower rule. □

This lemma implies that $\mathbb{E}\left[\boldsymbol{w}_{M^k}^{\pi^k}\right] = \mathbb{E}\left[\boldsymbol{w}_{M}^{\pi^*}\right]$ because M^k and π^k are o_{t^k}-measurable. Consequently, $\mathbb{E}\left[\Delta_{model}^k\right] = \mathbf{0}$ for PSRL. The term Δ_{conc}^k can be rewritten as in (3.9). Since d_t is a martingale difference term, its expected value is zero. The term $\left|r^k - r\right|$ and $\left\|\boldsymbol{p}^k - \boldsymbol{p}\right\|_{\ell_1}$ can be bounded using Hoeffding's and Weissman's inequalities.

PSRL enjoys the following Bayesian regret guarantee.

Proposition 3.8 ([ORV13, Theorem 1])
In the finite horizon setting with horizon H, if ϕ is the prior distribution of the unknown MDP M with S states and A actions, then

$$\text{BayesRegret}(\text{PSRL}, \phi, T) = \mathcal{O}\left(HS\sqrt{AT\ln\left(SAT\right)}\right), \tag{3.12}$$

where $T = KH$.

So, this bound is of order $\tilde{\mathcal{O}}(HS\sqrt{AT})$, where $T = KH$, and $\tilde{\mathcal{O}}(\cdot)$ hides the logarithmic terms. It is loose by a factor $\sqrt{HS}$ compared to the lower bound $\Omega(\sqrt{HSAT})$. Note that this bound is for Bayesian regret which means that it is a weaker guarantee compared to the regret bound with high probability of UCRL2 and UCBVI.

4 Markovian Bandit Problem

A multi-armed bandit problem is a sequential decision problem in which, at each time step, the decision maker chooses one among a set of available actions and obtains some incentive. The decision maker's goal is to maximize the total incentive by a sequence of actions. Fundamentally, there are at least **three types** of bandit problems depending on the nature of the incentive: stochastic, adversarial, and Markovian. In this thesis, we focus on Markovian bandits. We refer to [BC12] for more detailed discussion about stochastic and adversarial bandits.

In this chapter, we recall the existing formalizations of Markovian bandit in Section 4.1. Then, we present the rested Markovian bandit problem in Section 4.2. In discounted rested bandit problem, it is well-known that Gittins index policy is optimal. So, we recall its definition in Section 4.2.2. Next, we present the restless Markovian bandit problem in Section 4.3. For infinite horizon setting, there are two possible criteria for restless bandits depending on whether the reward is discounted or not. For both criteria, Whittle index policy is a popular heuristic thanks to its simplicity and strong empirical performance. So, we recall its definition in Section 4.3.2. Finally, in Section 4.4, we point out the questions that arise in the rested and restless bandit problems, and that will be treated in Part II. We also outline the questions arisen when learning with the rested and restless bandit model. These questions will be treated in Part III.

4.1 A brief summary of Markovian bandit formalizations

Markovian bandits form a subclass of multi-armed bandit problems in which each arm has an internal state that evolves in a Markovian manner, as a function of the decision maker's actions.

In *rested* Markovian bandit, each arm is modeled by a finite Markov reward process. At each time step, the decision maker observes the current state of each arm and activates one arm. The activated arm incurs a random reward and transitions to a new state in a Markovian fashion. Meanwhile, the unchosen arms remain in the same state. The name "rested" is adopted because the unchosen arms make no state transitions and incur no rewards. Rested bandit is also known as *restful* bandit or a *family of alternative bandit processes* (see e.g., [Git79; KV87; Duf95; TL10; GGW11]).

In [Whi88], Whittle extended the rested Markovian bandit to a *restless* case in which the unchosen arms also make a state transition and incur some reward (possibly always zero). Up to the current literature, two setups are made for restless bandits: (1) **partially observed** setup and (2) **fully observed** setup.

For partially observed restless bandits, each arm is a finite Markov reward process. At each time step, the decision maker activates one arm and observes its current state. The chosen arm incurs a random reward in a function of its current state. After that, all arms make a state transition. The decision maker only observes the current state of the activated arm. That is why the term "partially observed" is used. This setting is considered, for example, in [AL09; Ort+12; JT19; AM19a; WHL20].

For fully observed restless bandits, each arm is a finite MDP with binary action space. At each time step, the decision maker observes the current state of all arms and activates multiple arms. The activated arms incur random rewards and change state according to their active Markov reward process. The unchosen arms also incur random reward (possibly always zero) and change state according to their passive Markov reward process. This model is considered, for example, in [Whi96; AM19b; GGY20; Dah+22].

Lastly, an even more general model of the fully observed restless bandit is the case where each arm is a finite MDP with multiple actions. At each time step, the decision maker has to decide which action to be executed on each arm without violating the resource constraint. This kind of model is known in the literature as a *restless multi-armed multi-action bandit* (see e.g., [HG15; KPT21]).

Markovian bandit models in this thesis

We consider only **rested** and the **fully observed restless** Markovian bandits. For fully observed restless bandits, we consider the setup with binary action space and will simply call it "restless bandit". For both bandits, we consider the **infinite horizon** setting: the decision maker collects rewards over an infinite number of time steps.

4.2 Rested Markovian bandit

4.2.1 Notations and problem formulation

A rested Markovian bandit is a multi-armed bandit having $n \in \mathbb{N}^+$ arms. Each arm $\langle \mathcal{S}_i, \boldsymbol{r}_i, \boldsymbol{P}_i \rangle$ for $i \in [n]$ is a Markov reward process with a finite state space $\mathcal{S}_i$ of size S_i, a mean reward vector $\boldsymbol{r}_i \in [0,1]^{S_i}$ and a stochastic matrix $\boldsymbol{P}_i$. In state $s_i \in \mathcal{S}_i$, the arm i incurs a random reward with the expected value $r_i(s_i)$ and transitions to a new state s_i' with a probability $P_i(s_i, s_i')$.

The sequential decision problem is the following. At time step 1, the state of all arms denoted by $\boldsymbol{s}_1 := (s_{1,1}, \dots, s_{1,n})$ is sampled according to some initial distribution ρ over the state space $\mathcal{X} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. At time step $t \geq 1$, the decision maker observes the current state of all arms denoted by $\boldsymbol{s}_t := (s_{t,1}, \dots, s_{t,n})$ and activates one arm $i \in [n]$. The activated arm incurs a random reward discounted like $\gamma^{t-1} r_t$ where $\gamma \in (0,1]$ is a discount factor. After that, the activated arm transitions to a new state. The other arms incur no rewards and make no state transitions. The decision maker acts using a policy $\pi : \mathcal{X} \mapsto [n]$.

In rested bandit with no discount, $\gamma = 1$, the decision maker wants to find an optimal policy that maximizes the average reward of any initial state. This setting is well treated in the literature [AVW87; TL10]. With the assumption that each arm is ergodic, an optimal policy is immediately derived: let $\mu_i \in \Delta^{S_i}$ be the state distribution of arm i in its **steady regime**. It is optimal to always activate arm i^* where $i^* \in \arg\max_{i \in [n]} \sum_{s_i \in \mathcal{S}_i} r_i(s_i) \mu_i(s_i)$ [TL10].

In rested bandit with discount, $\gamma \in (0,1)$, the decision maker wants to find an optimal policy π^* that maximizes the expected cumulative discounted reward of any initial state. This is an infinite horizon discounted problem, presented in Section 2.3, in which the rested Markovian bandit is a structured MDP. We discuss more about this setting in the following section.

4.2.2 Discounted rested bandit: Gittins index policy

Since a rested bandit can be viewed as an MDP, it is then possible to find an optimal policy using iterative algorithms such as value iteration. However, those algorithms have a computational complexity that is polynomial in the MDP state size, thus, exponential in the number of arms. This makes them prohibitive for rested bandits with a large number of arms. So, efficiently computing an optimal policy π^* in rested bandits with discount was an open problem from the 1940s to 1970s [Whi96]. However, [Git79] had solved the problem in about 1970 by proposing an index policy.

Proposition 4.1 ([Whi96, Theorem 14.3.3])
In any rested Markovian bandit with a discount factor $\gamma \in (0,1)$ and $n \in \mathbb{N}^+$ arms, each with state space $\mathcal{S}_i$ for $i \in [n]$, there exists an index function that associates each state $s_i \in \mathcal{S}_i$ of each arm $i \in [n]$ with a real number $\mathrm{GittinsIndex}(s_i)$ where

$$\mathrm{GittinsIndex}(s_i) := \sup_{\tau>0} \frac{\mathbb{E}\left[\sum_{t=1}^{\tau} \gamma^{t-1} r_t \mid s_{1,i} = s_i\right]}{\mathbb{E}\left[\sum_{t=1}^{\tau} \gamma^{t-1} \mid s_{1,i} = s_i\right]}, \tag{4.1}$$

and $\{r_t\}_{1 \le t \le \tau}$ is the sequence of rewards incurred uniquely by arm i when it makes state transitions over time steps 1 to τ, where τ is the stopping time. The policy $\pi : \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \mapsto [n]$ where for all $\boldsymbol{s} \in \mathcal{X}$

$$\pi(\boldsymbol{s}) \in \arg\max_{i \in [n]} \mathrm{GittinsIndex}(s_i)$$

is optimal.

The index of state s_i is now called Gittins index, and the policy in Proposition 4.1 is called Gittins index policy. Gittins index policy from [Git79] is a breakthrough because the index value of state $s_i \in \mathcal{S}_i$ given in (4.1) depends only on the local parameters $\{\boldsymbol{r}_i, \boldsymbol{P}_i\}$ of arm i. Thus, the computational complexity of Gittins index policy is linear in the number of arms, and solving discounted rested bandit problem is broken down into computing Gittins index of each state of each arm.

Note that Gittins index policy is optimal when exactly one arm is activated at each time step. This index policy is suboptimal when the decision maker activates more than one arm at each time step or when the decision maker interact with the bandit over a finite horizon [GGW11].

4.3 Restless Markovian bandit

4.3.1 Notations and problem formulation

A restless Markovian bandit is a multi-armed bandit having n arms. Each arm $\langle \mathcal{S}_i, \{0,1\}, \{\boldsymbol{r}_i^0, \boldsymbol{r}_i^1\}, \{\boldsymbol{P}_i^0, \boldsymbol{P}_i^1\} \rangle$ is an MDP with a finite state space $\mathcal{S}_i$ of size S_i and a binary action space $\{0, 1\}$, where 0 denotes the action "rest" and 1 denotes the action "activate". If arm i is in state s_i and the decision maker executes $a_i \in \{0, 1\}$, the arm incurs a random reward with the expected value $r_i^{a_i}(s_i)$ and transitions to a new state $s'_i \in \mathcal{S}_i$ with a probability $P_i^{a_i}(s_i, s'_i)$.

The sequential decision problem is presented as the following. At time step 1, the state of all arms denoted by $\boldsymbol{s}_1 := (s_{1,1}, \dots, s_{1,n})$ is sampled according to some initial distribution ρ over the state space $\mathcal{X} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. At time step $t \geq 1$, the decision maker observes the current state of all arms denoted by $\boldsymbol{s}_t := (s_{t,1}, \dots, s_{t,n})$ and activates exactly m arms encoded by action $\boldsymbol{a}_t := (a_{t,1}, \dots, a_{t,n})$, such that $\boldsymbol{a}_t \in \{0, 1\}^n$ and $\sum_{i=1}^n a_{t,i} = m$, where $m \in [n]$ is constant over time. Each arm i incurs then a random reward discounted like $\gamma^{t-1} r_{t,i}$, where $\gamma \in (0, 1]$ is a discount factor, and makes a transition to a new state $s_{t+1,i}$ in function of $s_{t,i}$ and $a_{t,i}$ but independently of the other arms.

Similarly to rested case, the restless Markovian bandits are a specific MDP – that we denote by M – whose state space is $\mathcal{X}$, and action space is $\mathcal{A}(m) := \{\boldsymbol{a} \in \{0, 1\}^n : \sum_{i=1}^n a_i = m\}$. We say that M is the *global* MDP. There are two possible criteria. In discounted restless bandit, the decision maker wants to find an optimal policy π^* that maximizes the expected cumulative discounted reward of any initial state (see e.g., [Niñ07b; Fu+19; AM20; Niñ20]). In undiscounted case $\gamma = 1$, the maximization criterion is the long-run average reward (see e.g., [Whi88; Whi96; PT94; GJN21; AB22]). As mentioned in Section 2.4, the average reward criterion is a generalization of discounted criterion (see Equation (2.10)). So, we will only discuss the average reward criterion here.

In restless bandit with average reward criterion, the average reward or gain of policy π is defined by: for any $\boldsymbol{s} \in \mathcal{X}$,

$$\lim_{T \to +\infty} \frac{1}{T} \mathbb{E}^{\pi} \left[\sum_{t=1}^{T} \sum_{i=1}^{n} r_{t,i} \mid \boldsymbol{s}_1 = \boldsymbol{s} \right]. \tag{4.2}$$

As presented in Chapter 2, if the MDP M has a finite state and action spaces, then an optimal policy π^* that maximizes (4.2) for any $\boldsymbol{s} \in \mathcal{X}$ exists and is deterministic. However, due to the curse of dimensionality, computing such an optimal policy is notoriously difficult as its complexity grows exponentially with the number of arms. In fact, it is shown in [PT94, Theorem 4] that computing an optimal policy in the restless bandit with average reward criterion is PSPACE-hard. In about 1980, Peter Whittle proposed a heuristic in the form of the largest index rule, later known as *Whittle index policy*, for restless bandit problem with average reward criterion. The computational complexity of this heuristic is linear in the number of arms making it scalable for problems with a large number of arms. We discuss this index policy more in the following section.

4.3.2 Restless bandit with average reward criterion: Whittle index policy

In his seminal paper [Whi88], Peter Whittle proposes the following heuristic: *if all arms verify a technical condition known as indexability, then each state s_i of each arm i is associated with a real number $\lambda(s_i)$, which is now known as the Whittle index of state s_i. At each time step, the decision maker activates m arms whose Whittle index of their current state are the m greatest indices.* This heuristic performs extremely well in practice, see e.g., [GM02; Ans+03; GRK06]. In fact, Whittle index policy has been shown to be asymptotically optimal as the number of arms grows to infinity under certain technical assumptions [WW90; LT00; Ver16].

Following Whittle's development [Whi88], the hard constraint "for any time $t \geq 1$, $\sum_{i=1}^{n} a_{t,i} = m$" is relaxed to

$$\lim_{T\to+\infty} \frac{1}{T}\mathbb{E}^{\pi}\left[\sum_{t=1}^{T}\sum_{i=1}^{n} a_{t,i} \mid \boldsymbol{s}_1 = \boldsymbol{s}\right] = m.$$

The Lagrangian relaxation of maximizing (4.2) under the hard constraint is then written:

$$\lim_{T\to+\infty} \frac{1}{T}\mathbb{E}^{\pi}\left[\sum_{i=1}^{n}\sum_{t=1}^{T} (r_{t,i} - \lambda a_{t,i}) \mid \boldsymbol{s}_1 = \boldsymbol{s}\right] + \lambda m, \tag{4.3}$$

where λ is a Lagrangian multiplier associated to the constraint. Finding a policy π that maximizes (4.3) can be done by working on n independent local problems: for arm i, the optimal actions for state s_i must satisfy the Bellman optimality equation

$$g^*(s_i) + h^*(s_i) = \max_{a_i\in\{0,1\}} \Big(r_i^{a_i}(s_i) - a_i\lambda + \sum_{s_i'\in\mathcal{S}_i} P_i^{a_i}(s_i, s_i') h^*(s_i') \Big) \tag{4.4}$$

where $g^*(s_i)$ and $h^*(s_i)$ are the optimal gain and bias of state s_i of arm i respectively (see Section 2.4 for their formal definitions). In economical perspective, the term λ can be viewed as a "tax for activation", adjusted to ensure that m arms are activated on average. A negative tax would be viewed as a "subsidy". We will use the term "penalty" in the sense of tax. From [Whi88; Whi96], if arm i is indexable, then the

Whittle index of state s_i is the critical value $\lambda(s_i)$ where action rest and activate are equivalent in (4.4):

$$r_i^0(s_i) + \sum_{s_i' \in \mathcal{S}_i} P_i^0(s_i, s_i')h^*(s_i') = r_i^1(s_i) - \lambda(s_i) + \sum_{s_i' \in \mathcal{S}_i} P_i^1(s_i, s_i')h^*(s_i'). \tag{4.5}$$

So, the index is meaningful: the higher the tax needed to induce one to rest an arm, the more rewarding must it be to activate that arm.

Note that Whittle index can be defined for both discounted ($\gamma < 1$) or average reward ($\gamma = 1$) criteria. However, Whittle index policy is suboptimal in general. Precisely, under certain technical assumptions, Whittle index policy is asymptotically optimal for the restless bandit with average reward criterion as the number of arms grows to infinity [WW90].

Finally, there are efforts dedicated to studying asymptotically optimal policies in the finite-horizon setting. For example, [HF17; BS20; ZF21; GGY22a] use the relaxed version of the problem that shares some features in Whittle's relaxation, but that is adapted to the finite-horizon criteria. To the best of our knowledge, there is no generic method that achieves the optimality in finite-horizon Markovian bandits, neither in the rested nor in the restless case. For instance, there exists a definition of Gittins index for finite-horizon rested problems, but they are known to be suboptimal.

4.4 Questions studied in this thesis

4.4.1 Indexability

In discounted rested bandit, Gittins index is well-defined [CM14]. This means that all arms are indexable in the rested bandits with discount. However, the definition of indexability in the restless bandits would seem unsettled: In [Whi96, Chapter 14], the restless bandit with average reward criterion is considered. An arm is said to be indexable if the set of states for which the arm is rested increases from empty set to the set of all states as λ increases [Whi96, Page 280]. Yet, the optimality criterion to rest an arm is not explicitly specified: is the arm rested simply if the gain is maximized? This question is important because from [Put14], there are multiple optimality criteria in the average reward model. In addition, what could we say in the case where there are multiple sets of states that increase from empty set to the set of all states as λ increases? Lastly, if arm i is indexable, then the Whittle index of state s_i can be computed by (4.5). As we mentioned in Section 2.4, vector $\boldsymbol{h}^* \in \mathbb{R}^{S_i}$ that satisfies Bellman optimality equation with the optimal gain $\boldsymbol{g}^*$ is not uniquely determined in general. So, in an indexable arm, some states can have multiple index values?

These essential questions will be treated in Chapter 5 of Part II.

4.4.2 Index computation

Several numerical methods to compute Gittins index are presented in [CM14] such as state elimination [Son08] and fast-pivoting [Niñ07a] algorithms. Given an arm with S states, the current best methods compute Gittins index of all states in $(2/3)S^3 + \mathcal{O}(S^2)$ arithmetic operations[1] [CM14]. In [Niñ20, Page 4], the author claims that it is unlikely that the complexity $(2/3)S^3 + \mathcal{O}(S^2)$ can be improved.

For discounted restless bandits, fast-pivoting algorithm of [Niñ20] is the current best method to compute Whittle index. For an arm with S states, the algorithm performs $(2/3)S^3 + \mathcal{O}(S^2)$ arithmetic operations if the initialization phase is excluded from the count. This is done by using the parametric simplex method and exploiting the special structure of this linear system to reduce the complexity of simplex pivoting steps. This fast-pivoting algorithm is an efficient implementation of adaptive-greedy algorithm [Niñ07b], which outputs Whittle index if and only if the arm satisfies a technical condition called partial conservation law (PCL), which is more restrictive than just being indexable. The work of [AM20] proposes another implementation of adaptive-greedy algorithm that computes Whittle index for general indexable arm and not just restricted to PCL-indexable. However, the implementation performs in $\mathcal{O}(S^3)$ and the constant in $\mathcal{O}(\cdot)$ for S^3 is not specified. To the best of our knowledge, there are very few efficient general-purpose algorithms to test indexability in both discounted and average-reward restless bandits. For instance, [Niñ10] is the only work that proposes an efficient explicit algorithm to test indexability for discounted restless bandits. There is no explicit algorithm for computing Whittle index in average-reward restless bandits.

In summary, there are a few essential questions:

- Is it possible to improve the complexity of computing Gittins index?
- Is testing indexability computationally hard?

[1]multiplications and additions of real numbers, regardless of their values

- Is there an efficient and unified algorithm to compute Whittle index in discounted and average-reward restless bandits?
- Is Whittle index harder to compute than Gittins index?

These questions will be answered in Chapter 6 of Part II.

4.4.3 Learning in rested Markovian bandit

Thanks to Gittins index policy, the discounted rested Markovian bandit is an MDP that escapes from the **curse of dimensionality** in terms of computational complexity. Precisely, although the state size of the rested Markovian bandit is exponential in the number of arms n, Gittins index policy is an optimal policy that demands a computation linearly in n. It is then interesting to understand how the curse of dimensionality manifests in RL problems in which the environment is an unknown rested Markovian bandit. By Tables 3.1 and 3.2, directly apply existing RL algorithms to such RL problems incurs a regret that is exponential in n. However, if the n arms all have S states, there are only nS real values to be estimated for the expected reward and nS^2 for the state transition probability. So, the minimax regret bound should be smaller for rested Markovian bandit. This raises the following questions.

- What is the minimax regret bound for RL algorithms in the rested Markovian bandits with discount?
- Gittins index policy has a very appealing computational complexity. Can the OFU and Bayesian methods leverage this index policy to escape from the curse of dimensionality?

These questions are discussed in Chapter 7 of Part III.

4.4.4 Learning in restless Markovian bandit

Similarly to rested bandits, in restless bandits with n arms and S states per arm, there are only $2nS$ real values to be estimated for the expected reward and $2nS^2$ for the transition probability. It is then equally interesting to analyze the performance of RL algorithms when the environment is an unknown restless bandit with no discount. However, in the infinite horizon average reward model, the structure of the MDP must be taken into account in the learning. That is, from Table 3.2, one needs to assume some structures of the bandit. Yet, the size of the state space $\mathcal{X}$ is exponential in the number of arms n. So, testing the bandit's structure is computationally hard, and it is preferable to find arm's structures that imply desirable bandit's structures. In consequence, we are interested in the following questions.

- How do the arms' structure translate into the bandit's structure?
- Is there any RL algorithms whose regret is bounded linearly in n in learning the general class of restless Markovian bandits with average reward criterion?
- Although Whittle index policy has a computational complexity linear in n, this index policy is not defined when the restless bandit is not indexable. Worse yet, it is suboptimal in general. So, how can Whittle index policy be utilized for learning purpose?

These questions will be addressed in Chapter 8 of Part III.

Part II

Indexability and Index Computation

5 Indexability and Bellman optimality

In the previous chapter, we gave the formalism of Markovian bandits and recalled two index policies: Gittins and Whittle index policies. In this chapter, we discuss the existence of the Whittle index in restless Markovian arm with average reward criterion and provide a few main contributions to characterize such an existence. This work is part of our article published in Mathematical Methods of Operations Research (MMOR) journal [GGK23].

We present our main contributions in Section 5.1. Then, the notations and λ-penalized MDP that is the basis for defining the indexability are described in Section 5.2. After that, we cover the ambiguities in the classical definition of indexability accompanied by simple examples in Section 5.3. These ambiguities inspire us to introduce a new notion of Bellman optimality for MDPs with the average reward criterion in Section 5.4. Based on this notion, we introduce our definitions of indexability and Whittle index in Section 5.5. Next, we extend the discussion about indexability in Section 5.6. Section 5.7 contains the properties of Bellman optimal policies. Finally, we conclude the chapter in Section 5.8.

5.1 Contributions

In this chapter, we discuss the definition of indexability and present two main contributions.

Our first contribution is to propose a univocal definition of indexability. This definition clarifies the ambiguities in the classical definitions: Classical definitions assume that an arm is indexable if the optimal set of active states is a non-increasing function of some penalty term λ. While this definition works for most practical cases, it is not always precise enough because the optimal set is not unique in general. In our definition, we specify the notion of increasingness that should be used. Our definition guarantees the uniqueness of Whittle index when it exists.

Lastly, we introduce a new notion of Bellman optimality for the MDPs with average reward criterion. This notion is more restrictive than the classical gain optimality, but it is rich enough to allow us to redefine the indexability in restless Markovian bandits with average reward criterion. We also study the characterization of gain optimal and Bellman optimal policies. In particular, we provide a checkable condition for the uniqueness of a given Bellman optimal policy. This condition is useful for index computation in the chapter that follows. This is a new contribution to the vast literature of MDP and Markovian bandit.

5.2 Notations and problem formulation

Since index policies require computation on each arm individually, we will focus only on a single arm. Also, this chapter and the next chapter concern the computational complexity. So, we redefine some notations in the previous chapter. These redefined notations are used uniquely for Part II.

5.2.1 Restless bandit arm

An n-state restless bandit arm is a Markov decision process (MDP) with a discrete state space $[n] := \{1, \dots, n\}$ and a binary action space $\{0, 1\}$, where 0 denotes the action "rest", and 1 denotes the action "activate". The time is discrete, and the evolution is Markovian: If the MDP is in state i, and action a is chosen, the MDP incurs an instantaneous reward r_i^a and transitions to a new state j with probability P_{ij}^a. We denote this MDP by $\langle [n], \{0, 1\}, \{\boldsymbol{r}^0, \boldsymbol{r}^1\}, \{\boldsymbol{P}^0, \boldsymbol{P}^1\}\rangle$.

5.2.2 λ-penalized MDP and policy structure

We consider a single arm $\langle [n], \{0, 1\}, \{\boldsymbol{r}^0, \boldsymbol{r}^1\}, \{\boldsymbol{P}^0, \boldsymbol{P}^1\}\rangle$. Following the Lagrangian relaxation in the previous chapter, we define a λ-penalized MDP[1] for each $\lambda \in \mathbb{R}$ as the following. The state and action spaces, and transition of this MDP are the same as the ones of the arm. The instantaneous reward for action a when the arm is in state i is $r_i^a - \lambda a$. We recall from Chapter 4 that the quantity λ is a penalty for taking action "activate".

[1]not to be confused with γ-discounted MDPs, where the discount is on rewards and not on actions.

Since there are only two actions, a policy π is a subset of the state space, $\pi \subseteq [n]$, such that the policy chooses to activate the arm in state i if $i \in \pi$. We say that π is a set of *active* states, and state i is *passive* if $i \notin \pi$. By abuse of notation, we will write $\pi_i = 1$ if $i \in \pi$, and $\pi_i = 0$ if $i \notin \pi$. The sequential decision problem is the average reward criterion in the λ-penalized MDP. For a given λ, we denote $\pi^*(\lambda)$ an optimal policy for the penalty λ.

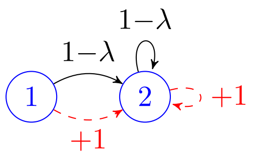
5.3 Discussion on the classical definition of indexability

The classical definition of indexability used in the literature [AM20; GJN21; Nak+21] says that an arm is indexable if and only if the optimal policy $\pi^*(\lambda)$ is non-increasing in λ (for the inclusion order). If an arm is indexable, these papers define the Whittle index of a state i as a real number λ_i such that $\pi^*(\lambda) = \{i \in [n] : \lambda_i \ > \lambda\}$. Yet, we argue that this definition has two problems:

1. What does "increasing" mean when $\pi^*(\lambda)$ is not unique? Two possibilities are: for all penalties $\lambda < \lambda'$:
   - ($\exists$) there exist policies π, π' with π optimal for λ and π' optimal for λ' such that $\pi \supseteq \pi'$;
   - ($\forall$) for all policies π, π' such that π is optimal for λ and π' is optimal for λ', we have $\pi \supseteq \pi'$.
2. What notion of "optimality" should be used? Should it be "gain optimal", "bias optimal" or another notion of optimality?

The most problematic choice is the notion of increasingness that applies to both discounted and average reward criteria: Interpretation ($\exists$) is more permissive: For instance, consider an arm with two states, and assume that the optimal policy is $\{1, 2\}$ for $\lambda < 0$, and either $\{1\}$ or $\emptyset$ for $\lambda > 0$. Interpretation ($\exists$) says that the arm is indexable while interpretation ($\forall$) says that this arm is not indexable. If the arm is indexable, what should be the index of state 1? Any choice of $\lambda_1 \in [0, +\infty]$ seems reasonable. Saying that the arm is not indexable clarifies the situation. So, we will choose the interpretation ($\forall$) in our new definition of indexability in Section 5.5.

The choice of optimality is due to the average reward criterion. To show it, let us consider the example in Figure 5.1. In this example, the gain optimal policies are $\{1, 2\}$ and $\{2\}$ for $\lambda < 0$, and $\{1\}$ and $\emptyset$ for $\lambda > 0$. According to the interpretation ($\exists$), the problem is indexable, but the index for state 1 is unclear. According to the interpretation ($\forall$), the problem is not be indexable. Yet, it is reasonable that this example is indexable, and the index of state 1 and 2 is $\lambda_1 = \lambda_2 = 0$. We argue that it is not the interpretation ($\forall$) to blame for the non-indexability of this example but the optimality criterion. That is, for $\lambda < 0$, the policy $\{2\}$ is gain optimal, but it rests the arm in state 1 and gets a reward of 1 while it is "better" to activate the arm in state 1 and get a reward of $1 - \lambda$. The policy $\{2\}$ is gain optimal because state 1 is a transient state, and state 2 is the recurrent state. So, "acting optimally" in the recurrent states "leads to" gain optimality (we will discuss this in Section 5.7.2). The same discussion goes for policy $\{1\}$ when $\lambda > 0$. This inspires us to introduce a new notion of optimality for policy $\{1, 2\}$ when $\lambda < 0$, and policy $\emptyset$ when $\lambda > 0$.



The gain optimal policies are $\{1, 2\}$ and $\{2\}$ for $\lambda < 0$, and $\{1\}$ and $\emptyset$ for $\lambda > 0$. When $\lambda < 0$, policy $\{2\}$ is gain optimal but acts suboptimal in state 1. When $\lambda > 0$, policy $\{1\}$ is gain optimal but acts suboptimal in state 1.

Figure 5.1.: Example in which the classical definition of indexability is ambiguous: All transitions are deterministic, and all labels on transitions indicate rewards. Solid black arrows correspond to the action "activate", and dashed red arrows to the action "rest".

5.4 Definition of Bellman optimality

Given a policy π, we denote by g_i^π the gain of state i when following policy π in λ-penalized MDP. Let $g_i^* = \max_\pi g_i^\pi$ be the maximal gain starting from state i. As mentioned in Section 2.4.3, the optimal gain $\boldsymbol{g}^*$ is uniquely defined. We recall from Section 2.4.3 that a policy π is *gain optimal* if $g_i^\pi = g_i^*$ for all state i. Also, the vector $\boldsymbol{g}^*$ is the optimal gain if and only if there exists a vector $\boldsymbol{h}^*$, called *optimal bias* vector that satisfies the Bellman *optimality* equations: for all $i \in [n]$,

$$\begin{aligned} g_i^* &= \max_{a \in \{0,1\}} \Big(\sum_{j=1}^n P_{ij}^a g_j^* \Big) \\ g_i^* + h_i^* &= \max_{a \in \{0,1\}} \Big(r_i^a - \lambda a_i + \sum_{j=1}^n P_{ij}^a h_j^* \Big). \end{aligned} \tag{5.1}$$

We define a new notion of optimality as the following.

Definition 5.1 (Bellman optimal policy)
A policy π is Bellman optimal *if there exists a vector $\boldsymbol{h}^* \in \mathbb{R}^n$ satisfying Bellman optimality equation* (5.1) *with the optimal gain $\boldsymbol{g}^*$ such that, for all state $i \in [n]$,*

$$\sum_{j=1}^{n} P_{ij}^{\pi_i} g_j^* = g_i^* \text{ and } \pi_i \in \operatorname*{arg\,max}_{a\in\{0,1\}} \Big(r_i^a - \lambda a_i + \sum_{j=1}^{n} P_{ij}^{a} h_j^* \Big). \qquad (5.2)$$

By [Put14, Theorem 9.1.7], Bellman optimal policy always exists in MDPs with finite state and action spaces, and Bellman optimality implies gain optimality. However, we will see in Section 5.7.2 that the converse is not true in general. So, the notion of Bellman optimality is more restrictive than the notion of gain optimality.

With this definition, in Figure 5.1, policy $\{2\}$ is not Bellman optimal for $\lambda < 0$, and policy $\{1\}$ is not Bellman optimal for $\lambda > 0$. The Bellman optimality is very natural, and we say that it is new because, to the best of our knowledge, the definition is never introduced in the literature of average reward MDPs. Note that the distinction between gain optimality and Bellman optimality disappears in infinite horizon discounted problems or in ergodic MDPs with average reward criterion. Also, Bellman optimality is equivalent to canonical optimality of finite horizon MDPs (see e.g. [GSZ00]).

In general MDPs, a Bellman optimal policy can be obtained using *Multichain Policy Iteration* algorithm. We refer to [Put14, Section 9.2.1] for more detail about this algorithm.

We stay focused on the indexability of restless Markovian arm in this chapter's progression and will discuss the Bellman optimality more at the end of the chapter.

5.5 Our unambiguous definition of indexability

To solve the two ambiguities in the classical definition of indexability, we use the following definition.

Definition 5.2 (Indexability)
Given a finite-state arm, let $\Pi^*(\lambda)$ be the set of all Bellman optimal policies for a penalty λ. We say that the arm is indexable if for all $\lambda < \lambda'$, and all policies $\pi \in \Pi^*(\lambda)$ and $\pi' \in \Pi^*(\lambda')$, $\pi \supseteq \pi'$.

This definition says that the function $\pi^*(\lambda) \supseteq \pi^*(\lambda')$ regardless of the choice of Bellman optimal policies. With this definition, the example in Figure 5.1 is indexable. Indeed, for $\lambda < 0$, $\{1, 2\}$ is the only Bellman optimal policy, and for $\lambda > 0$, $\emptyset$ is the only Bellman optimal policy.

As we show next, Definition 5.2 guarantees that Whittle index is uniquely defined when it exists.

Lemma 5.3 (Definition of the Whittle index)
In an n-state arm, the two following properties are equivalent:

(i) The arm is indexable.

(ii) For all state $i \in [n]$, there exists a unique penalty λ_i – called the Whittle index of state i – such that if $\pi \in \Pi^*(\lambda)$ is any Bellman optimal policy for the penalty λ, then $\pi_i = 1$ if $\lambda < \lambda_i$, and $\pi_i = 0$ if $\lambda > \lambda_i$.

With this lemma, the indices in Figure 5.1 are $\lambda_1 = \lambda_2 = 0$. We should stress that, by Lemma 5.3, the Whittle index λ_i can either be finite or infinite. When we say that "a policy π is optimal for the penalty $+\infty$", this means "there exists a penalty $\bar{\lambda}$ such that π is optimal for all $\lambda \geq \bar{\lambda}$". Similarly, when we say that "a policy π is optimal for the penalty $-\infty$", this means "there exists a penalty $\underline{\lambda}$ such that π is optimal for all $\lambda \leq \underline{\lambda}$".

Proof. The lemma is a direct consequence of the definition of indexability.

(i) $\Rightarrow$ (ii) – Assume first that the arm is indexable. Let $i \in [n]$ be a state, and let $\lambda_i = \sup\{\lambda : \exists \pi \in \Pi^*(\lambda) \text{ such that } \pi_i = 0\}$. By Definition 5.2, if π' is a Bellman optimal policy for a penalty $\lambda > \lambda_i$, then $\pi' \subseteq \pi$, which in turn implies that $\pi'_i = 0$. Similarly, if $\lambda < \lambda_i$, then $\pi'_i = 1$. This implies (ii).

(ii) $\Rightarrow$ (i) – Assume (ii), and let σ^k be the state with the kth smallest index (where ties are broken arbitrarily) for $1 \leq k \leq n$. Let λ_{σ^k} be the index of the state σ^k, and let $\lambda \in (\lambda_{\sigma^{k-1}}, \lambda_{\sigma^k})$. By (ii), any Bellman optimal policy for the penalty $\lambda < \lambda_{\sigma^{k-1}}$ contains $\pi^k := [n] \setminus \{\sigma^1, \dots, \sigma^{k-1}\}$. Similarly, π^k contains any Bellman optimal policy for the penalty $\lambda > \lambda_{\sigma^{k-1}}$. So, the condition in Definition 5.2 is satisfied. □

5.6 More definitions of indexability?

In our new definition of indexability, we clarify the notion of increasingness by using the interpretation $\forall$. By doing so, we already solve the ambiguity in the indexability of discounted restless bandits. Meanwhile, the last remaining ambiguity is due to the average reward setting. We solve this by using the notion of Bellman optimality because we believe that this notion is the most natural one. This allows the example of Figure 5.1 to be indexable. However, we must recognize that this new definition still does not cover all the problem of indexability in restless bandits with average reward criterion. Indeed, let us consider the two examples in Figure 5.2.

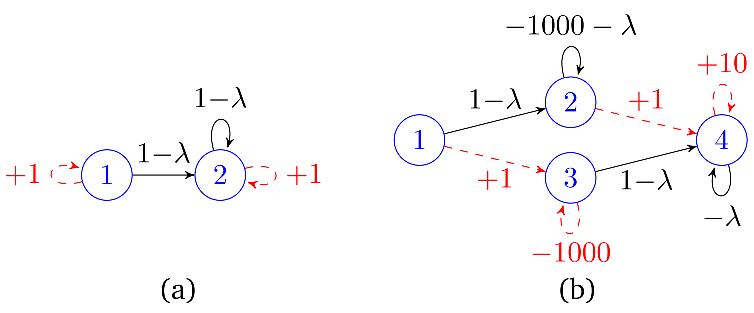


Figure 5.2.: Examples that are non-indexable when using Bellman optimality but indexable when using a stronger notion of optimality: All transitions are deterministic, and all labels on transitions indicate rewards. Solid black arrows correspond to the action "activate", and dashed red arrows to the action "rest".

For example (a), the Bellman optimal and gain optimal policies are identical: policy $\{1,2\}$ for $\lambda < 0$, and $\{1\}$ and $\emptyset$ for $\lambda > 0$. Hence, example (a) is not indexable according to Definition 5.2. However, the intuition would suggest that the index for this problem should be $\lambda_1 = \lambda_2 = 0$. This is what would have happened if we had used the notion of **bias optimality**, which is stronger than the notion of Bellman optimality. Yet, this would not solve the ambiguity of example (b), for which the index of state 1 is not clear unless one would use an even stronger notion of optimality (such as the notion of **Blackwell optimality** discussed in [Put14, Chapter 10]). In this thesis, we do not use these stronger definitions because computing a Blackwell-optimal policy in a n-state MDP is, to the best of our knowledge, not computable in $O(n^3)$ (the complexity of computing Blackwell-optimal policies is unknown to this date). Hence, in this thesis, we use the notion of Bellman optimality that we believe to be the best trade-off between expressiveness and computability. We discuss the Bellman optimality more in the following section.

5.7 Properties of Bellman optimal policies

In Section 5.4, we introduce the notion of Bellman optimality in λ-penalized MDP. However, this notion can be defined in any MDPs. In this section, we discuss the properties of Bellman optimality and the difference between this optimality and the gain optimality. To ease the exposition, we consider a fixed penalty $\lambda = 0$ so that we can drop λ from the notation. Thus, all the technical lemmas in this section apply to general MDPs.

5.7.1 Advantage function and characterization of Bellman optimality

Let $\boldsymbol{g}$ and $\boldsymbol{h}$ be a solution of Bellman evaluation equations (2.12) and (2.13) for policy π. We define the (dis-)advantage of action[2] a in state i over policy π by

$$B_i^a(\boldsymbol{g}, \boldsymbol{h}) := r_i^a + \sum_{j\in[n]} P_{ij}^a h_j - g_i - h_i. \tag{5.3}$$

If π is unichain, then $\boldsymbol{h}$ is determined up to a constant vector, and $B_i^a(\boldsymbol{g}, \boldsymbol{h})$ is then uniquely defined for any state $i \in [n]$, $a \in \{0, 1\}$, and $(\boldsymbol{g}, \boldsymbol{h})$ that satisfies (2.12) and (2.13).

For $(\boldsymbol{g}^*, \boldsymbol{h}^*)$ a solution of Bellman optimality equations (2.14) and (2.15), an optimal action a for state i satisfies both $\sum_{j\in[n]} P_{ij}^a g_j^* = g_i^*$ and $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$ [Put14; SF78].

Lemma 5.4 (Bellman optimality characterization)
Consider a policy $\pi : [n] \mapsto \{0, 1\}$. The three following properties are equivalent.

(i) *π is Bellman optimal.*

(ii) *There exists a vector $\boldsymbol{h}^* \in \mathbb{R}^n$ satisfying Bellman optimality equation* (2.15) *with $\boldsymbol{g}^*$ such that $\sum_{j\in[n]} P_{ij}^{\pi_i} g_j^* = g_i^*$ and $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$ for all states $i \in [n]$.*

(iii) *There exists a vector $\boldsymbol{h} \in \mathbb{R}^n$ satisfying Bellman evaluation equation* (2.13) *for π with $\boldsymbol{g}^*$ such that, for all state $i \in [n]$, $\sum_{j\in[n]} P_{ij}^{\pi_i} g_j^* = g_i^*$ and $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}) \le 0$ for all action $a \in \{0, 1\}$.*

Proof. (i) $\Leftrightarrow$ (ii) is a direct consequence of Definition 5.1 and (5.3).

[2]Note that it is also a function of $\boldsymbol{h}$.

(iii) $\Rightarrow$ (ii): By definition of advantage function, $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}) \leq 0$ for all action $a \in \{0,1\}$ and all state $i \in [n]$ if and only if $\boldsymbol{h}$ satisfies the optimality equation (2.15). Moreover, if $\boldsymbol{h}$ satisfies the evaluation equation (2.13) for π with $\boldsymbol{g}^*$, then $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}) = 0$ for all state $i \in [n]$.

(i) $\Rightarrow$ (iii) is a direct consequence of Definition 5.1 and (2.13). □

5.7.2 Difference between gain and Bellman optimalities

The immediate difference between both optimalities is that the Bellman optimality is more restrictive than the gain optimality. Indeed, Lemma 5.4 shows that a policy is Bellman optimal if and only if it achieves the maximum of the optimality equation (2.15) on all states for some $\boldsymbol{h}^*$. On the other hand, the following lemma shows that a policy is gain optimal if and only if it achieves the maximum of (2.15) on its recurrent states for any $\boldsymbol{h}^*$.

Lemma 5.5 (Gain optimality characterization)
Let $\pi : [n] \mapsto \{0,1\}$ be a policy, and Φ^π be the set of recurrent states under policy π. The three properties below are equivalent.

*(i) For all state i, $\sum_{j\in[n]} P_{ij}^{\pi_i} g_j^* = g_i^*$ and for **all** $\boldsymbol{h}^* \in \mathbb{R}^n$ satisfying Bellman optimality equation (2.15) with $\boldsymbol{g}^*$, $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$ for all $i \in \Phi^\pi$.*

*(ii) For all state i, $\sum_{j\in[n]} P_{ij}^{\pi_i} g_j^* = g_i^*$ and for **some** $\boldsymbol{h}^* \in \mathbb{R}^n$ satisfying Bellman optimality equation (2.15) with $\boldsymbol{g}^*$, $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$ for all $i \in \Phi^\pi$.*

(iii) π is gain optimal.

Proof. First, given a policy π, let $\boldsymbol{r}^\pi$ be the reward vector induced by π, $r_i^\pi = r_i^{\pi_i}$ for any $i \in [n]$. Also, let $\boldsymbol{P}^\pi$ be the transition matrix of the Markov chain induced by π, $P_{ij}^\pi = P_{ij}^{\pi_i}$ for any $i, j \in [n]$. Next, we define the *limiting matrix* $\bar{\boldsymbol{P}}^\pi := \underset{t\to+\infty}{\text{C-}\lim} (\boldsymbol{P}^\pi)^t$ ([Put14, Appendix A.4]). Since the MDP has a finite state space, $\bar{\boldsymbol{P}}^\pi$ always exists and well-defined for any stationary policy π. It describes the state transition in the steady regime under policy π. That is, $\bar{P}_{ij}^\pi$ is the probability that the arm transitions from state i to j in the steady regime under policy π. From [Put14, Section A.4], we have

- $\bar{\boldsymbol{P}}^\pi \boldsymbol{P}^\pi = \boldsymbol{P}^\pi \bar{\boldsymbol{P}}^\pi = \bar{\boldsymbol{P}}^\pi$
- If $j \notin \Phi^\pi$, then $\bar{P}_{ij}^\pi = 0$ for all $i \in [n]$

- If π is unichain, then the rows of $\bar{\boldsymbol{P}}^\pi$ are identical.

By [Put14, Theorem 8.2.6], the gain of policy π can be expressed by $g_i^\pi = \sum_{j\in[n]} \bar{P}_{ij}^\pi r_j^\pi$ for any $i \in [n]$. In vector notation, $\boldsymbol{g}^\pi = \bar{\boldsymbol{P}}^\pi \boldsymbol{r}^\pi$.

Now, we can prove the lemma.

(i) $\Rightarrow$ (ii) is trivial.

(ii) $\Rightarrow$ (iii): By definition of $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^*)$, we have $r_i^\pi - g_i^* = h_i^* - \sum_{j\in[n]} P_{ij}^\pi h_j^*$ for any recurrent state i under π. Multiply this with $\bar{P}_{ki}^\pi$, and sum over $i \in [n]$ (if i is not recurrent, then $\bar{P}_{ki}^\pi = 0$) gives

$$\sum_{i\in[n]} \bar{P}_{ki}^\pi \Big(r_i^\pi - g_i^*\Big) = \sum_{i\in[n]} \bar{P}_{ki}^\pi h_i^* - \underbrace{\sum_{i\in[n]} \bar{P}_{ki}^\pi \sum_{j\in[n]} P_{ij}^\pi h_j^*}_{=\sum_{j\in[n]} \bar{P}_{kj}^\pi h_j^* \text{ since } \bar{\boldsymbol{P}}^\pi \boldsymbol{P}^\pi = \bar{\boldsymbol{P}}^\pi.} = \boldsymbol{0}.$$

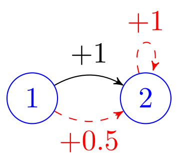
The gain of π is $\bar{\boldsymbol{P}}^\pi \boldsymbol{r}^\pi$. The above equation shows that $\bar{\boldsymbol{P}}^\pi \boldsymbol{r}^\pi = \bar{\boldsymbol{P}}^\pi \boldsymbol{g}^*$. Moreover, the assumption $\boldsymbol{P}^\pi \boldsymbol{g}^* = \boldsymbol{g}^*$ implies that $\bar{\boldsymbol{P}}^\pi \boldsymbol{g}^* = \boldsymbol{g}^*$, which in turn implies that $\bar{\boldsymbol{P}}^\pi \boldsymbol{r}^\pi = \boldsymbol{g}^*$. This shows that the gain of π is $\boldsymbol{g}^*$, and therefore π is gain optimal.

(iii) $\Rightarrow$ (i): If π is gain optimal, then $\boldsymbol{P}^\pi \boldsymbol{g}^* = \boldsymbol{g}^*$ and $\bar{\boldsymbol{P}}^\pi(\boldsymbol{r}^\pi - \boldsymbol{g}^*) = \boldsymbol{0}$. The latter rewrites as $\sum_{i\in[n]} \bar{P}_{ki}^\pi \Big(r_i^\pi - g_i^*\Big) = 0$ for all state k. For some $\boldsymbol{h}^*$ that satisfies the optimality equation (2.15) with $\boldsymbol{g}^*$, we have: for all state k

$$\sum_{i\in[n]} \bar{P}_{ki}^\pi B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = \sum_{i\in[n]} \bar{P}_{ki}^\pi \Big(r_i^\pi - g_i^* + \sum_{j\in[n]} P_{ij}^\pi h_j^* - h_i^*\Big) = 0.$$

As $\boldsymbol{h}^*$ satisfies (2.15), we have $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^*) \le 0$ for all action $a \in \{0, 1\}$ and all state $i \in [n]$. In particular, $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) \le 0$. This shows that for any state i such that $\bar{P}_{ki}^\pi > 0$, one must have $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$. Such state i are the recurrent states of π. This shows that $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^*) = 0$ for any $i \in \Phi^\pi$. □

Lemma 5.5 shows that gain optimal policies achieve the maximum of (2.15) on their recurrent states. We say that the gain optimal policies *act optimally* in their recurrent states. With this characterization, in an ergodic MDP, a policy is gain optimal if and only if it acts optimally in all states. Therefore, the distinction between gain and Bellman optimal disappears in the ergodic MDP. However, for MDPs that admit transient states, a policy can be gain optimal without acting optimally in all states, i.e., it acts optimally in its recurrent states, and possibly randomly in the transient states. To illustrate this, let us consider the example in Figure 5.3. This example is a unichain 2-state MDP, where state 1 is the transient state and allows two possible actions, and state 2 is the recurrent state and allows only one possible action. Then, there are two gain optimal policies π^1 and π^2. By computing the solution $(\boldsymbol{g}^*, \boldsymbol{h}^*)$ to the Bellman optimality equations (2.14) and (2.15), we see that, for any $\boldsymbol{h}^*$, π^1 does not achieve the maximum of the optimality equation (2.15) on state 1 while π^2 does.



There are two policies $\pi^1 := \emptyset$ and $\pi^2 := \{1\}$. Both policies are gain optimal. By solving the Bellman optimality equations (2.14) and (2.15), we have $\boldsymbol{g}^* = \mathbf{1}$ and $\boldsymbol{h}^* = c\mathbf{1}$ where c can be any real number. Consequently, $\boldsymbol{P}^{\pi^1}\boldsymbol{g}^* = \boldsymbol{g}^*$ and $\boldsymbol{P}^{\pi^2}\boldsymbol{g}^* = \boldsymbol{g}^*$, but policy π^1 does not satisfy the maximum of (2.15) on state 1 for any $\boldsymbol{h}^*$ while π^2 does. So, policy π^2 is more restrictive than π^1.

Figure 5.3.: An example where some gain optimal policies do not satisfy the Bellman optimality equation (2.15) in all states of the MDP. The black arrow shows state transition of action activate, and the red ones for action rest. The numbers along the arrows show the reward when executing the actions.

Beside showing the difference between gain and Bellman optimalities, Lemma 5.5 is very useful to prove the properties of Bellman optimal policy in the following section.

Remark. The characterization of gain optimal policies was analyzed in [Put14; SF78]. However, it was expressed and proved differently from what we have done for Lemma 5.5.

5.7.3 Conditions for the uniqueness of the Bellman optimal policy

As mentioned in Section 5.3, we use the interpretation "all Bellman optimal policies" to define the indexability. We will see in the next chapter that this direction will require the uniqueness of the Bellman optimal policy. So, we anticipate it in this section.

Conditions for Bellman optimal policies to induce the same bias vector

The previous lemma shows that a policy is gain optimal if and only if the actions in its recurrent states satisfy the optimality equation (2.15) for any $\boldsymbol{h}^*$. However, some gain optimal policies induce a bias vector that does not satisfy (2.15). That is, in Figure 5.3, the bias of policy π^1 is $h_1^{\pi^1} = c^{\pi^1} - 0.5$ and $h_2^{\pi^1} = c^{\pi^1}$, and the bias of policy π^2 is $\boldsymbol{h}_1^{\pi^2} = c^{\pi^2}\mathbf{1}$, where c^{π^1} and c^{π^2} are any real numbers (recall that $\boldsymbol{h}^{\pi^1}$

is the solution of the evaluation equation (2.13) for π^1, and $\boldsymbol{h}^{\pi^2}$ for π^2). So, $\boldsymbol{h}^{\pi^1}$ does not satisfy the optimality equation (2.15), but $\boldsymbol{h}^{\pi^2}$ does. While π^1 and π^2 are both gain optimal policies, unichain, and share one common recurrent state, namely state 2 (see Figure 5.3), we still cannot generalize the relationship between $h_i^{\pi^1}$ and $h_i^{\pi^2}$ for all state $i \in [n]$. In contrast, the following lemma clarifies this question for Bellman optimal policies that are unichain and share at least one common recurrent state.

Lemma 5.6

Suppose that two policies π and θ are Bellman optimal, unichain and have at least one common recurrent state: $\Phi^\pi \cap \Phi^\theta \neq \emptyset$.
Then for any $\boldsymbol{h}^\pi$ and $\boldsymbol{h}^\theta$ solutions of Bellman evaluation equation (2.13) for π and θ, there exists a constant c such that $h_i^\pi - h_i^\theta = c$ for all state $i \in [n]$. Moreover, $B_i^{\theta_i}(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^\theta) = 0$ for all state $i \in [n]$.

Proof. We define $\bar{\boldsymbol{P}}^\pi$ and $\bar{\boldsymbol{P}}^\theta$ for policies π and θ as we did in the proof of Lemma 5.5. Since π and θ are Bellman optimal, $\boldsymbol{h}^\pi$ and $\boldsymbol{h}^\theta$ satisfy the optimality equation (2.15) together with $\boldsymbol{g}^*$ (see Lemma 5.4). In consequence, we have

$$\boldsymbol{h}^\pi \geq \boldsymbol{r}^\theta - \boldsymbol{g}^* + \boldsymbol{P}^\theta \boldsymbol{h}^\pi.$$

By Lemma 5.5 (i), the above inequality is an equality for any state $i \in \Phi^\theta$ because policy θ is gain optimal.

As $\boldsymbol{h}^\theta$ satisfies the evaluation equation (2.13), we have

$$\boldsymbol{h}^\theta - \boldsymbol{h}^\pi \leq \boldsymbol{r}^\theta - \boldsymbol{g}^* + \boldsymbol{P}^\theta \boldsymbol{h}^\theta - (\boldsymbol{r}^\theta - \boldsymbol{g}^* + \boldsymbol{P}^\theta \boldsymbol{h}^\pi) = \boldsymbol{P}^\theta(\boldsymbol{h}^\theta - \boldsymbol{h}^\pi),$$

with equality for any state $i \in \Phi^\theta$. This shows that for all t, $\boldsymbol{h}^\theta - \boldsymbol{h}^\pi \leq (\boldsymbol{P}^\theta)^t(\boldsymbol{h}^\theta - \boldsymbol{h}^\pi)$, which implies that $\boldsymbol{h}^\theta - \boldsymbol{h}^\pi \leq \bar{\boldsymbol{P}}^\theta(\boldsymbol{h}^\theta - \boldsymbol{h}^\pi)$, with equality for any state $i \in \Phi^\theta$. Similarly, $\boldsymbol{h}^\pi - \boldsymbol{h}^\theta \leq \bar{\boldsymbol{P}}^\pi(\boldsymbol{h}^\pi - \boldsymbol{h}^\theta)$, with equality for any state $i \in \Phi^\pi$.

Let $c_i^\pi = \sum_{j\in[n]} \bar{P}_{ij}^\pi \left(h_j^\pi - h_j^\theta\right)$, and $c_i^\theta = \sum_{j\in[n]} \bar{P}_{ij}^\theta \left(h_j^\pi - h_j^\theta\right)$. By what we have just shown, for all state i, we have

$$c_i^\theta \quad \underbrace{\leq}_{\text{equality if } i \in \Phi^\theta} \quad h_i^\pi - h_i^\theta \quad \underbrace{\leq}_{\text{equality if } i \in \Phi^\pi} \quad c_i^\pi$$

As both policies are unichain, c_i^π and c_i^θ do not depend on state i. Moreover, if there exists $i \in \Phi^\theta \cap \Phi^\pi$, then $c_i^\pi = c_i^\theta = c$. In consequence, $h_i^\pi - h_i^\theta = c$ for all state i.

Furthermore, by definition of advantage function, $B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = B_i^{\pi_i}(\boldsymbol{g}^*, \boldsymbol{h}^\theta) = 0$ and $B_i^{\theta_i}(\boldsymbol{g}^*, \boldsymbol{h}^\theta) = B_i^{\theta_i}(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = 0$ for all state i. □

Uniqueness of the Bellman optimal policy

By Definition 5.1, checking if a given Bellman optimal policy is unique is not trivial due to the multiplicity of $\boldsymbol{h}^*$ that satisfies the optimality equation (2.15) with $\boldsymbol{g}^*$. So, the following lemma provides a condition that can be used to verify if a given Bellman optimal and unichain policy is the unique Bellman optimal policy.

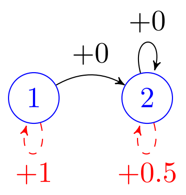
Lemma 5.7 (Condition for the uniqueness of Bellman optimal policy)
Let π be a Bellman optimal policy that is unichain. If π is not the unique Bellman optimal policy, then there exists a state i and an action $a \neq \pi_i$ such that $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = 0$.

Proof. Let $\theta \neq \pi$ be another Bellman optimal policy. Since θ is gain optimal, and $\boldsymbol{h}^\pi$ satisfies the optimality equation (2.15), Lemma 5.5 (i) implies that $B_i^{\theta_i}(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = 0$ for any $i \in \Phi^\theta$. If there exists $i \in \Phi^\theta$ such that $\theta_i \neq \pi_i$, then the proof is concluded. Otherwise, $\theta_i = \pi_i$ for any state $i \in \Phi^\theta$. In such a case, θ and π coincide for all recurrent states of θ, and $\Phi^\theta = \Phi^\pi$. Moreover, as π is unichain, θ is also unichain. Hence, Lemma 5.6 implies that $B_i^{\theta_i}(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = 0$ for all state i. Since $\theta \neq \pi$, there exists at least one state $i \in [n]$ such that $\theta_i \neq \pi_i$. □

Lemma 5.7 implies that a Bellman optimal and unichain policy π is the unique Bellman optimal policy if and only if $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^\pi) < 0$ for all action $a \neq \pi_i$ and all state i. To the best of our knowledge, this lemma is a new contribution to the literature of MDP. Also, it will play an important role in the indexability testing and index computation in the next chapter.

Note that if a weakly communicating MDP has a single Bellman optimal policy, the policy must be unichain. Indeed, suppose that, in a given weakly communicating MDP, there exists a Bellman optimal policy that is multichain with two recurrent classes Φ^1 and Φ^2. Then, we can construct two unichain Bellman optimal policies: a policy π^1 whose set of recurrent states is Φ^1 and the other policy π^2 whose set of recurrent states is Φ^2. Since the MDP is weakly communicating, the optimal gain is state-independent: the optimal gain of any state in Φ^1 is the same as the optimal gain of any state in Φ^2. So, both π^1 and π^2 are gain optimal. Finally, since the original Bellman optimal policy is multichain, it is possible to construct two optimal bias vectors $\boldsymbol{h}^{*1}$ and $\boldsymbol{h}^{*2}$ such that π^1 satisfies (2.15) with $\boldsymbol{h}^{*1}$ and π^2 with $\boldsymbol{h}^{*2}$. Therefore, both π^1 and π^2 become Bellman optimal.

In MDPs that are not weakly communicating (see Figure 2.1 and Definition 2.2), it is possible that the unique Bellman optimal policy is multichain as given by the example in Figure 5.4.



The unique Bellman optimal policy is π^* such that $\pi^*_1 = 0$ and $\pi^*_2 = 0$ (i.e., $\pi^* = \emptyset$). It is clear that π^* is a multichain policy. This MDP is not weakly communicating because state 1 is not accessible by state 2, and there is no transient state.

Figure 5.4.: An example where the MDP is not weakly communicating, and the unique Bellman optimal policy is multichain. The black arrow shows state transition of action activate, and the red ones for action rest. The numbers along the arrows show the reward when executing the actions.

5.8 Conclusion

In conclusion, we presented the ambiguities in the classical indexability definition and proposed a new definition that clarified those ambiguities. Along with this new definition of indexability, we gave the corresponding definition of Whittle index. We introduced a new notion of Bellman optimality for MDPs with average reward criterion. We derived a technical lemma to check if a given Bellman optimal policy was unique. This uniqueness property will play a crucial role in computing Whittle index.

In the next chapter, we will present a unified algorithm that can compute the Whittle index of restless Markovian bandit in both discounted and average reward criteria and the Gittins index in discounted rested bandit.

6 Testing Indexability and Computing Whittle and Gittins Index in Subcubic Time

In the previous chapter, we introduced a new definition of indexability of a restless bandit arm and defined the Whittle index accordingly. In this chapter, we develop an algorithm for testing such indexability and computing the Whittle indices of any finite-state restless bandit arm. Our algorithm works in both discounted and average reward criteria and can compute the Gittins index. This algorithm will be built on three tools: (1) a careful characterization of Whittle index that allows one to recursively compute the kth smallest index from the $(k-1)$th smallest, and to test indexability, (2) the use of the Sherman-Morrison formula to make this recursive computation efficient, and (3) a sporadic use of the fastest matrix inversion and multiplication methods to obtain a subcubic complexity. We will show that efficient use of the Sherman-Morrison formula leads to an algorithm that computes the Whittle index in $(2/3)n^3 + o(n^3)$ arithmetic operations, where n is the number of states of the arm. Furthermore, the careful use of fast matrix multiplication leads to the first subcubic algorithm to compute the Whittle or Gittins index: By using the current fastest matrix multiplication, the theoretical complexity of our algorithm is $O(n^{2.5286})$. We also develop an efficient implementation of our algorithm that compute indices of restless Markovian arms with several thousands of states in less than a few seconds. This work is part of our article published in Mathematical Methods of Operations Research (MMOR) journal [GGK23].

We present our contributions in Section 6.1 and summarize the related work in Section 6.2. In Section 6.3, we provide a technical lemma to characterize an indexable arm. Section 6.4 provides a general idea of how to compute the Whittle index of a finite-state arm. Then, in Section 6.5, we show how to use the Sherman-Morrison formula to compute the indices efficiently. We then show how to reduce the algorithm's complexity by using the fast matrix multiplication method in Section 6.6. We compare the numerical result of different variants of our algorithm in Section 6.7. Next, we show how to adapt this approach to the discounted case in Section 6.8. After that, in Section 6.9, we provide a detailed comparison between our algorithm and the algorithms of [Niñ20; AM20], which are designed for discounted restless bandits. Finally, we conclude in Section 6.10.

6.1 Contributions

In this chapter, we investigate the Whittle index computation in restless Markovian bandit problems and present three main contributions.

Our first contribution is to propose a unified algorithm that computes the Whittle indices of restless bandits for both discounted and average reward criteria. Our algorithm, which can be viewed as a refinement of the algorithm in [AM20], tests whether an input arm is indexable and computes the Whittle index if the arm is. As a byproduct, our algorithm can compute the Gittins index in rested bandits, a subclass of restless bandits. This algorithm computes the indices in increasing order and relies on the efficient use of the Sherman-Morrison formula to compute the Whittle index in $(2/3)n^3 + O(n^2)$ plus a subcubic time [Str69] to solve a linear system of order n. Moreover, the algorithm can detect on the fly if a computed index violates the indexability condition, adding an extra $(1/3)n^3 + O(n^2)$ arithmetic operations. This test is optional: the complexity of our algorithm is $n^3 + o(n^3)$ when testing the indexability and $(2/3)n^3 + o(n^3)$ without the test. These two complexities are comparable to the ones excluding the common initialization phase of reduced-pivoting indexability (RPI) and fast-pivoting adaptive greedy (FPAG) algorithms in [Niñ10]. For discounted problems, our algorithm works for any finite-state arm regardless of its structure. For average reward problems, our algorithm takes as input any arm and outputs three possibilities: the arm is (1) indexable, (2) non-indexable, or (3) multichain. We show the correctness of the algorithm, which proves that for unichain arms, our algorithm can fully characterize if the arm is indexable or not (that is, the output is either (1) indexable or (2) non-indexable). The possible outputs of our algorithm are summarized in Figure 6.1.

Our second contribution is to show how to reduce the complexity of the above algorithm for obtaining the first subcubic algorithm to compute the Whittle index. This improvement is made possible by the fact that a linear system can be solved in subcubic time. By carefully reordering the computations, we show that it is possible to reduce the use of the Sherman-Morrison formula at the price of solving more linear systems. The subcubic complexity comes by striking a good balance between having too many or too few linear systems to solve. Using the current fastest matrix multiplication method, our algorithm can test indexability and compute the Whittle

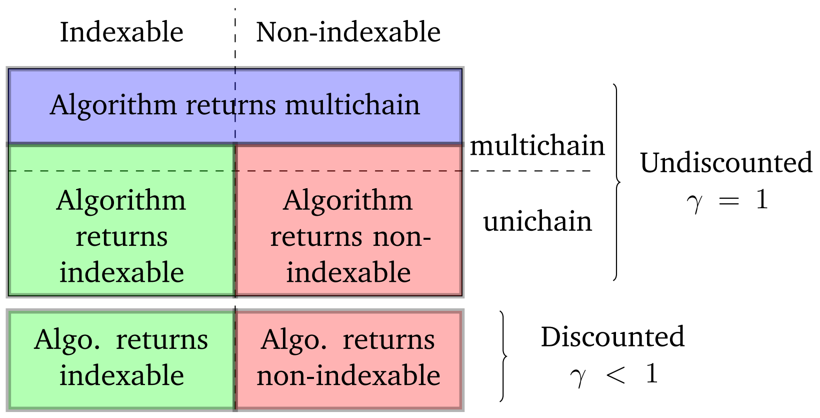

Indexable
Non-indexable
Algorithm returns multichain
multichain
unichain
Undiscounted
$\gamma = 1$
Algorithm returns indexable
Algorithm returns non-indexable
Algo. returns indexable
Algo. returns non-indexable
Discounted
$\gamma < 1$

Figure 6.1.: Possible outputs of our algorithm: For unichain or discounted problems, our algorithm tests indexability and returns the index if and only if the problem is indexable. For some multichain problems, the algorithm can test the indexability and compute the Whittle index. For the others, it only returns that the problem is multichain.

index in $O(n^{2.5286})$. Our algorithm is also the first subcubic algorithm to compute the Gittins index.

Our last contribution is to provide an open-source implementation of our algorithm in `Python` with `Numba`, and to present an empirical evaluation of the performance of our implementation. Our results show that the algorithm is very efficient in testing indexability and computing the Whittle index. Moreover, our simulations indicate that the subcubic version of our algorithm not only has an asymptotically small complexity but is also, in practice, faster than our original $(2/3)n^3$ algorithm. Testing indexability and computing the index takes less than one second for $n = 1000$ states and less than 10 minutes for $n = 15000$ states. This is 15 to 20 times faster than the original computation times reported in [Niñ20] (for a Matlab implementation), and about 5 times faster than an optimized implementation of [Niñ20] (for a Julia implementation).

6.2 Related work

The computation of Gittins index has received a lot of attention in the past, see for instance [CK86; KV87; Niñ07a; Son08] and the recent survey [CM14]. For an n-state arm, the algorithms having the smallest complexity perform $(2/3)n^3 + O(n^2)$ arithmetic operations [CM14]. Note that on page 4 of [Niñ20], the author claims that it is unlikely that this complexity can be improved. As we see later, we do improve upon this complexity.

Concerning the Whittle index, to the best of our knowledge, there are very few efficient general-purpose algorithms to test indexability. [Niñ10] is the only work that proposes an efficient algorithm to test indexability. Otherwise, most papers studying the Whittle index either assume that the studied model is indexable or focus on specific classes of restless bandits for which the structure of arms can be used to show indexability, see, i.e., [AAR11; AM19b; AM21; BP17]. Assuming indexability, the computation of Whittle index has been considered by a few papers.

For restless bandits with a discount factor $\gamma \in (0, 1)$, the most efficient algorithm for computing the Whittle index was recently presented in [Niñ20]. This algorithm, called *fast-pivoting* algorithm, performs $(2/3)n^3 + O(n^2)$ arithmetic operations[1] if the initialization phase is excluded from the count. This is done by using the parametric simplex method and exploiting the special structure of this linear system to reduce the complexity of simplex pivoting steps. This fast-pivoting algorithm is an efficient implementation of the adaptive-greedy algorithm [Niñ07b], which outputs the Whittle index if and only if the arm satisfies a technical condition called partial conservation law (PCL), which is more restrictive than just being indexable. So, it is not applicable for all indexable restless bandits. Based on a geometric interpretation of the Whittle index, the authors in [AM20] propose a refinement of the adaptive-greedy algorithm of [Niñ07b] to compute the Whittle index of all indexable restless bandits. For an n-state arm, the refined algorithm of [AM20] achieves a $O(n^3)$ complexity by using the Sherman-Morrison formula. The authors also propose a few checkable conditions to test indexability. However, those conditions are not necessary for indexability, which means that if an arm does not verify the conditions, we cannot conclude that the arm is non-indexable, and an algorithm to check indexability is still needed. Also, no detailed description is given for adapting those conditions and their algorithm to restless bandits with average reward criterion. A thorough comparison between our algorithm and [AM20; Niñ20] will be given in this chapter. While computing the Whittle indices of a known arm's model is still a challenge, there is exciting work that tries to learn Whittle index when only the arm's simulator is given, and the arm's model is unknown. For instance, [GJN21; AB22; Fu+19] use Q-learning algorithm to estimate Whittle index as time evolves in finite-state restless bandits. Moreover, the work of [Nak+21] uses a deep reinforcement learning framework to estimate the Whittle indices of arms with a large state space or convoluted transition kernel, assuming a notion of strong indexability.

For restless bandits with average reward criterion $\gamma = 1$, the author of [Niñ20] only provides a brief description of how the fast-pivoting algorithm developed for

[1]multiplications and additions of real numbers, regardless of their values

discounted criterion can be adapted, although no explicit algorithm is given in that paper. For continuous-time n-state restless bandits, the work of [AGV21] proposes an algorithm checking the indexability and computing the Whittle index with a complexity exponential in the number of states n. According to Remark 4.1 of that paper, this complexity can be reduced to $O(n^5)$ if the restless bandit is known to be indexable, and threshold-based policies are optimal. It is stated that their approach is not applicable for discounted restless bandits. For learning aspect, the work of [GJN21] shows as to learn Whittle index in average reward criterion by maintaining two Q-functions, updating them using Q-learning algorithm, and deducing the Whittle index from them when needed. The way the Whittle indices are computed is very close to our work but less efficient than our algorithm since the authors are more interested in learning the index.

6.3 Characterization of indexability in restless arm with average reward criterion

As what is done in the previous chapter, we consider the λ-penalized MDP defined based on a restless Markovian arm $\langle [n], \{0,1\}, \{\boldsymbol{r}^0, \boldsymbol{r}^1\}, \{\boldsymbol{P}^0, \boldsymbol{P}^1\}\rangle$ having n finite states.

In this chapter, we also introduce some more notations when developing our algorithm. The notations in this chapter and the previous chapter are used only for Part II of this thesis.

The following lemma proposes a characterization of any indexable arm, which we will later use to derive our algorithm.

Lemma 6.1 (Characterization of indexable arm)
In an n-state arm, the two following properties are equivalent.

(i) The arm is indexable.

(ii) There is a non-decreasing sequence of penalties $\mu^0_{\min} := -\infty \leq \mu^1_{\min} \leq \mu^2_{\min} \leq \cdots \leq \mu^n_{\min} \leq \mu^{n+1}_{\min} := +\infty$ and a sequence of policies $\pi^1 := [n] \supsetneq \pi^2 \supsetneq \cdots \supsetneq \pi^{n+1} := \emptyset$ such that:

- *If $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$, there exists a unique Bellman optimal policy π^k.*

- *If k is such that $\mu^{k-1}_{\min} < \mu^k_{\min}$, then all Bellman optimal policies for the penalty $\mu^{k-1}_{\min}$ contain π^k, and π^k contains all Bellman optimal policies for the penalty $\mu^k_{\min}$.*

Moreover, if the arm is indexable, then the index of state i is $\mu^k_{\min}$, where $k \geq 1$ is such that $i \in \pi^k \setminus \pi^{k+1}$.

In the above lemma, we use a subscript "min" in the penalties $\mu^k_{\min}$ in order to be consistent with the same quantities used in Algorithm 2 and 3. The signification of this "min" is because it will be a minimum of values of the form μ^k_i. We should stress that these quantities (and the Whittle index λ_i) can either be finite or infinite. We recall that when we say "a policy π is optimal for the penalty $+\infty$", this means "there exists a penalty $\bar{\lambda}$ such that π is optimal for all $\lambda \geq \bar{\lambda}$". Symmetrically, when we say "a policy π is optimal for the penalty $-\infty$", this means "there exists a penalty $\underline{\lambda}$ such that π is optimal for all $\lambda \leq \underline{\lambda}$". Also, the last part of the lemma implies that π^k is the unique Bellman optimal policy for all penalty $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$.

Proof. The lemma is a consequence of the indexability's definition, namely Definition 5.2.

$(i) \Rightarrow (ii)$ – Assume first that the arm is indexable. By Lemma 5.3, there exists a sequence $\{\lambda_i\}_{i\in[n]}$ such that λ_i is the Whittle index of state i, and if $\pi \in \Pi^*(\lambda)$ is any Bellman optimal policy for the penalty λ, then $\pi_i = 1$ if $\lambda < \lambda_i$, and $\pi_i = 0$ if $\lambda > \lambda_i$. Now, let σ^k be the state with the kth smallest index (where ties are broken arbitrarily). Let $\mu^k_{\min} := \lambda_{\sigma^k}$ be the index of the state σ^k, and let $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$. Then, any Bellman optimal policy for a penalty $\lambda < \lambda_{\sigma^{k-1}}$ contains $\pi^k := [n] \setminus \{\sigma^1, \dots, \sigma^{k-1}\}$. Similarly, π^k contains any Bellman optimal policy for all penalty $\lambda > \lambda_{\sigma^{k-1}}$. This implies that the policy π^k is the unique Bellman optimal policy for all $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$.

$(ii) \Rightarrow (i)$ – The property (ii) implies that π^k is the unique Bellman optimal policy for all $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$. So, the condition in Definition 5.2 is satisfied. □

6.4 Condition for indexability and basic algorithm

This section aims to provide a basic algorithm to detect whether an arm is indexable and, if so, compute the Whittle index of all states. This algorithm tries to construct a sequence of *unichain* policies $\pi^1 \supsetneq \pi^2 \supsetneq \dots$ that satisfy the conditions of Lemma 6.1.

We prove the correctness of our algorithm: if it can construct such a sequence, then the problem is indexable, and the computed indices are correct. On the contrary, if the algorithm cannot compute such a sequence of policies, this is either because the arm is not indexable or because the arm is multichain.

6.4.1 Condition for Bellman optimality

In this section, we provide two technical lemmas that we will use in our algorithm. They provide conditions to check whether a given unichain policy is Bellman optimal and, if so, whether it is the unique Bellman optimal policy.

Since λ is a variable, the gain and bias functions depend on λ in the λ-penalized MDP. We will write them as the functions of λ: For instance, given a policy π, its gain is $\boldsymbol{g}^{\pi}(\lambda)$, and its bias is $\boldsymbol{h}^{\pi}(\lambda)$. Also, the optimal gain is $\boldsymbol{g}^{*}(\lambda)$.

Let $\pi \subseteq [n]$ be a unichain policy and $(g^{\pi}, \boldsymbol{h}^{\pi})$ satisfies the Bellman evaluation equation (2.13) for π (see Proposition 2.1). Along with the notion of advantage in Section 5.4, we denote by α_i^{π} the *active advantage* in state i under policy π, which is the difference between the value in state i of action activate and the one of action rest. It is defined by:

$$\alpha_i^{\pi}(\lambda) := r_i^1 - r_i^0 - \lambda + \sum_{j=1}^{n}(P_{ij}^1 - P_{ij}^0)h_j^{\pi}(\lambda). \tag{6.1}$$

For any unichain policy π, the evaluation equation (2.13) uniquely determines the vector $\boldsymbol{h}^{\pi}$ up to an additive constant $c\mathbf{1}$ (see [Put14, Chapter 8]). Hence, the active advantage vector $\boldsymbol{\alpha}^{\pi}$ is uniquely determined for any unichain policy π. As we will see later, the function $\boldsymbol{\alpha}^{\pi}(\lambda)$ is affine in λ. Note that despite the name "advantage", $\boldsymbol{\alpha}^{\pi}(\lambda)$ can be negative.

Our algorithm computes the Whittle index in increasing order by trying to eliminate states one by one. The following lemma shows that to identify the next state to eliminate, one should look at when the active advantage of the active state equals 0. In this lemma, $\pi \ominus \{i\}$ denotes the symmetric difference between π and $\{i\}$, *i.e.*, $\pi \ominus \{i\} = \pi \setminus \{i\}$ if $i \in \pi$ and $\pi \ominus \{i\} = \pi \cup \{i\}$ if $i \notin \pi$. Also, the active advantage provides necessary and sufficient condition for a unichain policy to be Bellman optimal, and/or to be the unique Bellman optimal policy.

Lemma 6.2

In a finite-state arm, let π be a unichain policy. Then, for any penalty λ:

(i) *Suppose that π is gain optimal. Then, π is Bellman optimal if and only if $\alpha_i^\pi(\lambda) \geq 0$ for all $i \in \pi$ and $\alpha_i^\pi(\lambda) \leq 0$ for all $i \notin \pi$.*

(ii) *Suppose that π is Bellman optimal and $\alpha_i^\pi(\lambda) = 0$. Then, $\pi \ominus \{i\}$ is also Bellman optimal. In addition, if $\pi \ominus \{i\}$ is unichain, then $\boldsymbol{\alpha}^\pi(\lambda) = \boldsymbol{\alpha}^{\pi\ominus\{i\}}(\lambda)$.*

(iii) *Suppose that π is Bellman optimal. Then, π is the unique Bellman optimal policy if and only if $\alpha_i^\pi(\lambda) > 0$ for all $i \in \pi$ and $\alpha_i^\pi(\lambda) < 0$ for all $i \notin \pi$.*

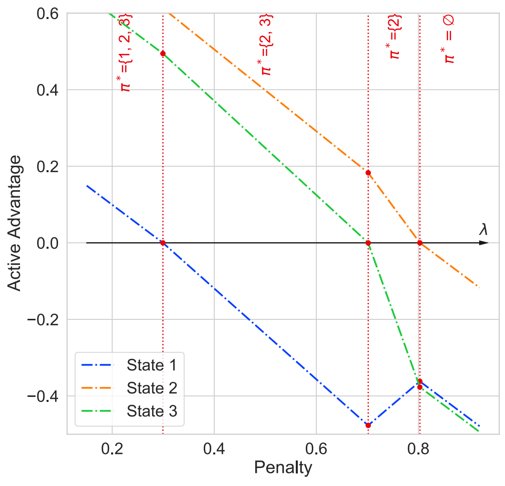
Proof. For the first point (i), one direction of the equivalence is direct: If policy π is Bellman optimal, then $\alpha_i^\pi(\lambda) \geq 0$ for all $i \in \pi$ and $\alpha_i^\pi(\lambda) \leq 0$ for all $i \notin \pi$. This is because a bias vector $\boldsymbol{h}^\pi$ that is a solution of Bellman evaluation equation (2.13) satisfies Bellman optimality equation (2.15).

We now prove the other direction of Point (i). To do so, we will use Lemma 5.4 (iii). Since π is gain optimal and unichain, $\boldsymbol{g}^* = \boldsymbol{g}^\pi = g^\pi\mathbf{1}$. This implies that $\boldsymbol{P}^\pi\boldsymbol{g}^* = \boldsymbol{g}^*$. If $\alpha_i^\pi(\lambda) \geq 0$ for all $i \in \pi$ and $\alpha_i^\pi(\lambda) \leq 0$ for all $i \notin \pi$, then the advantage function $B_i^a(\boldsymbol{g}^*, \boldsymbol{h}^\pi) = B_i^a(\boldsymbol{g}^\pi, \boldsymbol{h}^\pi) \leq 0$ for all action $a \in \{0, 1\}$ and all state $i \in [n]$. Then, Lemma 5.4 (iii) applies.

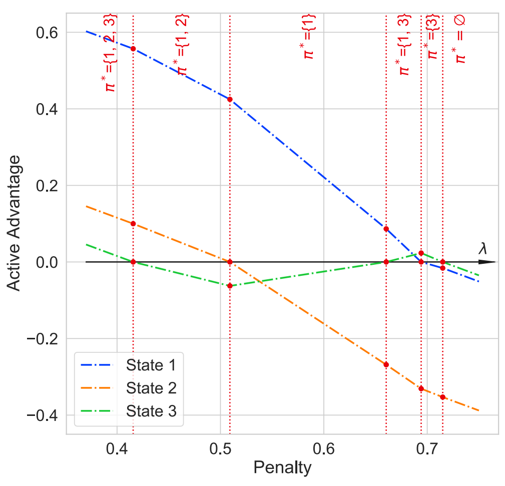
For the second point (ii), since π is unichain, the optimal gain $g_i^* = g^\pi$ for all $i \in [n]$. So, $\pi \ominus \{i\}$ satisfies the first condition of (5.2). Moreover, $\alpha_i^\pi(\lambda) = 0$ implies that policy $\pi \ominus \{i\}$ verifies evaluation equation (2.13) for π. Since π is Bellman optimal, $\boldsymbol{h}^\pi$ is one of the solutions of the optimality equation (2.15). So, $\pi \ominus \{i\}$ satisfies the second condition of (5.2). We conclude that $\pi \ominus \{i\}$ is Bellman optimal. In addition, if $\pi \ominus \{i\}$ is unichain, then $\boldsymbol{h}^\pi$ is a solution of the evaluation equations (2.13) for policy $\pi \ominus \{i\}$. Consequently, $\boldsymbol{\alpha}^\pi(\lambda) = \boldsymbol{\alpha}^{\pi\ominus\{i\}}(\lambda)$.

Points (i) and (ii) also show one direction of the equivalence of (iii): If policy π is the unique Bellman optimal policy, then for all state i, $\alpha_i^\pi(\lambda) \neq 0$. The non-trivial property is the other direction of the equivalence. This is a consequence of Lemma 5.7 that we proved in the previous chapter. □

Note that the main difficulty in proving Lemma 5.7 is that we do not assume the arm to be unichain: for a unichain arm, the bias vector of an optimal policy is unique up to a constant vector (see [SF78] and [Put14, Section 8.4]). This implies that if π is a Bellman optimal policy and $\alpha_i^\pi(\lambda) \neq 0$ for all i, then π is the unique Bellman optimal policy. The proof of Lemma 5.7 that we did in the previous chapter does not require the MDP to be unichain, but only the policy π to be Bellman optimal and unichain.



(a) Indexable arm with 3 states



(b) Non-indexable arm with 3 states

Figure 6.2.: The active advantage as a function of penalty for two unichain examples, one is indexable (Figure 6.2a), and the other is not (Figure 6.2b). The red dots mark where the lines change their slope.

To illustrate Lemma 6.1 and Lemma 6.2, we consider two three-state arms. The first arm is presented in Figure 6.2a and has the following numerical data:

$$\boldsymbol{P}^0 = \begin{bmatrix} 0.363 & 0.503 & 0.134 \\ 0.082 & 0.754 & 0.164 \\ 0.246 & 0.029 & 0.724 \end{bmatrix} \boldsymbol{P}^1 = \begin{bmatrix} 0.172 & 0.175 & 0.653 \\ 0.055 & 0.931 & 0.014 \\ 0.155 & 0.627 & 0.218 \end{bmatrix} \boldsymbol{r}^1 = \begin{bmatrix} 0.441 \\ 0.803 \\ 0.426 \end{bmatrix} \boldsymbol{r}^0 = \boldsymbol{0}$$

The second arm is presented in Figure 6.2b and has the following numerical data:

$$\boldsymbol{P}^0 = \begin{bmatrix} 0.005 & 0.793 & 0.202 \\ 0.027 & 0.558 & 0.415 \\ 0.736 & 0.249 & 0.015 \end{bmatrix} \boldsymbol{P}^1 = \begin{bmatrix} 0.718 & 0.254 & 0.028 \\ 0.347 & 0.097 & 0.556 \\ 0.015 & 0.956 & 0.029 \end{bmatrix} \boldsymbol{r}^1 = \begin{bmatrix} 0.699 \\ 0.362 \\ 0.715 \end{bmatrix} \boldsymbol{r}^0 = \boldsymbol{0} \quad (6.2)$$

For each model, we plot in Figure 6.2 the active advantage $\alpha_i^{\pi^*(\lambda)}(\lambda)$ as a function of the penalty λ for all state $i \in \{1, 2, 3\}$. By Lemma 6.2, we know that an optimal policy should activate all states having a positive advantage and rest all states having a negative advantage. Combined with the characterization of Lemma 6.1, this shows that:

- The model presented in Figure 6.2a is indexable: the optimal policy is a non-increasing function of λ, and the indices are $\lambda_1 \approx 0.3$, $\lambda_2 \approx 0.8$ and $\lambda_3 \approx 0.7$.
- The model presented in Figure 6.2b is not indexable: the optimal policy $\pi^*(0.7) = \{3\}$ is not included in $\pi^*(0.6) = \{1\}$.

6.4.2 Overview of the algorithm

Our algorithm computes the Whittle index in increasing order by navigating through unichain Bellman optimal policies and using the characterization provided by Lemma 6.1. It follows the graphical construction given in Figure 6.2a. It uses the following facts:

- If policy $\pi^1 := [n]$ is unichain, then $\boldsymbol{\alpha}^{\pi^1}$ is decreasing in λ
- Similarly, if policy $\pi^{n+1} := \emptyset$ is unichain, then $\boldsymbol{\alpha}^{\pi^{n+1}}$ is decreasing in λ
- For an indexable arm, computing the index can be done by a greedy algorithm that constructs a sequence of penalties $\mu^1_{\text{min}} \leq \mu^2_{\text{min}} \leq \cdots \leq \mu^n_{\text{min}}$ and a sequence of unichain policies $\pi^1 \supsetneq \cdots \supsetneq \pi^n$ by looking at where $\alpha_i^{\pi^k}(\lambda)$ intersects horizontal axis for all $i \in \pi^k$.
- The arm is indexable if and only if for all k such that $\mu^{k-1}_{\text{min}} < \mu^k_{\text{min}}$, the constructed π^k is the largest Bellman optimal policy for the penalty μ^k_{min}.

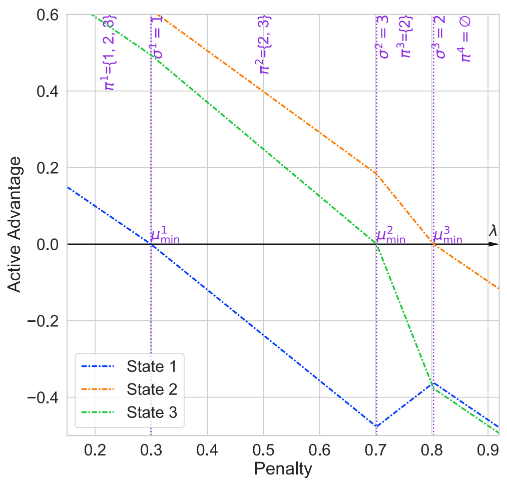
In order to compute Whittle index and test indexability, our algorithm needs all policies π^k to be unichain. It does not require the arm to be unichain. This leads to Algorithm 2, that we write in pseudocode. This algorithm relies on two subroutines: on Line 6, to compute the next index and on Line 7 to test if a policy is Bellman optimal. We will describe later in this chapter how to implement these functions in an efficient manner. Note that in all the chapter, we use the superscript k (i.e., π^k, μ^k, σ^k) to refer to the quantities computed at iteration k. We use the subscripts i or j (i.e., $\pi_i, \lambda_i, \mu_i, \pi_j$) to refer to the quantities related to states i or j.

Note that when $\mu^k_{\text{min}} = +\infty$, the quantity $\alpha_i^{\pi^k}(\mu^k_{\text{min}})$ defined in Line 6 of Algorithm 2 should be understood as $\lim_{\lambda \to \infty} \alpha_i^{\pi^k}(\lambda) \in \mathbb{R} \cup \{-\infty, +\infty\}$. These limits are well-defined because the functions αs are affine in λ.

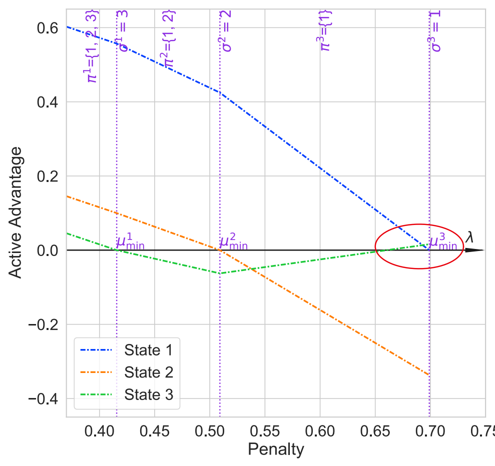
To illustrate how the algorithm works, we plot in Figure 6.3 the values computed by the algorithm for the two arms represented in Figure 6.2. For both models (indexable and non-indexable), the algorithm starts with the policy $[n]$ for which

Algorithm 2: Test indexability and compute Whittle index (if indexable).

```
Input : $n$-state arm
1  Init.
2      Set $\pi^1 := [n], \mu^0_{\min} := -\infty$
3      if $\pi^1$ is multichain then return the arm is multichain.
4  for $k = 1$ to $n$ do
5      Compute $\boldsymbol{\alpha}^{\pi^k}(\lambda)$
6      Let $\mu^k_{\min} := \inf\{\lambda \geq \mu^{k-1}_{\min} : \exists i \in \pi^k, \alpha^{\pi^k}_i(\lambda) = 0\}$
7      if $\mu^{k-1}_{\min} < \mu^k_{\min}$ and for some $i \notin \pi^k$, $\alpha^{\pi^k}_i(\mu^k_{\min}) \geq 0$ then
8          return the arm is not indexable
9      if $\mu^k_{\min} = +\infty$ then
10         Set $\lambda_i := +\infty$ for all $i \in \pi^k$
11         return the arm is indexable and the indices are $\{\lambda_i\}_{i \in [n]}$.
12     Let $\sigma^k \in \pi^k$ be such that $\alpha^{\pi^k}_{\sigma^k}(\mu^k_{\min}) = 0$ and $\lambda_{\sigma^k} = \mu^k_{\min}$
13     Set $\pi^{k+1} := \pi^k \setminus \{\sigma^k\}$
14     if $\pi^{k+1}$ is multichain then return the arm is multichain
15 return the arm is indexable and the indices are $\{\lambda_i\}_{i \in [n]}$.
```



(a) Indexable arm with 3 states. Note that the algorithm does not compute $\boldsymbol{\alpha}^{\pi^4}$. It checks if policy $\pi^4 = \emptyset$ is unichain or not. If it is, then $\alpha^{\pi^4}_i$ is decreasing in λ for each i.



(b) Non-indexable arm with 3 states. The algorithm stops at iteration 3 because $\alpha^{\pi^3}_3(\mu^3_{\min}) > 0$ (the green line at the zone circled with red ellipse).

Figure 6.3.: The active advantage $\alpha^{\pi^k}_i(\lambda)$ computed by the algorithm, for the two examples of Figure 6.2.

the derivative of the active advantage with respect to λ is -1 for all states. It then computes $\mu^1_{\min}$ which is the potential index of State $\sigma^1 = 1$ for the example 6.3a and of State $\sigma^1 = 3$ for the example 6.3b. The algorithm then moves to iteration 2 and computes $\mu^2_{\min} > \mu^1_{\min}$ for both models and observes that $\alpha^{\pi^2}_{\sigma^1}(\mu^2_{\min}) < 0$. Then the algorithm moves to iteration 3 and computes $\mu^3_{\min} > \mu^2_{\min}$. There are now two cases:

- For the example in 6.3a, the algorithm verifies that $\pi^4 := \emptyset$ is unichain ($\alpha^{\emptyset}_i$ is decreasing in λ for all i), and returns that the arm is indexable.
- For the example in 6.3b, the algorithm realizes that $\alpha^{\pi^3}_{\sigma^1}(\mu^3_{\min}) > 0$, which shows that this model is not indexable.

Note that for the indexable example of Figure 6.3a, the active advantage function is not a decreasing function of λ. Hence, this example is neither PCL-indexable (defined in [Niñ20, Definition 3]) nor strongly-indexable (defined in [Nak+21]). However, this does not prevent our algorithm from working.

6.4.3 Correctness of Algorithm 2

The following result shows that Algorithm 2 is correct.

Theorem 6.3 (Correctness of Algorithm 2)
Given an n-state arm:

(i) if Algorithm 2 outputs "the arm is indexable and the indices are $\{\lambda_i\}_{i\in[n]}$", then the arm is indexable and each λ_i is the Whittle index of state i;

(ii) if Algorithm 2 outputs "non-indexable", then the arm is non-indexable;

(iii) if Algorithm 2 outputs "multichain", then the arm is multichain.

A direct consequence of Theorem 6.3 is that for unichain arms, Algorithm 2 provides a full characterization of indexability.

Corollary 6.4 (Full characterization of indexability of unichain arm)
Given a unichain arm with finite states, Algorithm 2 outputs "the arm is indexable" if and only if it is indexable.

Note that Algorithm 2 does not require the arm to be unichain to work. The required condition is that all Bellman optimal policies $\{\pi^k\}_{k\geq 1}$ that Algorithm 2 uses are unichain. In particular, there exist examples of arms that are multichain and indexable and for which the algorithm returns "indexable". Similarly, there exist examples of arms that are multichain and non-indexable and for which the algorithm returns "non-indexable". We provide such examples in Figure 6.4.

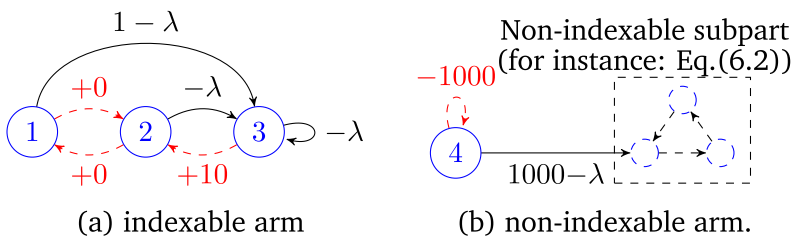


Figure 6.4.: Two examples of multichain arms for which our algorithm does not return "multichain" but returns "indexable" (a) or "non-indexable" (b).

In the first example shown in Figure 6.4(a), the arm is multichain because the policy $\{3\}$ has two recurrent classes: $\{1, 2\}$ and $\{3\}$. Yet, this arm is indexable and the indices are $\{11, 8, -10\}$. Our algorithm will output that this arm is indexable because it will explore the sequence of policies $\pi^1, \pi^2, \pi^3, \pi^4$, where

- $\pi^1 = \{1, 2, 3\}$ is the unique Bellman optimal policy for $\lambda < -10$;
- $\pi^2 = \{1, 2\}$ is the unique Bellman optimal policy for $\lambda \in (-10, 8)$;
- $\pi^3 = \{1\}$ is the unique Bellman optimal policy for $\lambda \in (8, 11)$;
- $\pi^4 = \emptyset$ is the unique Bellman optimal policy for $\lambda > 11$.

All these policies are unichain, and the policy $\{3\}$ will never be explored. Hence, our algorithm will output "indexable" for this case and will compute the indices.

In the second example, shown in Figure 6.4(b), we construct a non-indexable arm by taking the non-indexable 3-state example shown in Figure 6.3b (parameters are given in (6.2)) to which we add an extra state "4". For this state, the action activate has a very high reward and leads to the non-indexable recurrent class. The action rest has a very low reward and stays in state 4. Any policy that rests the arm in state 4 is multichain. The algorithm will start by exploring policies that activate state 4. As for the original example presented in Figure 6.3b, our algorithm will realize that the arm is non-indexable when exploring values around $\lambda \approx 0.70$. The algorithm will stop and answer "not indexable" before trying the action rest in state 4 because the active avantage for state 4 is larger than $1000 - \lambda$. The output of the algorithm is thus "non-indexable".

Our algorithm works by exploring unichain policies because (in general) the bias vector of multichain policy is not unique up to a constant vector. The characterization of Bellman optimal policies is much more difficult for multichain models, and the notion of indexability becomes more elusive (see the examples in Figure 5.2 in the previous chapter). When Algorithm 2 returns "multichain", it means that the algorithm is unable to decide whether the arm is indexable or not (but the algorithm knows that the arm is multichain because it has just found a multichain policy).

Proof of Theorem 6.3. **Proof of (i)** – We first prove by induction on k that:

If Algorithm 2 completes iteration $k \geq 1$, then π^k and π^{k+1} are unichain and

(A) π^k is the unique Bellman optimal policy for all $\lambda \in (\mu^{k-1}_{\min}, \mu^k_{\min})$;

(B) π^{k+1} is Bellman optimal for $\mu^k_{\min}$ and $\boldsymbol{\alpha}^{\pi^{k+1}}(\mu^k_{\min}) = \boldsymbol{\alpha}^{\pi^k}(\mu^k_{\min})$.

Base case $k = 1$: As we prove later in (6.16), $\frac{\partial \alpha_i^{\pi^1}}{\partial \lambda} = -1$ for all $i \in [n]$. So, for all $i \in \pi^1$, $\alpha_i^{\pi^1}(\lambda)$ is decreasing in λ. By definition, $\mu^1_{\min}$ is the smallest λ such that one of the $\alpha_i^{\pi^1}(\lambda) = 0$. Hence, for all $\lambda < \mu^1_{\min}$: $\alpha_i^{\pi^1}(\lambda) > 0$. By Lemma 6.2, this shows that π^1 is the unique Bellman optimal policy for all $\lambda \in (\mu^0_{\min}, \mu^1_{\min})$, so (A) is true. Moreover, since π^1 is Bellman optimal for the penalty $\mu^1_{\min}$ and π^2 is unichain, Lemma 6.2 implies that π^2 is Bellman optimal for the penalty $\mu^1_{\min}$ and $\boldsymbol{\alpha}^{\pi^2}(\mu^1_{\min}) = \boldsymbol{\alpha}^{\pi^1}(\mu^1_{\min})$. This shows (B).

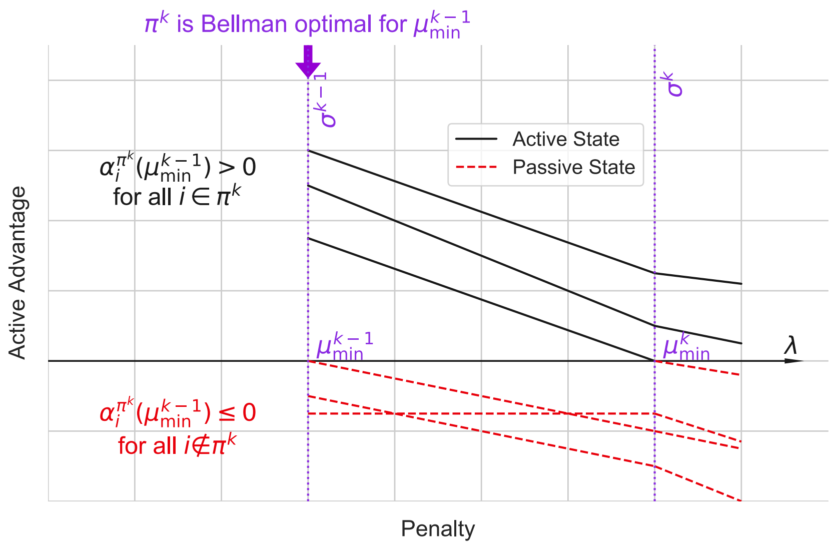


Figure 6.5.: Illustration of what happens when $\mu^{k-1}_{\min} < \mu^k_{\min}$, and when the test of Line 7 is successful (recall that the function $\alpha_i^{\pi^k}(\lambda)$ is affine in λ). The black lines are the advantage functions $\alpha_i^{\pi^k}(\lambda)$ of active state $i \in \pi^k$. The dashed red lines are the advantage functions $\alpha_i^{\pi^k}(\lambda)$ of passive state $i \notin \pi^k$.

Suppose that the induction is true until iteration $k-1$ and that the algorithm completes iteration k. If $\mu^{k-1}_{\min} = \mu^k_{\min}$, (A) is trivial and (B) is a direct consequence of the definition of $\mu^k_{\min}$ and Lemma 6.2. Consider now that $\mu^{k-1}_{\min} < \mu^k_{\min}$ and observe what happens in Figure 6.5. By the induction hypothesis, π^k is Bellman optimal for the penalty $\mu_{\min}^{k-1}$. Moreover, by the definition of $\mu_{\min}^{k}$, together with $\mu_{\min}^{k-1} < \mu_{\min}^{k}$, $\alpha_i^{\pi^k}(\mu_{\min}^{k-1}) \neq 0$ for all $i \in \pi^k$. Hence:

$$\alpha_i^{\pi^k}(\mu_{\min}^{k-1}) > 0 \text{ for } i \in \pi^k \qquad \text{and} \qquad \alpha_i^{\pi^k}(\mu_{\min}^{k-1}) \leq 0 \text{ for } i \notin \pi^k.$$

Finally, thanks to the test on Line 7 of the algorithm, we have:

$$\alpha_i^{\pi^k}(\mu_{\min}^{k}) \geq 0 \text{ for } i \in \pi^k \qquad \text{and} \qquad \alpha_i^{\pi^k}(\mu_{\min}^{k}) < 0 \text{ for } i \notin \pi^k. \tag{6.3}$$

In consequence, for each $\lambda \in (\mu_{\min}^{k-1}, \mu_{\min}^{k})$, $\alpha_i^{\pi^k}(\lambda) > 0$ for $i \in \pi^k$ and $\alpha_i^{\pi^k}(\lambda) < 0$ for $i \notin \pi^k$. Lemma 6.2 implies that π^k is the unique Bellman optimal policy for each $\lambda \in (\mu_{\min}^{k-1}, \mu_{\min}^{k})$. This shows (A). Also, (6.3) implies that π^k is Bellman optimal for the penalty $\mu_{\min}^{k}$. Combine this with the fact that π^{k+1} is unichain, Lemma 6.2 implies that π^{k+1} is Bellman optimal for $\mu_{\min}^{k}$ and $\boldsymbol{\alpha}^{\pi^{k+1}}(\mu_{\min}^{k}) = \boldsymbol{\alpha}^{\pi^k}(\mu_{\min}^{k})$. This shows (B). So, the induction is also true for iteration k.

This shows that the induction property is true for all $k \in [n]$. In particular, when $\pi^{n+1} := \emptyset$ is unichain, $\alpha_i^{\pi^{n+1}}$ is decreasing in λ as we prove later in (6.17) that $\frac{\partial \alpha_i^{\pi^{n+1}}}{\partial \lambda} = -1$ for all $i \in [n]$. Combine this with the fact that $\alpha_i^{\pi^{n+1}}(\mu_{\min}^{n}) = \alpha_i^{\pi^n}(\mu_{\min}^{n}) \leq 0$ for all i, Lemma 6.2 implies that π^{n+1} is the unique Bellman optimal policy for $\lambda > \mu_{\min}^{n}$. In summary, there are two cases for which the algorithm outputs that the arm is indexable:

1. if the algorithm goes until the end of iteration n, then the sequence of values $\{\mu_{\min}^{k}\}_{k \in [n+1]}$ and of policies $\{\pi^k\}_{k \in [n+1]}$ satisfies the conditions of Lemma 6.1 (ii) and the arm is indexable.

2. if the algorithm stops at iteration k because $\mu_{\min}^{k} = +\infty$, then one can set $\mu_{\min}^{k+1} := \ldots := \mu_{\min}^{n+1} := +\infty$ and define a sequence of policies $\pi^{k+1} \supsetneq \cdots \supsetneq \pi^{n+1} := \emptyset$ by eliminating all states of π^k in an arbitrary order. These two sequences satisfy the conditions of Lemma 6.1 (ii) and the arm is indexable.

Proof of (ii) – Our algorithm outputs non-indexable if there exists an iteration k and a state $j \notin \pi^k$, such that $\alpha_j^{\pi^k}(\mu_{\min}^{k}) \geq 0$ when $\mu_{\min}^{k-1} < \mu_{\min}^{k}$. We know that π^k is Bellman optimal for $\mu_{\min}^{k-1}$, otherwise the algorithm would have stopped before. Assume that:

$$\text{All Bellman optimal policies for any } \lambda \in (\mu_{\min}^{k-1}, \mu_{\min}^{k}] \text{ are included in } \pi^k. \tag{6.4}$$

We will see that this leads to a contradiction. We distinguish two possibilities:

1. π^k is Bellman optimal for the penalty $\mu^k_{\min}$ – This implies that $\alpha_i^{\pi^k}(\mu^k_{\min}) \leq 0$ for all $i \notin \pi^k$ which, together with $\alpha_j^{\pi^k}(\mu^k_{\min}) \geq 0$, implies that $\alpha_j^{\pi^k}(\mu^k_{\min}) = 0$. By Lemma 6.2, this would imply that $\pi^k \cup \{j\}$ is Bellman optimal for $\mu^k_{\min}$. This is in contradiction with (6.4).

2. π^k is not Bellman optimal for $\mu^k_{\min}$ – In this case, we denote $\tilde{\lambda}$ the smallest penalty $\lambda \in [\mu^{k-1}_{\min}, \mu^k_{\min}]$ such that there exists $\pi \subsetneq \pi^k$ that is Bellman optimal for $\tilde{\lambda}$ (it exists because π^k is not Bellman optimal for $\mu^k_{\min}$, and we assumed (6.4)). By definition of $\tilde{\lambda}$, π and π^k are both Bellman optimal for the penalty $\tilde{\lambda}$. Let $i \in \pi^k \setminus \pi$. By Lemma 6.2, this implies that $\alpha_i^{\pi^k}(\tilde{\lambda}) = 0$. The problem is that by definition, $\mu^k_{\min}$ is the smallest penalty λ for which there exists $i \in \pi^k$ such that $\alpha_i^{\pi^k}(\lambda) = 0$. This implies that $\tilde{\lambda} = \mu^k_{\min}$ which in turn implies that π^k is optimal for $\mu^k_{\min}$. This leads to a contradiction.

This shows that neither case 1 nor 2 are possible. So, (6.4) cannot be true. In consequence, the negation of (6.4) is true: there exists $\lambda > \mu^{k-1}_{\min}$ and $\pi \not\subseteq \pi^k$ such that π is Bellman optimal for λ. This contradicts Definition 5.2 and therefore implies that the arm is not indexable.

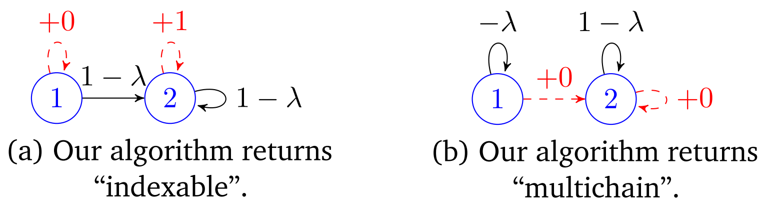
Proof of (iii) – if our algorithm outputs multichain, then the arm is multichain. This is straightforward based on the definition of multichain MDP. □

We should note that by Line 12, it is possible to have $\mu^k_{\min} = \mu^{k-1}_{\min}$. This happens when several states have the same value of Whittle index. This is not problematic because we are sure that $\sigma^k \neq \sigma^{k-1}$ by Line 13.

In the proof of Theorem 6.3 (i), we showed that when policy $[n]$ is unichain, the function $\alpha_i^{\pi^1}(\lambda)$ is decreasing in λ, which implies that it crosses the line 0 at some finite value μ_i^1. This implies that for an indexable arm, if $[n]$ is unichain then the Whittle index are all strictly larger than $-\infty$. A symmetric argument shows that if policy $\emptyset$ is unichain, then all Whittle index are strictly smaller than $+\infty$. This implies the following result.

Corollary 6.5 (Whittle index value of indexable unichain arm is finite)
Given a unichain arm with n states, if the arm is indexable, then the indices of the n states are finite: $\lambda_i \notin \{-\infty, +\infty\}$ for all $i \in [n]$.

This is not necessarily true for multichain arms. Consider the two examples of Figure 6.6.



+0
+1
$1-\lambda$
1
2
$1-\lambda$
(a) Our algorithm returns "indexable".
$-\lambda$
$1-\lambda$
+0
1
2
+0
(b) Our algorithm returns "multichain".

Figure 6.6.: Example of an indexable multichain problem with infinite Whittle index. Transitions are deterministic, and labels on edges indicate rewards (for the λ-penalized arm). Solid black arrows correspond to action "activate", and dashed red ones to the action "rest".

The examples are multichain because policy $\emptyset$ has two irreducible classes for example (a) and policy $\{1, 2\}$ has two irreducible classes for example (b). These two problems are indexable:

- For (a), the Bellman optimal policy for $\lambda < 0$ is $\{1, 2\}$ and $\{1\}$ for $\lambda > 0$. The indices are $\lambda_2 = 0$ and $\lambda_1 = +\infty$.
- For (b), the Bellman optimal policy is $\{2\}$ for $\lambda < 0$ and $\{1, 2\}$ for $\lambda > 0$. The indices are $\lambda_1 = 0$ and $\lambda_2 = -\infty$.

For the first example, our algorithm returns the correct indices because the constructed policies are $\pi^1 := \{1, 2\} \supsetneq \pi_2 := \{1\}$, and they are both unichain. For the second example, our algorithm will start with the policy $\{1, 2\}$ and will stop by saying that this example is multichain.

6.4.4 Naive implementation of Algorithm 2 (in $O(n^4)$)

For a given penalty λ, we consider a policy π that is Bellman optimal and unichain. Recall that $\boldsymbol{r}^\pi$ and $\boldsymbol{P}^\pi$ are reward vector and transition matrix induced by policy π. Also, $g^*(\lambda)$ is the maximal gain, and $\boldsymbol{h}^\pi(\lambda) \in \mathbb{R}^n$ is a solution of the evaluation equation (2.13). We consider $\boldsymbol{h}^\pi(\lambda)$ such that $h_1^\pi(\lambda) = 0$. Recall from (2.13) that for all $i \in [n]$:

$$g^*(\lambda) + h_i^\pi(\lambda) = r_i^\pi - \lambda\pi_i + \sum_{j=1}^{n} P_{ij}^\pi h_j^\pi(\lambda). \tag{6.5}$$

The above system is a system of $n+1$ linear equations with $n+1$ variables (the additional equation begins with $h_1^\pi(\lambda) = 0$). As π is unichain, the maximal gain $g^*(\lambda)$ and $\boldsymbol{h}^\pi(\lambda)$ are uniquely determined by the system of linear equations (6.5), together with the condition that $h_1^\pi(\lambda) = 0$. Note that in (6.5) the sum is for $j = 1$ to n. Since $h_1^\pi(\lambda) = 0$, it can be transformed into a sum from $j = 2$ to n.

Let us define the vector $\boldsymbol{v}^\pi(\lambda) := [g^*(\lambda)\ h_2^\pi(\lambda)\ \dots\ h_n^\pi(\lambda)]^\top$ which is similar to the vector $\boldsymbol{h}^\pi(\lambda)$ in which we replaced $h_1^\pi(\lambda)$ by $g^*(\lambda)$. We can write Equation (6.5) under a matrix form as:

$$\boldsymbol{A}^\pi \boldsymbol{v}^\pi(\lambda) = \boldsymbol{r}^\pi - \lambda \boldsymbol{\pi}, \tag{6.6}$$

where $\boldsymbol{r}^\pi := [r_1^{\pi_1}\ \dots\ r_n^{\pi_n}]^\top$, $\boldsymbol{\pi} := [\pi_1\ \dots\ \pi_n]^\top$, and $\boldsymbol{A}^\pi$ is the following square matrix:

$$\boldsymbol{A}^\pi := \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ 1 & & \ddots & \\ 1 & & & 1 \end{bmatrix} - \begin{bmatrix} 0 & P_{12}^\pi & \dots & P_{1n}^\pi \\ 0 & P_{22}^\pi & \dots & P_{2n}^\pi \\ & \vdots & & \vdots \\ 0 & P_{n2}^\pi & \dots & P_{nn}^\pi \end{bmatrix} = \begin{bmatrix} 1 & -P_{12}^\pi & \dots & -P_{1n}^\pi \\ 1 & 1-P_{22}^\pi & \dots & -P_{2n}^\pi \\ \vdots & & & \\ 1 & -P_{n2}^\pi & \dots & 1-P_{nn}^\pi \end{bmatrix}. \tag{6.7}$$

The following lemma shows that $\boldsymbol{A}^\pi$ is invertible if and only if policy π is unichain.

Lemma 6.6
Given a two-action MDP $\langle [n], \{0,1\}, \{\boldsymbol{r}^0, \boldsymbol{r}^1\}, \{\boldsymbol{P}^0, \boldsymbol{P}^1\}\rangle$, let $\boldsymbol{P}^\pi$ be the transition matrix under a Bellman optimal policy π. Policy π is unichain *if and only if the matrix $\boldsymbol{A}^\pi$ defined in* (6.7) *is invertible.*

Proof. $\boldsymbol{A}^\pi$ is not invertible if there exists a column vector $\boldsymbol{u} \neq \boldsymbol{0}$ such that $\boldsymbol{u}^\top \boldsymbol{A}^\pi = \boldsymbol{0}$. We prove that such $\boldsymbol{u}$ does not exist when policy π is unichain. Let $\boldsymbol{u} \in \mathbb{R}^n$ be an arbitrary vector such that $\boldsymbol{u}^\top \boldsymbol{A}^\pi = \boldsymbol{0}$. Then, we have

$$\begin{cases} \sum_{i=1}^n u_i & = 0 \\ u_i - \sum_{j=1}^n u_j P_{ji}^\pi & = 0, \text{ for } 2 \le i \le n \end{cases}$$

Combining the above equation with $\sum_j P_{ji}^\pi = 1$, we get:

$$u_1 = -\sum_{i=2}^n u_i = -\sum_{i=2}^n \sum_{j=1}^n u_j P_{ji}^\pi = -\sum_{j=1}^n u_j(1 - P_{j1}^\pi) = \sum_{j=1}^n u_j P_{j1}^\pi,$$

where we used that $\sum_{i=1}^n u_i = 0$ to obtain the last equality. This shows that

$$\begin{cases} \sum_{i=1}^n u_i & = 0 \\ \boldsymbol{u}^\top \boldsymbol{P}^\pi & = \boldsymbol{u}^T \end{cases}$$

The set of vector $\boldsymbol{u}$ such that $\boldsymbol{u}^\top \boldsymbol{P}^\pi$ is a vector space. It is of dimension 1 if and only if π is unichain, in which case the vector $\boldsymbol{u}$ verifying $\boldsymbol{u}^\top \boldsymbol{P}^\pi = \boldsymbol{u}^\top$ are multiples of a stationary distribution under policy π [Put14]. Thus, if the policy π induces a unichain Markov chain, then $\sum_{i=1}^n u_i = 0$ implies $\boldsymbol{u} = \boldsymbol{0}$. If policy π is not unichain, there exists $\boldsymbol{u} \neq \boldsymbol{0}$ such that $\sum_{i=1}^n u_i = 0$. $\square$

By Lemma 6.6, $\boldsymbol{v}^\pi$ is an affine function of λ:

$$\boldsymbol{v}^\pi(\lambda) = (\boldsymbol{A}^\pi)^{-1}(\boldsymbol{r}^\pi - \lambda\boldsymbol{\pi}) = (\boldsymbol{A}^\pi)^{-1}\boldsymbol{r}^\pi - \lambda(\boldsymbol{A}^\pi)^{-1}\boldsymbol{\pi} \tag{6.8}$$

For a state i, let $\delta_i := r_i^1 - r_i^0$, $\Delta_{i1} := 0$ and $\Delta_{ij} := P_{ij}^1 - P_{ij}^0$ for $j \in \{2, \dots, n\}$. By definition of active advantage function (6.1), we have:

$$\boldsymbol{\alpha}^\pi(\lambda) = \boldsymbol{\delta} - \lambda\boldsymbol{1} + \boldsymbol{\Delta}\boldsymbol{v}^\pi(\lambda).$$

For each active state i, we want to find the smallest penalty $\mu_i^k \geq \mu_{\min}^{k-1}$ such that $\alpha_i^{\pi^k}(\mu_i^k) = 0$. Suppose that π^{k-1} and π^k are unichain and π^{k-1} is Bellman optimal for $\mu_{\min}^{k-1}$. By Lemma 6.2, $\boldsymbol{\alpha}^{\pi^k}(\mu_{\min}^{k-1}) = \boldsymbol{\alpha}^{\pi^{k-1}}(\mu_{\min}^{k-1})$. Let $\boldsymbol{d}^{\pi^k} := -(\boldsymbol{A}^{\pi^k})^{-1}\boldsymbol{\pi}^k$. By (6.8), $\boldsymbol{\alpha}^{\pi^k}(\lambda)$ is a linear function of λ whose derivative is $(-\boldsymbol{1} + \boldsymbol{\Delta}\boldsymbol{d}^{\pi^k})$. In particular, $\boldsymbol{\alpha}^{\pi^k}(\lambda) = \boldsymbol{\alpha}^{\pi^{k-1}}(\mu_{\min}^{k-1}) - (\lambda - \mu_{\min}^{k-1})(\boldsymbol{1} - \boldsymbol{\Delta}\boldsymbol{d}^{\pi^k})$. Thus, $\alpha_i^{\pi^k}(\lambda) = 0$ if and only if

$$\alpha_i^{\pi^{k-1}}(\mu_{\min}^{k-1}) = (\lambda - \mu_{\min}^{k-1})(1 - \sum_{j=2}^n \Delta_{ij} d_j^{\pi^k}). \tag{6.9}$$

Recall that $\alpha_i^{\pi^{k-1}}(\mu_{\min}^{k-1})$ is non-negative for any active state $i \in \pi^k$. The value μ_i^k is the smallest $\lambda \geq \mu_{\min}^{k-1}$ that satisfies Equation (6.9). There are three cases:

1. if $\alpha_i^{\pi^{k-1}}(\mu_{\min}^{k-1}) = 0$, then $\mu_i^k := \mu_{\min}^{k-1}$;
2. if $\alpha_i^{\pi^{k-1}}(\mu_{\min}^{k-1}) > 0$ and
   a) if $1 - \sum_{j=2}^n \Delta_{ij} d_j^{\pi^k} > 0$, then

$$\mu_i^k := \mu_{\min}^{k-1} + \frac{\alpha_i^{\pi^{k-1}}(\mu_{\min}^{k-1})}{1 - \sum_{j=2}^n \Delta_{ij} d_j^{\pi^k}}; \tag{6.10}$$

   b) if $1 - \sum_{j=2}^n \Delta_{ij} d_j^{\pi^k} \leq 0$, then $\mu_i^k := +\infty$.

This shows that, for a given k, computing $\mu_{\min}^k$ of Line 6 can be done in $O(n^3)$: A first part in $O(n^3)$ to compute the inverse of matrix $\boldsymbol{A}^\pi$ and to compute $\boldsymbol{d}^\pi$, plus some smaller order terms to compute the solutions of (6.9). Similarly, the test in

Line 7 of Algorithm 2 can also be implemented in $O(n^3)$ by using $\boldsymbol{\alpha}^{\pi^k}(\mu^k_{\min}) = \boldsymbol{\alpha}^{\pi^{k-1}}(\mu^{k-1}_{\min}) - (\mu^k_{\min} - \mu^{k-1}_{\min})(\mathbf{1} - \boldsymbol{\Delta d}^{\pi^k})$ with the convention that when $\mu^k_{\min} = +\infty$, $+\infty \times 0 = 0$ and $+\infty \times x = \operatorname{sign}(x)\infty$ for any $x \neq 0$.

This leads to an overall complexity of $O(n^4)$ for Algorithm 2 that contains n loops each having a $O(n^3)$ complexity. If at some iteration k the matrix $\boldsymbol{A}^{\pi^k}$ is not invertible, then Lemma 6.6 implies that π^k is multichain. In consequence, the algorithm outputs multichain and stops. We integrate this in the newer version of our algorithm below.

6.5 The $(2/3)n^3 + o(n^3)$ algorithm

This section describes a way to implement Algorithm 2 efficiently using $O(n^3)$ operations. The main idea is to use the Sherman-Morrison formula to compute in $O(n^2)$ the active advantage vector $\boldsymbol{\alpha}^{\pi^k}(\lambda)$ associated to π^k from the one associated to π^{k-1}. This leads to a $O(n^3)$ algorithm. Once this main idea is in place, we show how to avoid unnecessary computations to obtain an algorithm that performs $(2/3)n^3 + o(n^3)$ arithmetic operations.

6.5.1 Additional notations

In order to obtain a more efficient and compact algorithm, for an iteration k and a state i, we define $y^k_i := \sum_{j=2}^n \Delta_{ij} d^{\pi^k}_j$ and $z^k_i := \alpha^{\pi^k}_i(\mu^k_{\min})$, where $d^{\pi^k}_j$, Δ_{ij} and $\alpha^{\pi^k}_i$ are as in (6.10). Equation (6.10) can be rewritten as

$$\mu^k_i = \mu^{k-1}_{\min} + \frac{z^{k-1}_i}{1 - y^k_i}. \tag{6.11}$$

The above equation can be used to compute μ^k_i and $\mu^k_{\min}$ easily from y^k_i and z^{k-1}_i. Indeed, from the previous section, we have $\alpha^{\pi^k}_i(\mu^k_{\min}) = \alpha^{\pi^{k-1}}_i(\mu^{k-1}_{\min}) - (\mu^k_{\min} - \mu^{k-1}_{\min})(1 - \sum_{j=2}^n \Delta_{ij} d^{\pi^k}_j)$, which translates into

$$z^k_i = z^{k-1}_i - (\mu^k_{\min} - \mu^{k-1}_{\min})(1 - y^k_i). \tag{6.12}$$

This shows that the critical values to compute are the variables y^k_i. In the remainder of this section, we show that the quantity y^k_i can be computed efficiently by a recursive formula.

6.5.2 Application of the Sherman-Morrison formula

To compute $y_i^{k+1} := \sum_{j=2}^{n} \Delta_{ij} d_j^{\pi^{k+1}}$, we need to compute the quantities $\boldsymbol{d}^{\pi^{k+1}} := -(\boldsymbol{A}^{\pi^{k+1}})^{-1}\boldsymbol{\pi}^{k+1}$. This requires the inverse of $\boldsymbol{A}^{\pi^{k+1}}$. By definition of π^k, two policies π^k and π^{k+1} differ by exactly one state: $\boldsymbol{\pi}^{k+1} = \boldsymbol{\pi}^k - \boldsymbol{e}_{\sigma^k}$ where $\boldsymbol{e}_j$ denotes the column vector with a 1 in jth coordinate and 0's elsewhere. Also, by definition of $\boldsymbol{A}^{\pi}$ in (6.7), the two matrices $\boldsymbol{A}^{\pi^k}$ and $\boldsymbol{A}^{\pi^{k+1}}$ differ only at the row σ^k:

$$\boldsymbol{A}^{\pi^{k+1}} = \boldsymbol{A}^{\pi^k} + \boldsymbol{e}_{\sigma^k}\boldsymbol{\Delta}_{\sigma^k} \tag{6.13}$$

where $\boldsymbol{\Delta}_{\sigma^k}$ is a row vector defined as in the previous section.

One can efficiently compute the inverse of matrix $\boldsymbol{A}^{\pi^{k+1}}$ from the one of $\boldsymbol{A}^{\pi^k}$ by using the Sherman-Morrison formula, which says that if $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is an invertible square matrix and $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^n$ are two column vectors, then the matrix $\boldsymbol{A} + \boldsymbol{p}\boldsymbol{q}^\top$ is invertible if and only if $1 + \boldsymbol{q}^\top\boldsymbol{A}^{-1}\boldsymbol{p} \neq 0$ and if $\boldsymbol{A} + \boldsymbol{p}\boldsymbol{q}^\top$ is invertible, then:

$$(\boldsymbol{A} + \boldsymbol{p}\boldsymbol{q}^\top)^{-1} = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}^{-1}\boldsymbol{p}\boldsymbol{q}^\top\boldsymbol{A}^{-1}}{1 + \boldsymbol{q}^\top\boldsymbol{A}^{-1}\boldsymbol{p}}.$$

Let $X_{ij}^k := \boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^k})^{-1}\boldsymbol{e}_j$. Following (6.13), we can apply the Sherman-Morrison formula with matrix $\boldsymbol{A}^{\pi^k}$, and vectors $\boldsymbol{p} = \boldsymbol{e}_{\sigma^k}$ and $\boldsymbol{q}^\top = \boldsymbol{\Delta}_{\sigma^k}$. After some simplification, we get:

$$\begin{aligned} X_{ij}^{k+1} &:= \boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^{k+1}})^{-1}\boldsymbol{e}_j = \boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^k} + \boldsymbol{e}_{\sigma^k}\boldsymbol{\Delta}_{\sigma^k})^{-1}\boldsymbol{e}_j \\ &= X_{ij}^k - \frac{X_{i\sigma^k}^k}{1 + X_{\sigma^k\sigma^k}^k} X_{\sigma^k j}^k \end{aligned} \tag{6.14}$$

$$\text{In particular, } X_{i\sigma^k}^{k+1} = \frac{X_{i\sigma^k}^k}{1 + X_{\sigma^k\sigma^k}^k}.$$

Before computing X_{ij}^{k+1}, we need to verify that π^{k+1} is unichain. With the help of Lemma 6.6 and the Sherman-Morrison formula, this can be done easily: π^{k+1} is unichain if and only if $1 + X_{\sigma^k\sigma^k}^k \neq 0$.

For $y_i^{k+1} := \boldsymbol{\Delta}_i \boldsymbol{d}^{\pi^{k+1}} = -\boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^{k+1}})^{-1}\boldsymbol{\pi}^{k+1}$, we use $\boldsymbol{\pi}^{k+1} = \boldsymbol{\pi}^k - \boldsymbol{e}_{\sigma^k}$ and apply the Sherman-Morrison formula to get:

$$\begin{aligned}
y_i^{k+1} &= -\boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^k} + \boldsymbol{e}_{\sigma^k}\boldsymbol{\Delta}_{\sigma^k})^{-1}(\boldsymbol{\pi}^k - \boldsymbol{e}_{\sigma^k}) \\
&= -\boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^k})^{-1}(\boldsymbol{\pi}^k - \boldsymbol{e}_{\sigma^k}) + \frac{\boldsymbol{\Delta}_i(\boldsymbol{A}^{\pi^k})^{-1}\boldsymbol{e}_{\sigma^k}\boldsymbol{\Delta}_{\sigma^k}(\boldsymbol{A}^{\pi^k})^{-1}}{1+\boldsymbol{\Delta}_{\sigma^k}(\boldsymbol{A}^{\pi^k})^{-1}\boldsymbol{e}_{\sigma^k}}(\boldsymbol{\pi}_k - \boldsymbol{e}_{\sigma^k}) \\
&= y_i^k + X_{i\sigma^k}^k + \frac{X_{i\sigma^k}^k(-y_{\sigma^k}^k - X_{\sigma^k\sigma^k}^k)}{1+X_{\sigma^k\sigma^k}^k} \\
&= y_i^k + \frac{X_{i\sigma^k}^k(1-y_{\sigma^k}^k)}{1+X_{\sigma^k\sigma^k}^k} = y_i^k + (1-y_{\sigma^k}^k)X_{i\sigma^k}^{k+1} \qquad (6.15)
\end{aligned}$$

The above formula indicate how to compute $\boldsymbol{y}^{k+1}$ from $\boldsymbol{y}^k$. To complete this analysis, let us show that $\boldsymbol{y}^1 = \boldsymbol{0}$. For a given policy π, the vector $\boldsymbol{d}^\pi$ satisfies the same equation as Equation (6.6) but replacing $r_i^{\pi_i} - \lambda\pi_i$ by $-\pi_i$. This implies that for $\boldsymbol{\pi} = \boldsymbol{\pi}^1 = [1\ \dots\ 1]^\top$, one has $\boldsymbol{d}^\pi = [-1\ 0\ \dots\ 0]^\top$ as d_1^π is the long-term average reward of a Markov reward process whose reward is negative one in all states and $d_2^\pi, \dots, d_n^\pi$ is the bias of this process. This shows that for all i, one has $y_i^1 := \sum_{j=2}^n \Delta_{ij} d_j^{\pi^1} = 0$. Moreover, by (6.8), one has

$$\begin{aligned}
\boldsymbol{\alpha}^{\pi^1}(\lambda) &= \boldsymbol{\delta} - \lambda\boldsymbol{1} + \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^1})^{-1}\boldsymbol{r}^{\pi^1} - \lambda\underbrace{\boldsymbol{\Delta}(\boldsymbol{A}^{\pi^1})^{-1}\boldsymbol{\pi}^1}_{=:\boldsymbol{y}^1} \\
&= \boldsymbol{\delta} - \lambda\boldsymbol{1} + \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^1})^{-1}\boldsymbol{r}^{\pi^1}. \qquad (6.16)
\end{aligned}$$

Finally, for $\boldsymbol{\pi}^{n+1} = [0\ \dots\ 0]^\top$, one has

$$\boldsymbol{\alpha}^{\pi^{n+1}}(\lambda) = \boldsymbol{\delta} - \lambda\boldsymbol{1} + \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^{n+1}})^{-1}\boldsymbol{r}^{\pi^{n+1}}. \qquad (6.17)$$

6.5.3 Detailed algorithm

Equation (6.11) shows how to compute μ_i^k from the values of y_i^k and z_i^{k-1} while (6.15), (6.14) and (6.12) show how to compute the values of $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{X}$ recursively in k. In order to compute $\mu_{\min}^k$ and σ^k, one only needs to compute the values μ_i^k for $i \in \pi^k$. Once $\mu_{\min}^k = \min_{i\in\pi^k}\mu_i^k$ is determined, if $\mu_{\min}^{k-1} < \mu_{\min}^k$, then Line 7 of Algorithm 2 can be performed, based on (6.12), by checking if $z_i^k \geq 0$ for some $i \in [n] \setminus \pi^k$.

This leads to Algorithm 3 that can be decomposed as follows:

Algorithm 3: Test indexability and compute Whittle indices (if indexable).

```
Input  : $n$-state arm $\langle [n], \{0,1\}, \{\boldsymbol{r}^0, \boldsymbol{r}^1\}, \{\boldsymbol{P}^0, \boldsymbol{P}^1\}\rangle$
1  Init.
2      Set $\pi^1 := [n]$, $k_0 = 1$, $\Delta_{i1} := 0, \Delta_{ij} := P^1_{ij} - P^0_{ij}, \forall i \in [n], j \in \{2, \dots, n\}$,
       $\boldsymbol{y}^1 = \boldsymbol{0}$ and $\boldsymbol{A}^{\pi^1}$ is defined by (6.7).
3      if $\pi^1$ is multichain then return the arm is multichain.
4      Set $\boldsymbol{X}^1 := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^1})^{-1}$
5  Set $\boldsymbol{\mu}^1 := \boldsymbol{r}^1 - \boldsymbol{r}^0 + \boldsymbol{X}^1\boldsymbol{r}^1$
6  Let $\sigma^1 := \arg\min_{i\in\pi^1} \mu^1_i$, $\lambda_{\sigma^1} = \mu^1_{\min} := \mu^1_{\sigma^1}$, and $\pi^2 := \pi^1 \setminus \{\sigma^1\}$
7  Set $\boldsymbol{z}^1 = \boldsymbol{\mu}^1 - \mu^1_{\min}\boldsymbol{1}$
8  for $k = 2$ to $n$ do
9      Update_X($k-1$) // Here we call Subroutine 4 or Subroutine 5
10     Set $\boldsymbol{y}^k = \boldsymbol{y}^{k-1} + (1 - y^{k-1}_{\sigma^{k-1}})\boldsymbol{X}^k_{:\sigma^{k-1}}$
11     for $i \in \pi^k$ do
12         Set $\mu^k_i := \begin{cases} \mu^{k-1}_{\min}, & \text{if } z^{k-1}_i = 0 \\ \mu^{k-1}_{\min} + \dfrac{z^{k-1}_i}{1-y^k_i}, & \text{if } z^{k-1}_i > 0 \text{ and } 1-y^k_i > 0 \\ +\infty, & \text{otherwise} \end{cases}$
13     Let $\sigma^k := \arg\min_{i\in\pi^k} \mu^k_i$ and $\mu^k_{\min} := \mu^k_{\sigma^k}$
14     Set $\boldsymbol{z}^k = \boldsymbol{z}^{k-1} - (\mu^k_{\min} - \mu^{k-1}_{\min})(\boldsymbol{1} - \boldsymbol{y}^k)$
15     if $\mu^{k-1}_{\min} < \mu^k_{\min}$ and $z^k_i \geq 0$ for some $i \in [n] \setminus \pi^k$ then
16         return the arm is not indexable
17     if $\mu^k_{\min} = +\infty$ then
18         Set $\lambda_i := +\infty$ for all $i \in \pi^k$
19         return the arm is indexable and the indices are $\{\lambda_i\}_{i\in[n]}$.
20     Set $\lambda_{\sigma^k} = \mu^k_{\min}$ and $\pi^{k+1} := \pi^k \setminus \{\sigma^k\}$
21 if $\pi^{n+1}$ is multichain then return the arm is multichain
22 return the arm is indexable and the indices are $\{\lambda_i\}_{i\in[n]}$.
```

Subroutine 4: Update_X(k)

```
for $\ell = 1$ to $k-1$ do
    for $i \in [n]$ do // or $i \in \pi^{\ell+1}$ if we do not test indexability.
        $X^{\ell+1}_{i\sigma^k} = X^{\ell}_{i\sigma^k} - X^{\ell+1}_{i\sigma^\ell} X^{\ell}_{\sigma^\ell\sigma^k}$
if $1 + X^k_{\sigma^k\sigma^k} = 0$ then
    return the arm is multichain
for $i \in [n]$ do // or $i \in \pi^{k+1}$ if we do not test indexability.
    $X^{k+1}_{i\sigma^k} = \dfrac{X^k_{i\sigma^k}}{1 + X^k_{\sigma^k\sigma^k}}$
```

1. In Lines 2 to 7, we initialize the various variables. The main complexity of this part is to compute the matrix $\boldsymbol{X}^1$, which is equivalent to solving the linear system $\boldsymbol{X}\boldsymbol{A}^{\pi^1} = \boldsymbol{\Delta}$. It can be done by inverting the matrix $\boldsymbol{A}^{\pi^1}$ and multiplying this by the matrix $\boldsymbol{\Delta}$. This can be done in a subcubic complexity by using for instance Strassen's algorithm [Str69].

2. We then enter the main loop:

   - We update the vectors $\boldsymbol{\mu}$, $\boldsymbol{z}$ by using Equations (6.11) and (6.12), and test indexability. This costs $O(n)$ operations per iteration, thus $O(n^2)$ in total.

   - We update the vector $\boldsymbol{X}$ according to (6.14). The "naive" way to do so is to use Subroutine 4. At iteration k this costs $2kn$ arithmetic operations if we test indexability, and $2\sum_{l=1}^{k}(n-l)$ if we do not test indexability. The total complexity of computing $\boldsymbol{X}$ is $n^3 + O(n^2)$ arithmetic operations if we test indexability and $(2/3)n^3 + O(n^2)$ if we do not. A detailed study of the arithmetic complexity is provided in Appendix 6.A, where we also provide details on how to efficiently implement the algorithm, including how to optimize the cost of memory access.

   - In Line 10, we update the vector $\boldsymbol{y}$ by using Equation (6.15), which costs $O(n)$ per iteration.

3. Testing if π^{n+1} is unichain can be done in $O(n^2)$ by Tarjan's strongly connected component algorithm.

Hence, the total complexity of this algorithm is $n^3 + o(n^3)$ if we test indexability and $(2/3)n^3 + o(n^3)$ if we do not test indexability. Without testing the indexability, our algorithm has the same main complexity as [Niñ20]. However, the algorithm of [Niñ20] computes Whittle index only for an arm that is PCL-indexable. Hence, we can claim that our algorithm is the first algorithm that computes Whittle index with $(2/3)n^3$ main complexity for all indexable undiscounted restless arms. It is also the first algorithm with cubic complexity that detects non-indexability of restless bandits. Note that the ties are broken arbitrarily for Lines 6 and 13.

Remark: Equivalently, instead of calling Subroutine 4 at Line 9, one could do the following update for all $j \in \pi^{k+1}$ and $i \in [n]$ (or $i \in \pi^{k+1}$ if we do not test indexability):

$$X_{ij}^{k+1} = X_{ij}^{k} - \frac{X_{i\sigma^k}^{k}}{1 + X_{\sigma^k\sigma^k}^{k}} X_{\sigma^k j}^{k}. \tag{6.18}$$

This iterative update is very close to the one used in [AM20; Niñ20] for discounted restless bandit. This results in an algorithm that has the same total complexity as Subroutine 4 (of $n^3 + O(n^2)$ or $(2/3)n^3 + O(n^2)$ with or without the indexability test) because both algorithms will have computed the same values of X_{ij}^k. The reason to use Subroutine 4 is that, as we will see in the next section, not all values of X_{ij}^k are needed at iteration k: in particular, for $\ell < k$, the computation of the value $X_{i\sigma^{k-1}}^{\ell}$ has no interest per se and is only useful because it allows to recursively compute $X_{i\sigma^{k-1}}^{k}$. In the section below, we show how to reduce the cost by avoiding the computation of $X_{i\sigma^{k-1}}^{\ell}$ when ℓ is much smaller than k. We comment more on the differences with [AM20; Niñ20] in Section 6.9 and in particular we explain why our approach can be tuned into a subcubic algorithm while (6.18) cannot.

6.6 The subcubic algorithm

6.6.1 Main idea: recomputing $\boldsymbol{X}^k$ from $\boldsymbol{A}^{\pi^k}$ periodically

The main computational burden of Algorithm 3 is concentrated on two lines: on Line 4 where we compute $\boldsymbol{X}^1$ by solving a linear system, and on Line 9 where we compute the column vector $\boldsymbol{X}_{:\sigma^{k-1}}^{k}$ from $\boldsymbol{X}_{:\sigma^{k-1}}^{1}$. The remainder of the code runs in $O(n^2)$ operations and is therefore negligible for large matrices. In fact, the computation of $\boldsymbol{X}^1$ is done by solving a linear system, which can be computed by using a subcubic algorithm (like Strassen [Str69]). In this section, we show how to optimize our algorithm by reducing the complexity of the update_X() function, at the price of recomputing the full matrix $\boldsymbol{X}^{k_0}$ from $\boldsymbol{A}^{\pi^{k_0}}$ periodically.

At iteration k of Algorithm 3, the quantities $X^k_{i\sigma^{k-1}}$ are used at Line 10 to obtain the values y^k_i. The matrix $\boldsymbol{X}^k$ is defined as $\boldsymbol{X}^k := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^k})^{-1}$. It also satisfies Equation (6.14), that is, for an iteration ℓ and states i and j, we have:

$$X^{\ell+1}_{ij} = X^{\ell}_{ij} - X^{\ell+1}_{i\sigma^\ell} X^{\ell}_{\sigma^\ell j}. \tag{6.19}$$

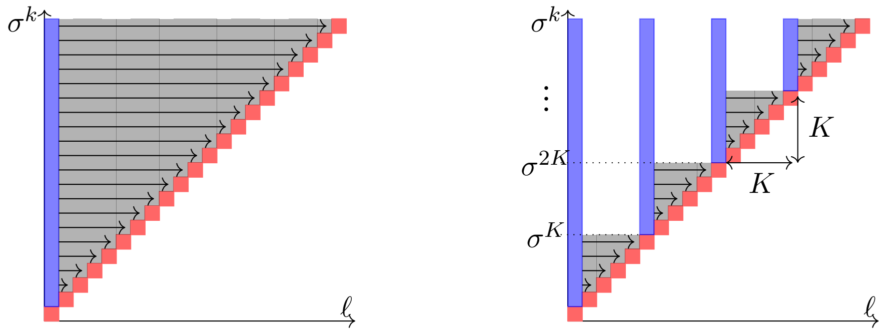
The way Subroutine 4 is implemented is to initialize $\boldsymbol{X}^1 := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^1})^{-1}$ and then use (6.19) recursively to compute the column vector $\boldsymbol{X}^k_{:\sigma^{k-1}}$ from $\boldsymbol{X}^1_{:\sigma^{k-1}}$ at each iteration.

Here, we propose an alternative formulation which consists in recomputing the whole matrix $\boldsymbol{X}^k := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^k})^{-1}$ every K iterations. In the meantime, we use (6.19) to compute the values $X^\ell_{i\sigma^{k-1}}$ for $\ell \in \{k_0+1, \dots, k\}$ where k_0 is the iteration at which we recomputed the whole matrix $\boldsymbol{X}^{k_0} := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^{k_0}})^{-1}$. This can be implemented by replacing the call to Subroutine 4 at Line 9 with a call to Subroutine 5.

Subroutine 5: Update_X_FMM(k)

```
1  if $k$ is an iteration at which we recompute the whole $\boldsymbol{X}^{k+1}$ then
2      Set $k_0 := k+1$;
3      if $\pi^{k_0}$ is multichain then
4          return the arm is multichain
5      Set $\boldsymbol{X}^{k_0} := \boldsymbol{\Delta}(\boldsymbol{A}^{\pi^{k_0}})^{-1}$
6  else
7      for $\ell = k_0$ to $k-1$ do
8          for $i \in [n]$ do // or $i \in \pi^{\ell+1}$ if we do not test indexability.
9              $X^{\ell+1}_{i\sigma^k} = X^{\ell}_{i\sigma^k} - X^{\ell+1}_{i\sigma^\ell} X^{\ell}_{\sigma^\ell \sigma^k}$
10     if $1 + X^k_{\sigma^k\sigma^k} = 0$ then
11         return the arm is multichain
12     for $i \in [n]$ do // or $i \in \pi^{k+1}$ if we do not test indexability.
13         $X^{k+1}_{i\sigma^k} = \dfrac{X^k_{i\sigma^k}}{1 + X^k_{\sigma^k\sigma^k}}$
```

To see why this can be more efficient, we illustrate in Figure 6.7 the pairs (ℓ, σ^{k-1}) for which we compute the vector $\boldsymbol{X}^\ell_{:\sigma^{k-1}}$, either for Subroutine 4 or Subroutine 5. In each case, a vertical blue line indicates that we recompute the whole matrix $\boldsymbol{X}^k$ by solving a linear system. The gray zone corresponds to the values (ℓ, σ^{k-1}) for which we compute $\boldsymbol{X}^\ell_{:\sigma^{k-1}}$ using Equation (6.19), and the red squares represent the vector $\boldsymbol{X}^k_{:\sigma^{k-1}}$ used at Line 10 of Algorithm 3. For Subroutine 4, we do one matrix inversion at the beginning and then compute for all (ℓ, σ^{k-1}) with $\ell \leq k$ because the red square at value (k, σ^{k-1}) is computed starting from the vertical blue line at value $(1, \sigma^{k-1})$. For Subroutine 5, we do $\lfloor n/K \rfloor$ (here $\lfloor n/K \rfloor = 4$) full recalculation of $\boldsymbol{X}^k$, which correspond to the vertical blue lines. We gain in terms of operations because the surface of the gray zone to compute is divided by $\lfloor n/K \rfloor$.



(a) Computation load of Subroutine 4. (b) Computation load of Subroutine 5

Figure 6.7.: Illustration of the improvement proposed by replacing Subroutine 4 with Subroutine 5 when we do not test the indexability. For Subroutine 5, we solve more linear systems (each vertical blue line corresponds to solving a linear system) but we reduce the gray zone to compute. A linear arrow corresponds to the internal loop of Line 7 of Subroutine 5.

Note that the y-axis of Figure 6.7 is ordered by increasing value of σ^k (and not by increasing value of k). The value of σ^k is computed at iteration k but unknown before iteration k. This explains why in Subroutine 5, when we recompute the matrix (X^k_{ij}) at an iteration k (vertical blue lines in Figure 6.7(b)), we recompute it for all i, j and not just $i, j \in \{\sigma^k, \dots, \sigma^K\}$ (which are the only values that we will use): Indeed, $\sigma^k, \dots, \sigma^K$ are unknown at iteration k.

6.6.2 A subcubic algorithm for Whittle index

We now assume to have access to a subcubic matrix multiplication algorithm that satisfies the following property:

(FMM) There exists an algorithm to multiply a matrix of size $n \times n$ by a matrix of size[2] $n \times n^\varrho$ that runs in $O(n^{\omega(\varrho)})$, where $\omega : [0,1] \to [2,3]$ is a non-decreasing function.

Going back to Algorithm 3 where Line 9 is Subroutine 5, we now assume that we recompute the whole matrix $\boldsymbol{X}^k$ every $O(n^\varrho)$ iterations. The new algorithm has a subcubic complexity:

[2]We write n^ϱ which is possibly non-integer. For the sake of simplicity, we write n^ϱ and $n^{1-\varrho}$ but they should be understood as $\lfloor n^\varrho \rfloor$ and $\lfloor n^{1-\varrho} \rfloor$ respectively.

Theorem 6.7 (Subcubic algorithm)
Given an n-state arm, Algorithm 3 with Subroutine 5 checks indexability and computes Whittle (and Gittins) index in time at least $\Omega(n^{2.5})$ and at most $O(n^{2.5286})$ when choosing $\varrho = 0.5286$.

We believe that Theorem 6.7 is the first theoretical result that shows that Whittle index can be computed in subcubic time. As we show in Section 6.8, this algorithm can be directly extended to discounted index. As a byproduct, we also obtain the first subcubic algorithm to compute Gittins index.

Proof. The algorithm starts by computing $\boldsymbol{X}^1$ which can be done in $O(n^{\omega(1)})$. Then, there are $n^{1-\varrho}$ times that we do:

1. We fill the "gray" mini matrices by using (6.19). This amounts to three `for` loops of size n (`for` i), n^ϱ (`for` k) and n^ϱ (`for` ℓ). Hence, each small gray matrix costs $O(n^{1+2\varrho})$.
2. At the end of a cycle, we recompute the full inverse by updating $(\boldsymbol{A}^{\pi^k})^{-1}$ from $(\boldsymbol{A}^{\pi^{k-n^\varrho}})^{-1}$. As we show in Lemma 6.8 (stated below), this can be done in $O(n^{\omega(\varrho)})$.

This implies that the algorithm has a complexity:

$$O(n^{\omega(\varrho)}) + n^{1-\varrho}\left(O(n^{1+2\varrho}) + O(n^{\omega(\varrho)})\right) = O(n^{\max\{2+\varrho, 1-\varrho+\omega(\varrho)\}}).$$

To compute the optimal value of ϱ minimizing this expression requires the knowledge of the function $\omega(\varrho)$ which is not known. The current state of the art only gives a lower bound ($\omega(\varrho) \geq 2$) and an upper bound described in [GU18].

It is shown in [GU18, Table 3] that $\varrho = 0.5286$ is the smallest currently known value of ϱ for which $\omega(\varrho) < 1 + 2\varrho$. This implies that the complexity is at most $O(n^{2.5286})$.

As for the lower bound, $\omega(\varrho) \geq 2$ implies that the complexity of the algorithm is at least $\Omega(n^{2.5})$. $\square$

In the next lemma, $\boldsymbol{B}$ plays the role of $\boldsymbol{A}^{\pi^k}$ and $\boldsymbol{A}$ the role of $\boldsymbol{A}^{\pi^{k-n^\varrho}}$. Note that as required in the lemma, exactly n^ϱ rows and columns are changed between the two matrices.

Lemma 6.8 (Fast matrix inversion)
Assume (FMM). Let $\boldsymbol{A}$ be a square matrix whose inverse $\boldsymbol{A}^{-1}$ has already been computed, and let $\boldsymbol{B}$ be an invertible square matrix such that $\boldsymbol{A} - \boldsymbol{B}$ is of rank smaller than n^{ϱ}. Then, it is possible to compute the inverse of $\boldsymbol{B}$ in $O(n^{\omega(\varrho)})$.

Proof. The matrix $\boldsymbol{B}$ can be written as $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{UCV}$ where $\boldsymbol{U}$ is an $n \times n^{\varrho}$ matrix, $\boldsymbol{C}$ is $n^{\varrho} \times n^{\varrho}$ and $\boldsymbol{V}$ is $n^{\varrho} \times n$. The Sherman–Morrison–Woodbury formula [Woo50] states that

$$\boldsymbol{B}^{-1} = (\boldsymbol{A} + \boldsymbol{UCV})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}\left(\boldsymbol{C}^{-1} + \boldsymbol{VA}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{VA}^{-1}.$$

This shows that $\boldsymbol{B}^{-1}$ can be computed by:

- Computing $\boldsymbol{D} := \boldsymbol{A}^{-1}\boldsymbol{U}$ and $\boldsymbol{E} := \boldsymbol{VA}^{-1}$: this takes $O(n^{\omega(\varrho)})$.
- Computing $\boldsymbol{F} := (\boldsymbol{C}^{-1} + \boldsymbol{VA}^{-1}\boldsymbol{U})^{-1}$: as this is the inversion of an $n^{\varrho} \times n^{\varrho}$ matrix, it can be done in $O(n^{\varrho\omega(1)})$ where $\varrho\omega(1) \leq \omega(\varrho)$.
- Computing $\boldsymbol{G} := \boldsymbol{DF}$ and then $\boldsymbol{GE}$: this again takes $O(n^{\omega(\varrho)})$.

Hence, computing $\boldsymbol{B}^{-1}$ can be done in $O(n^{\omega(\varrho)})$ operations for the inversion and all multiplications plus an additional $O(n^2)$ term for the subtraction and the addition. As $\omega(\varrho) \geq 2$, this concludes the proof of the lemma. $\square$

6.6.3 The subcubic algorithm in practice

The complexity of $O(n^{2.5286})$ given in Theorem 6.7 is mainly of theoretical interest. The value $\varrho = 0.5286$ is obtained by using the best upper bound on $\omega(\varrho)$ known today which is based on the Coppersmith-Winograd algorithm and its variants. The Coppersmith-Winograd algorithm (or its variants) are, however, known as a *galactic algorithm*: the hidden constant in the $O()$ is so large that their runtime is prohibitive for any reasonable value of n. Hence, the existence of these algorithms is of theoretical interest but has limited applicability.

This does not discard the practical improvement provided by Subroutine 5 which is based on the mere fact that multiplying two matrices (or inverting a matrix) is faster than three nested loops even for matrices of moderate size. To verify this, we launched a detailed profiling of the code of Algorithm 3 with the non-optimized Subroutine 4. It shows that for a problem of dimensions 5000, the update of Line 9 takes more than 90% of the computation time, the initialization of $\boldsymbol{X}^1$ on Line 4 takes about 5% of the time and the rest of the code takes less than 1% of the running time.

Now, if inverting the full matrix takes about 5% of the execution time, and updating the gray zone takes 95%, then by doing 5 updates, one can hope to obtain an algorithm whose running time is roughly $5 \times 5 + 95/5 \approx 43\%$ of the running time of the original implementation. As we observe in Section 6.7, this is close to the gain that we obtain in practice. A general way to choose the best number of updates is used in the numerical section. It is based on the following reasoning. For large matrices (say $n \geq 10^3$), the fastest implementations of matrix multiplication and inversion are based on Strassen's algorithm [Hua+16; Hua18]. As we report in Appendix 6.A.1, the time to solve a linear system of size n by using the default installation of `scipy` seems to run in $O(n^{2.8})$. By replacing the function $\omega(\varrho)$ used in Theorem 6.7 by a more practical bound ($\omega(\varrho) = 2.8$), the best value for ϱ becomes $\varrho = 0.9$. This indicates that our algorithm can be implemented in $O(n^{2.9})$ by doing $O(n^{0.1})$ recalculation of $\boldsymbol{X}^k$ from $\boldsymbol{A}^{\pi^k}$. Note that even for very large values of n (like $n = 15000$), $n^{0.1}$ remains quite small, *e.g.*, $15000^{0.1} \approx 2.6$. In practice, we observe that updating $\mathrm{int}(2n^{0.1})$ times (the notation $\mathrm{int}(x)$ indicates that it is rounded to the closest integer) gives the best performance among all algorithms, as reported in the next section.

6.7 Numerical experiments

In complement to our theoretical analysis, we developed a python package that implements Algorithm 3 and gives the choice of using the variant of Subroutine 4 or of Subroutine 5 to do the "update_X()" function. This package relies on three python libraries: `scipy` and `numpy` for matrix operations, and `Numba` to compile the python code. To facilitate its usage, this package can be installed by using `pip install markovianbandit-pkg`.

All experiments were conducted on a laptop (Macbook Pro 2020) with an Intel Core i9 CPU at 2.3 GHz with 16GB of Memory using Python 3.6.9 :: Anaconda custom (64-bit) under macOS Big Sur version 11.6.2. The version of the packages are `scipy` version 1.5.4, `numpy` version 1.19.5 and `Numba` version 0.53.1. The code of all experiments is available at `https://gitlab.inria.fr/markovianbandit/efficient-whittle-index-computation`.

In all of our experiments, the way we generated arms guarantees that they are almost surely unichain because all the elements on their diagonal as well as on the upper and lower diagonals are positive.

6.7.1 Time to compute Whittle indices

To test the implementation of our algorithm, we randomly generate restless bandit arms with n states where $n \in \{100, 1000, \dots, 15000\}$. In each case, both transition matrices are uniform probabilistic matrices: for each row of each matrix, we generate n i.i.d. entries following the exponential distribution and divide the row by its sum. This means that all matrices are dense. We use dense matrix since it is the worst case for computational interest. By running our algorithms on sparse matrix, we would expect to have faster running time. Note that all tested matrices are indexable. This is coherent with [GGY20; Niñ07b] that report that for uniform matrices, the probability of finding a non-indexable example decreases very rapidly with the dimension n. Finally, reward vectors were generated from random Uniform[0,1) entries.

Table 6.1.: Running time (in seconds) of the variants of Algorithm 3

	$O(n^3)$ algorithm (Subroutine 4)		Subcubic algorithm (Subroutine 5)	
n	(a) With index. test	(b) Without test	(c) With index. test	(d) Without test
100	0.006	0.004	0.007	0.005
1000	0.2	0.2	0.2	0.2
2000	1.9	1.2	1.1	1.0
3000	7.2	5.0	3.2	2.6
4000	17	12	8	6
5000	34	25	16	12
6000	60	43	27	20
7000	95	68	42	33
8000	142	99	63	49
9000	199	141	92	70
10000	275	191	122	95
11000	361	257	164	133
12000	471	339	225	190
13000	620	428	286	243
14000	790	553	408	317
15000	965	685	501	403

We record the runtime of the different variants of our algorithm and report the results in Table 6.1. Note that these results present the whole execution time of the algorithm, including the initialization phase in which $\boldsymbol{X}^1$ is computed. For each value of n, we run Algorithm 3 with four variants:

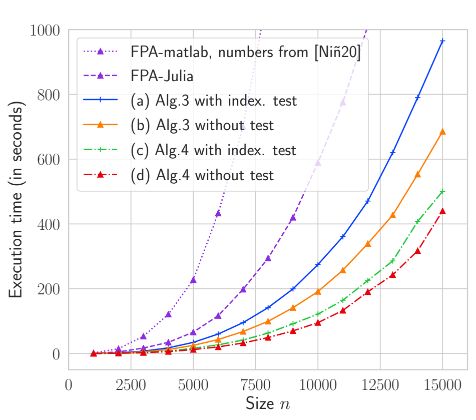
- The first two columns correspond to the $O(n^3)$ algorithm (that uses Subroutine 4 for Line 9), either (a) with the indexability test, or (b) without the indexability test.

- The last two columns correspond to the subcubic algorithm (that uses Subroutine 5 for Line 9 with int($2n^{0.1}$) updates), either (c) with the indexability test, or (d) without the indexability test.

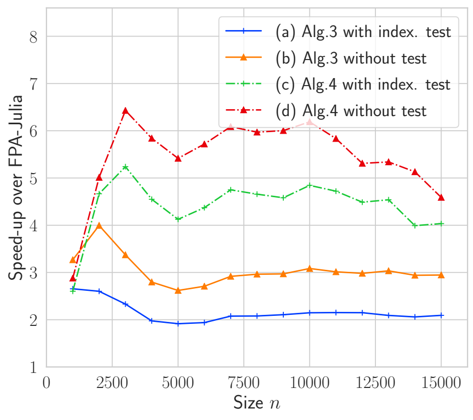
Our numbers show that our algorithm can compute the Whittle index in less than one second for $n = 1000$ states and slightly less than 7 minutes for $n = 15000$ states with variant (d). As expected, not doing the indexability test does improve the performance compared to doing the indexability test (here by a factor approximately $1/3$ for Subroutine 4 and $1/5$ for Subroutine 5). More importantly, this table shows that the time when using the subcubic variant, Subroutine 5, diminishes the computation time by about 40% to 50% compared to when using Subroutine 4. Note that for $n = 100$, using the Subroutine 4 is slightly faster than using the Subroutine 5 (while both takes only a few milliseconds). This indicates that the subcubic algorithm becomes interesting when n is large enough (say $n \geq 2000$).

To give a visual idea of how the various variants of the algorithm compare, we plot in Figure 6.8a the runtime of the four variants along with two variants of the algorithms of [Niñ20]: FPA-Matlab (the original matlab implementation), and FPA-Julia (a Julia implementation provided by the authors). We choose to compare to this algorithm as it was the one with the smallest complexity up to now. The numbers for FPA-Matlab are extracted in [Niñ20]. They are comparable to the numbers that we obtained on our machine when using the same algorithm. FPA-Julia is significantly faster. Hence, we plot in Figure 6.8b the runtime of each variant divided by the runtime of FPA-Julia. For large n, our best implementation is about 4 to 6 times faster than the best one of [Niñ20]. For instance, for $n = 15000$, our implementation takes about 7 minutes to compute the index (or 9 minutes when checking indexability) whereas FPA-Julia takes 33 minutes (and does not check indexability on the fly). In our implementation, not testing indexability reduces the computation time of $15-20\%$ for Subroutine 5 or $25-30\%$ for Subroutine 4 compared to the version that tests indexability.

It should be clear that the comparison between our implementation and FPA-Matlab or FPA-Julia has its limits. First, we do not use the same programming language since our code uses `Python` with `Numba`. To obtain a fairer comparison, we tried to rewrite the algorithm of [Niñ20] in `Python` with `Numba`, but our implementation was significantly slower than the one of FPA-Julia. Second, while the two algorithms are similar, our algorithm has some technical advantage (a simpler internal loop, the use of Subroutine 5), and our implementation is optimized for better data locality (see Appendix 6.A.2).



(a) Runtime of our implementations as a function of the state size n.



(b) Speedup of each implementation over FPA-Julia [Niñ20] as a function of the state size n.

Figure 6.8.: Numerical result over 7 simulations: in each simulation, we run the algorithm over randomly generated RBs with the state size ranging over $\{1000, \dots, 15000\}$. We plot the average runtime over 7 simulations. The solid lines represent the result of Subroutine 4 and the dashed-dot lines represent the one of Subroutine 5. The marker "+" indicates that algorithms test indexability and the triangles indicates that algorithms do not test indexability.

6.7.2 Statistics of indexable problems

To the best of our knowledge, our algorithm provides the first indexability test that scales well with the dimension n. We used this to answer a very natural question: given a randomly generated arm, how likely is it to be indexable? This question was partially answered in [Niñ07b] that shows that when generating dense arms, the probability of generating a non-indexable arm is close to 10^{-n} for $n \in \{3, \dots, 7\}$. This suggests that most arms are indexable. Below, we answer two questions: what happens for larger values of n, and more importantly, what happens when the state transition matrices are not dense?

To answer these questions, we consider randomly generated arms where the matrices $\boldsymbol{P}^0$ and $\boldsymbol{P}^1$ are b-diagonal matrices with b non-null diagonals. In particular, $b = 3$ corresponds to tridiagonal matrices, $b = 5$ corresponds to pentadiagonal matrices and $b = 7$ corresponds to septadiagonal matrices. We also compare with the classical case of dense matrices (which corresponds to $b = 2n - 1$). For each model, the entries are generated from the exponential distribution for each row, and we divide the row by the sum of generated entries for this row. We vary n from 3 to 50 and for each case, we generate 100000 arms. We report in Table 6.2 the number of indexable arms for each case. Note that pentadiagonal matrices are dense matrices for $n = 3$ and septadiagonal matrices are dense matrices for $n = 4$ and do not make sense for $n = 3$, which is why no numbers are reported.

Table 6.2.: Number of indexable problems among 100 000 randomly generated problems.

Problem size n	Tridiagonal	5-diagonal	7-diagonal	Dense
3	98 731	–	–	99 883
4	95 067	99 655	–	99 931
5	89 198	99 309	99 902	99 969
10	54 129	90 377	98 914	100 000
30	7 094	29 699	66 143	100 000
50	1 823	9 332	32 069	100 000

Based on these results, we can assert that dense models are essentially always indexable which conforms with the data reported in [Niñ07b]. The situation is, however, radically different for sparse models: the number of indexable problems decreases quickly with the number of states. For instance, there are only 1 823 indexable 50-state problems among 100 000 generated tridiagonal models (*i.e.* around 1.8% are indexable). Note that a tridiagonal model is a birth-death Markov chain which is frequently used for queuing systems. Hence, it is very important to check the indexability of the problem because it is not a prevalent property for sparse models. This also calls for new efficient policies in restless multi-arm bandit problems that are not based on Whittle indices.

6.8 Extension to the discounted case

Whittle index is also defined for the discounted case [Niñ20; AM20]. Notably, the discounted Whittle index simplifies into Gittins index when the bandit is rested (*i.e.*, when $\boldsymbol{P}^0 = \boldsymbol{I}$ and $\boldsymbol{r}^0 = \boldsymbol{0}$). In this section, we show how to adapt our algorithm to the discounted case. As a by product, we obtain the first subcubic algorithm to compute Gittins index rested bandit.

6.8.1 Discounted Whittle index

We now consider a λ-penalized MDP in which the instantaneous reward received at time $t \geq 1$ is discounted by a factor γ^{t-1}, where $\gamma \in (0,1)$ is called the discount factor: when executing action a in state i at time $t \geq 1$, the decision maker earns a reward $\gamma^{t-1}(r_i^a - \lambda a)$. For a given policy π, we denote by $u_i^\pi(\lambda)$ the expected sum of discounted rewards earned by the decision maker when the MDP starts in state i at time 1. The vector $\boldsymbol{u}^\pi(\lambda) = [u_1^\pi(\lambda) \ \dots \ u_n^\pi(\lambda)]^\top$ is called the value function of the policy π. From Section 2.3, it satisfies Bellman evaluation equation: for all state i we have:

$$u_i^\pi(\lambda) = r_i^\pi - \lambda\pi_i + \gamma\sum_{j=1}^{n} P_{ij}^\pi u_j^\pi(\lambda). \tag{6.20}$$

The above equation is a linear equation, whose solution is unique because $\gamma < 1$. It is given by:

$$\boldsymbol{u}^\pi(\lambda) = (\boldsymbol{I} - \gamma\boldsymbol{P}^\pi)^{-1}(\boldsymbol{r}^\pi - \lambda\boldsymbol{\pi}). \tag{6.21}$$

For a given penalty λ and a state i, we denote by $u_i^*(\lambda) := \max_\pi u_i^\pi(\lambda)$ the optimal value of state i. A policy π is optimal for the penalty λ, i.e., $\pi \in \Pi^*(\lambda)$, if for all state $i \in [n]$, $u_i^*(\lambda) = u_i^\pi(\lambda)$. From Section 2.3, such a policy exists, $|\Pi^*(\lambda)| > 0$. As mentioned in Section 5.4, the distinction between Bellman optimal and gain optimal disappears in discounted MDP in which we are concerned with maximizing the value function. Similarly to the average reward criterion studied before, a γ-discounted RB is called indexable if for all penalty $\lambda < \lambda'$, all $\pi \in \Pi^*(\lambda)$ and $\pi' \in \Pi^*(\lambda')$, one has $\pi \supseteq \pi'$.

6.8.2 Analogy between the average reward and the discounted versions

Let π be a policy and $i \in \pi$ be an active state. Similarly to average reward model studied before, the advantage of action activate over action rest in state i right before following policy π is given by, $\alpha_i^\pi(\lambda) := r_i^1 - r_i^0 - \lambda + \gamma\sum_{j=1}^{n}(P_{ij}^1 - P_{ij}^1)u_j^\pi(\lambda)$. Then $\alpha_i^\pi(\lambda) = 0$ if and only if $\lambda = \delta_i + \sum_{j=1}^{n}\tilde{\Delta}_{ij}u_j^\pi(\lambda)$ where δ_i is defined as for the average reward model and $\tilde{\boldsymbol{\Delta}}$ is such that for all[3] states $i,j \in [n]$: $\tilde{\Delta}_{ij} := \gamma(P_{ij}^1 - P_{ij}^0)$.

[3]Note that the definition of $\tilde{\boldsymbol{\Delta}}$ is identical to $\boldsymbol{\Delta}$ except when $j = 1$, for which $\Delta_{i1} := 0$ but $\tilde{\Delta}_{i1} := \gamma(P_{i1}^1 - P_{i1}^0)$.

To finish the derivation of the algorithm, one should note that the value function $\boldsymbol{u}(\lambda)$ plays the same role as the vector $\boldsymbol{v}(\lambda)$ defined for the average reward model. In particular, the definition of $\boldsymbol{u}$ in Equation (6.21) is the analogue of the definition of $\boldsymbol{v}$ in (6.8) up to the replacement of the matrix $\boldsymbol{A}^{\pi}$ in (6.8) by the matrix $\boldsymbol{I} - \gamma\boldsymbol{P}^{\pi}$. This means that similarly to $\boldsymbol{v}$, the value function $\boldsymbol{u}(\lambda)$ is affine in λ.

Hence, following the same development in Section 6.5, we can modify Algorithm 3 to compute the discounted Whittle index by modifying only the initialization phase:

$$\tilde{\boldsymbol{\Delta}} := \gamma(\boldsymbol{P}^1 - \boldsymbol{P}^0) \text{ and } \boldsymbol{X}^1 := \tilde{\boldsymbol{\Delta}}(\boldsymbol{I} - \gamma\boldsymbol{P}^{\pi^1})^{-1}.$$

Note that we still have $\boldsymbol{y}^1 = \boldsymbol{0}$ because $(\boldsymbol{I} - \gamma\boldsymbol{P}^{\pi^1})^{-1}\boldsymbol{\pi}^1 = \frac{1}{1-\gamma}\boldsymbol{1}$ (value function in a γ-discounted Markov reward process with reward equals to 1 in all states) and $\tilde{\boldsymbol{\Delta}}\boldsymbol{1} = \boldsymbol{0}$. Also, if Subroutine 5 is used, Line 5 should be changed to $\boldsymbol{X}^{k_0} := \tilde{\boldsymbol{\Delta}}(\boldsymbol{I} - \gamma\boldsymbol{P}^{\pi^{k_0}})^{-1}$. Last but not least, in the discounted case, we no longer needed the optimal policies to be unichain because the matrix $(\boldsymbol{I} - \gamma\boldsymbol{P}^{\pi})$ is invertible for any policy π as long as $\gamma < 1$ (from Perron-Frobenius' Theorem).

6.8.3 Gittins index

The notion of "restless" bandit comes from the fact that even when the action "rest" is taken, the Markov chain can still change state and generate rewards. When this is not the case (*i.e.*, when $\boldsymbol{P}^0 = \boldsymbol{I}$ and $\boldsymbol{r}^0 = \boldsymbol{0}$), an arm is no longer restless and is simply called a Markovian bandit (or a *rested* Markovian bandit if one wants to emphasize that it is not restless).

In a discounted rested bandit, the notion of Whittle index coincides with the notion of Gittins index (in fact, Whittle index was first introduced as a generalization of Gittins index to restless bandit in [Whi88]). In such a case, there is no notion of indexability: a discounted rested bandit is always indexable. Its index can be computed by Algorithm 3 without testing indexability. The best known algorithms to compute Gittins index runs in $(2/3)n^3 + O(n^2)$ [CM14]. When using fast matrix multiplication, our algorithm computes Gittins index in $O(n^{2.5286})$ which makes it the first algorithm to compute Gittins index in subcubic time.

Note that when $\boldsymbol{P}^1 = \boldsymbol{I}$, it is possible to compute $\boldsymbol{X}^1 = \tilde{\boldsymbol{\Delta}}/(1-\gamma)$ without having to solve a linear system which means that, when $\boldsymbol{P}^1 = \boldsymbol{I}$, our algorithm with the variant Subroutine 4 has complexity $(2/3)n^3 + O(n^2)$. This shows that our cubic algorithm can compute the Gittins index also in $(2/3)n^3 + O(n^2)$ instead of $(2/3)n^3 + o(n^3)$.

6.9 Detailed comparison with [AM20] and [Niñ20]

In this section, we compare our algorithm with two main related works for finite-state restless bandit problem.

6.9.1 Comparison with [AM20]

The paper presents an algorithm that computes Whittle indices in $O(n^3)$ (no explicit constant before n^3 is given) for all indexable problems. Despite following a different approach, our algorithm for computing Whittle index can be viewed as a refinement of this work. Let us recall once again that our approach also allows one to check the indexability of general restless bandits.

In the following, we show how we can refine the work of [AM20] to obtain an algorithm that is exactly the same as ours. Let D^π and N^π be two vectors defined as in [AM20] (we use the same notation, D^π and N^π, as the cited paper),

$$D^\pi = (1-\gamma)(\boldsymbol{I}-\gamma\boldsymbol{P}^\pi)^{-1}\boldsymbol{r}^\pi, \quad \text{and} \quad N^\pi = (1-\gamma)(\boldsymbol{I}-\gamma\boldsymbol{P}^\pi)^{-1}\boldsymbol{\pi}.$$

Then, we have $D^\pi - \lambda N^\pi = (1-\gamma)\boldsymbol{u}^\pi(\lambda)$ where $\boldsymbol{u}^\pi(\lambda)$ is defined as in (6.21). In our proposition, at each iteration k, we compute μ_i^k by Line 12. Instead, it is defined in [AM20] by two steps:

1. for all state $j \in [n]$ such that $N_j^{\pi^k\setminus\{i\}} \neq N_j^{\pi^k}$, one computes $\mu_{ij}^k = \dfrac{D_j^{\pi^k\setminus\{i\}} - D_j^{\pi^k}}{N_j^{\pi^k\setminus\{i\}} - N_j^{\pi^k}}$

2. compute $\mu_i^k = \operatorname*{arg\,min}_{j\in[n]:N_j^{\pi^k\setminus\{i\}}\neq N_j^{\pi^k}} \mu_{ij}^k$.

From [AM20, Theorem 2], in an indexable problem, for state σ^k, there exists a state $j \in [n]$ such that $N_j^{\pi^k\setminus\{\sigma^k\}} \neq N_j^{\pi^k}$. Now, suppose that for any active state $i \in \pi^k$, there exists $j \in [n]$ such that $N_j^{\pi^k\setminus\{i\}} \neq N_j^{\pi^k}$. Using the Sherman-Morrison formula, we have [4]

$$D^{\pi^k\setminus\{i\}} - D^{\pi^k} = -\frac{(1-\gamma)\delta_i + \tilde{\boldsymbol{\Delta}}_i D^{\pi^k}}{1+\tilde{\boldsymbol{\Delta}}_i[(\boldsymbol{I}-\gamma\boldsymbol{P}^{\pi^k})^{-1}]_{:i}}[(\boldsymbol{I}-\gamma\boldsymbol{P}^{\pi^k})^{-1}]_{:i},$$

$$\text{and} \quad N^{\pi^k\setminus\{i\}} - N^{\pi^k} = -\frac{(1-\gamma) + \tilde{\boldsymbol{\Delta}}_i N^{\pi^k}}{1+\tilde{\boldsymbol{\Delta}}_i[(\boldsymbol{I}-\gamma\boldsymbol{P}^{\pi^k})^{-1}]_{:i}}[(\boldsymbol{I}-\gamma\boldsymbol{P}^{\pi^k})^{-1}]_{:i}.$$

[4] the expression of $D^{\pi\setminus\{i\}}$ and $N^{\pi\setminus\{i\}}$ given by Equation (19) in [AM20] are erroneous.

Then, for any $j \in [n]$ such that $N_j^{\pi^k \setminus \{i\}} \neq N_j^{\pi^k}$, $\mu_{ij}^k = \dfrac{(1-\gamma)\delta_i + \tilde{\boldsymbol{\Delta}}_i D^{\pi^k}}{(1-\gamma) + \tilde{\boldsymbol{\Delta}}_i N^{\pi^k}}$ which does not depend on j. Then, we simply have $\mu_i^k = \dfrac{(1-\gamma)\delta_i + \tilde{\boldsymbol{\Delta}}_i D^{\pi^k}}{(1-\gamma) + \tilde{\boldsymbol{\Delta}}_i N^{\pi^k}}$. Also, we have

$$\begin{aligned}
\tilde{\boldsymbol{\Delta}}_i N^{\pi^k} &= (1-\gamma)\tilde{\boldsymbol{\Delta}}_i(\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi^k})^{-1}\boldsymbol{\pi}^k = -(1-\gamma) y_i^k \quad \text{and} \\
\tilde{\boldsymbol{\Delta}}_i D^{\pi^k} &= (1-\gamma)\tilde{\boldsymbol{\Delta}}_i \boldsymbol{u}^{\pi^k}(\mu_{\min}^{k-1}) + \mu_{\min}^{k-1}\tilde{\boldsymbol{\Delta}}_i N^{\pi^k} = (1-\gamma) z_i^{k-1} - (1-\gamma)\mu_{\min}^{k-1} y_i^k.
\end{aligned}$$

So, replacing these terms in μ_i^k, we get the formula in Equation (6.11) of our work.

Note that the algorithm of [AM20] was only developed for the discounted case. Our approach for the average reward case is different because we use the active advantage function defined in (6.1) instead of working with the expected discounted total reward D^{π^k} and total number of activations N^{π^k} under policy π^k. Note that the counterpart of D^{π^k} in average reward criterion is the average reward g^{π^k} and as we have seen in the previous chapter, utilizing gain optimality is not rich enough for MDPs with transient states. In addition, our code is also optimized to avoid unnecessary computation and to reduce memory usage. Finally, the way we do the update of our matrix $\boldsymbol{X}$ makes it possible to obtain a subcubic algorithm whereas their approach does not (see also below).

6.9.2 Comparison with the algorithm of [Niñ20]

The algorithm [Niñ20] has the best complexity up to date for discounted restless bandit. There is a square matrix $\boldsymbol{A}$ that plays a similar role as the square matrix $\boldsymbol{X}$ in our proposed algorithm. The most costly operations in the algorithm of [Niñ20] is to update their matrix $\boldsymbol{A}$ at each iteration, and it is done by Equation (6.18) that we recall here (using the same notation $\boldsymbol{A}$ as the cited paper):

$$\text{for } i, j \in \pi^k, \; \boldsymbol{A}_{ij}^{k+1} = \boldsymbol{A}_{ij}^k - \frac{\boldsymbol{A}_{i\sigma^k}^k}{\boldsymbol{A}_{\sigma^k\sigma^k}^k}\boldsymbol{A}_{\sigma^k j}^k. \tag{6.22}$$

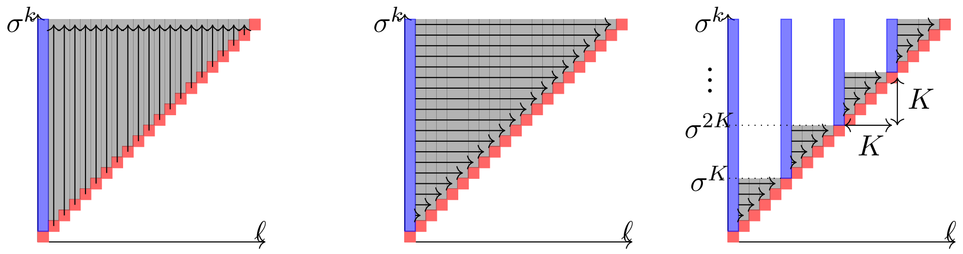
This incurs a total complexity of $(2/3)n^3 + O(n^2)$ arithmetic operations. As mentioned in Section 6.5.3, if we updated $\boldsymbol{X}^{k+1}$ as given by (6.18), our algorithm would also have a $(2/3)n^3 + O(n^2)$ complexity, but this version of update cannot be optimized by using fast matrix multiplication.

6.9.3 Their approach cannot be directly transformed into a subcubic algorithm

In addition to all the previously cited differences, one of the major contribution of our algorithm with respect to [AM20; Niñ20] is that the most advanced version of our algorithm runs in a subcubic time. The approach[5] used in [AM20; Niñ20] is to update the full matrix $\boldsymbol{X}^{k+1}$ at iteration k, by using (6.22). This idea is represented in Figure 6.9(a): for a given ℓ, their algorithm compute $\boldsymbol{X}^{\ell}_{:\sigma^k}$ for all k (i.e., the full vertical lines represented by arrows). Our first Subroutine 4 uses a horizontal approach based on (6.19), which we recall here:

$$X^{\ell+1}_{i\sigma^k} = X^{\ell}_{i\sigma^k} - X^{\ell+1}_{i\sigma^\ell} X^{\ell}_{\sigma^\ell\sigma^k}.$$

At iteration $k+1$, we use $\boldsymbol{X}^{1}_{:\sigma^k}$ to compute all values of $\boldsymbol{X}^{\ell}_{:\sigma^k}$ up to $\ell = k+1$. This is represented in Figure 6.9(b). Our approach can be used to obtain the subcubic algorithm illustrated in Figure 6.9(c) by using subcubic algorithms for multiplication.



(a) [AM20; Niñ20] use (6.22). (b) Subroutine 4. (c) Subroutine 5.

Figure 6.9.: Comparison of the computation load of (6.22) used in [AM20; Niñ20] with the one of Subroutine 4 and Subroutine 5.

This leads to the next fundamental question: why should the computation of Whittle index be harder than matrix inversion (or multiplication)? To us, the main difference is that when computing Whittle indices, the permutation σ is not known a priori but discovered as the algorithm progresses: σ^k is only known at iteration k. Hence, while all terms of the matrices X^k_{ij} are not needed, it is difficult to know a priori which ones are needed and which ones are not. Hence, a simple divide and conquer algorithm cannot be used. This is why when recomputing $\boldsymbol{X}^{k+1}$ in Subroutine 5, we recompute the whole matrix (the vertical blue line) and not just the part that will be used to compute the gray zone: we do not know *a priori* what part of X^{k+1}_{ij} will be useful or not.

[5]Equation (18) of [AM20], which is central to their algorithm is the same as the above equation (6.22).

6.10 Conclusion

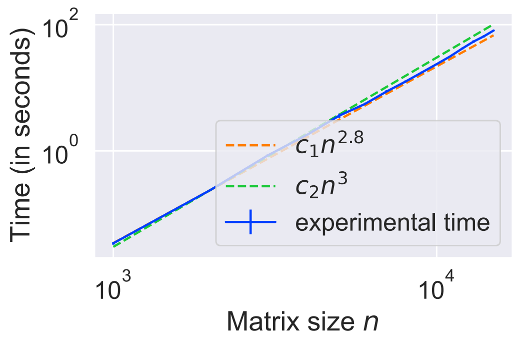
In this chapter, we presented an algorithm that was efficient for detecting the non-indexability and computing the Whittle index of all indexable finite-state restless bandits whose arms are unichain. Without any assumption on the arm's structure, this algorithm can still test indexability and compute Whittle index of some arms that are multichain and remains efficient if it is able to do so. Our algorithm is based on the efficient application of the Sherman-Morrison formula. This is a unified algorithm that works for both discounted and average reward criteria, and can be used for Gittins index computation. We presented the first version of our algorithm that runs in $n^3 + o(n^3)$ arithmetic operations (or in $(2/3)n^3 + o(n^3)$ if we do not test indexability). So, we conclude that Whittle index is not harder to compute than Gittins index. The second version of our algorithm uses the fastest matrix multiplication method and has a theoretical complexity of $O(n^{2.5286})$. This makes it the first subcubic algorithm to compute Whittle index or Gittins index. We provided numerical simulations that show that our algorithm is very efficient in practice: it can test indexability and compute the index of an n-state restless bandit arm in less than one second for $n = 1000$, and in a few minutes for $n = 15000$. These numbers are provided for dense matrices. One might expect to have more efficient algorithms if the arm has a sparse structure. We leave this question for future work.

Appendix of the chapter

6.A Implementations

6.A.1 Analysis of the experimental time to solve a linear system

In this section, we report in Figure 6.10 the time taken by the default implementation to solve a linear system of the form $\boldsymbol{AX} = \boldsymbol{B}$ where $\boldsymbol{A}$ and $\boldsymbol{B}$ are two square matrices. To obtain this figure, we generated random (full) matrices where each entry is between 0 and 1 and use the function `scipy.linalg.solve` from the library `scipy`. The reported numbers suggest that the complexity of the solver is closer to $O(n^{2.8})$ than to $O(n^3)$, although we agree that the difference between the $O(n^{2.8})$ and the $O(n^3)$ curves is small. Note that this is in accordance with the papers [Hua+16; Hua18] that claim that the fastest implementations of matrix multiplication and inversion are based on Strassen's algorithm and should therefore be in $O(n^{2.8})$.



n	Time (in second)
1000	0.03 ± 0.02
2000	0.24 ± 0.01
4000	1.83 ± 0.05
5000	3.6 ± 0.3
7000	8.7 ± 0.5
10000	23.7 ± 0.6
13000	54.0 ± 2.5
15000	80.8 ± 1.8

Figure 6.10.: Time taken of the default implementations `scipy.linalg.solve` of scipy to solve a linear system $\boldsymbol{AX} = \boldsymbol{B}$ where $\boldsymbol{A}$ and $\boldsymbol{B}$ are two square $n \times n$ matrices.

6.A.2 Arithmetic complexity of Subroutine 4 and memory usage

Recall that in Subroutine 4, we compute the values X^{ℓ}_{ij} by doing the update (for all iteration k, for all $\ell = 1$ to k and for all $i \in [n]$ or all $i \in \pi^{\ell+1}$ if we do not test indexability):

$$X^{\ell+1}_{i\sigma^k} = X^{\ell}_{i\sigma^k} - \frac{X^{\ell}_{i\sigma^\ell}}{1 + X^{\ell}_{\sigma^\ell\sigma^\ell}} X^{\ell}_{\sigma^\ell\sigma^k} \tag{6.23}$$

If we test indexability, there are $\sum_{k=1}^{n} kn = n^3/2 + O(n^2)$ such updates. If we do not test indexability, there are $\sum_{k=1}^{n}\sum_{\ell=1}^{k}(n-\ell) = n^3/3 + O(n^2)$ such updates. Below, we show each update of Equation (6.23) can be done in two arithmetic operations (one addition and one multiplication), which leads to the complexity of $n^3 + O(n^2)$ (or $(2/3)n^3 + O(n^2)$) arithmetic operations for the computation of all the needed X^k_{ij}. We also show how to reduce the memory size to $O(n^2)$.

Let $W_{i\ell} := X^{\ell}_{i\sigma^\ell}/(1 + X^{\ell}_{\sigma^\ell\sigma^\ell})$ and $V_i := X^{\ell}_{i\sigma^k}$. Using this, Equation (6.23) can be rewritten as:

$$V_i = V_i - W_{i\ell}V_{\sigma^\ell}. \tag{6.24}$$

This results in the following loop at iteration k:

- Initialize V_i from $X^1_{i\sigma^k}$.
- For all $\ell \in \{1, \dots, k-1\}$, and all $i \in [n]$ (or $i \in \pi^{\ell+1}$), apply (6.24).
- Compute $W_{ik} = V_i/(1 + V_{\sigma^k})$

Note that the value of $\boldsymbol{V}$ is not necessary for iteration k (only the values of $W_{i\ell}$ are needed). This shows that the algorithm can be implemented with a memory $O(n^2)$.

6.A.3 Speedup when not checking the indexability: First found, go last

When the indexability is not tested, the update (6.24) is computed for all $i \in \pi^{\ell+1}$. This creates inefficiencies (due to inefficient cache usage) because the elements V_i are not accessed sequentially.

To speed up the memory accesses, our solution is to sort the items during the execution of the algorithm. At iteration k, the algorithm computes σ^k. When this is done, our implementation switches all quantities in positions σ^k and $n-k+1$.

These quantities are $\delta, y, z, \boldsymbol{W}$ and $\boldsymbol{X}$. For instance, once σ^1 is found, we know that the state at position n is state n, and we do the following switches:

$$\begin{aligned}
\delta_{\sigma^1}, \delta_n &\to \delta_n, \delta_{\sigma^1} \\
y^1_{\sigma^1}, y^1_n &\to y^1_n, y^1_{\sigma^1} \\
z^1_{\sigma^1}, z^1_n &\to z^1_n, z^1_{\sigma^1} \\
\boldsymbol{W}_{\sigma^1:}, \boldsymbol{W}_{n:} &\to \boldsymbol{W}_{n:}, \boldsymbol{W}_{\sigma^1:} \\
\text{and } \boldsymbol{X}_{\sigma^1:}, \boldsymbol{X}_{n:} &\to \boldsymbol{X}_{n:}, \boldsymbol{X}_{\sigma^1:}.
\end{aligned}$$

To do so, we need an array to store all states such that at iteration k, the first $n - k$ states of the array are the active states. We will need to track the position of each state in such array.

Part III

Learning in Markovian bandits

7 Learning Algorithms for Rested Markovian Bandits

In this chapter, we study the scalability of model-based algorithms learning the optimal policy of the discounted rested Markovian bandit problem with n arms. We construct variants of three algorithms specially tailored to Markovian bandits (MB) that we call MB-PSRL, MB-UCRL2, and MB-UCBVI. We consider an episodic setting with geometrically distributed episode length and measure the performance of the algorithm in terms of regret (Bayesian regret for MB-PSRL, and expected regret for MB-UCRL2 and MB-UCBVI). For this setting, we prove that all algorithms have a low regret in $\tilde{\mathcal{O}}(S\sqrt{nK})$ – where K is the number of episodes, n is the number of arms, and S is the number of states of each arm. Up to a factor $\sqrt{S}$, these regrets match a Bayesian minimax regret lower bound of $\Omega(\sqrt{SnK})$ that we also derive.

Even if their theoretical regrets are comparable, the *time complexities* of the three agorithms vary greatly: We show that the time complexity of MB-UCRL2 and all optimistic algorithms that use confidence bonuses on transition matrices grows exponentially in n. In contrast, MB-UCBVI does not use bonuses on transition matrices, and we show that it can be implemented efficiently, with a time complexity linear in n. Our numerical experiments show, however, that its empirical regret is large. Our Bayesian algorithm, MB-PSRL, enjoys the best of both worlds: its running time is linear in the number of arms, and its empirical regret is the smallest of all three algorithms. This is a new addition to understanding the power of Bayesian algorithms, which can often be tailored to the structure of the problems to learn. This study is published in Transactions on Machine Learning Research (TMLR) ([GGK22]).

Section 7.2 describes the work in learning with structured MDPs such as factored MDPs and rested bandit with average reward criterion. We recall the formulation of the rested bandit problem with the discount factor γ in Section 7.3. Section 7.4 contains the episodic learning setup in which the episode length is geometrically distributed. We present the modification of PSRL, UCRL2, and UCBVI to rested bandits with the discount factor γ in Section 7.5. We provide an analysis of their regret in Section 7.6. Section 7.7 contains our discussion on the scalability of each algorithm. We report numerical evidence about each algorithm's learning performance and runtime in Section 7.8. Section 7.9 summarizes the theoretical and numerical results of the modified algorithms and concludes the chapter.

7.1 Contributions

We consider an episodic setting with a geometrically distributed episode length, in which the optimal strategy is the Gittins index policy. We study a specialization of PSRL [ORV13] to Markovian bandits. We call it Markovian bandit posterior sampling (MB-PSRL), which consists in using PSRL with a prior tailored to Markovian bandits. We show that the Bayesian regret of MB-PSRL is sublinear in the number of episodes and arms. We also provide an expected regret guarantee for two optimistic algorithms that we call MB-UCRL2 and MB-UCBVI. They are, respectively, a modification of UCRL2 [JOA10] and UCBVI [AOM17]. They use modified confidence bounds adapted to Markovian bandit problems. The upper bound for their regret is similar to the bound for MB-PSRL. This shows that in terms of regret, the Bayesian approach (MB-PSRL) and the optimistic approach (MB-UCRL2 and MB-UCBVI) scale well with the number of arms. We also provide a Bayesian minimax regret lower bound for any learning algorithm in the rested Markovian bandit problem with the aforementioned setting, which shows that the regret bounds that we obtain for the three algorithms are close to optimal.

The situation is radically different when considering the processing time: the runtime of MB-PSRL is linear (in the number of arms), while the runtime of MB-UCRL2 is exponential. We show that this is not an artifact of our implementation of MB-UCRL2 by exhibiting a Markovian bandit problem for which being optimistic in each arm is not optimistic in the global MDP. This implies that UCRL2 and its variants [BMT20; Fru+18; TM18; FCG10] cannot be adapted to have efficient runtime in Markovian bandit problems unless an oracle gives the optimistic policy. We argue that this non-scalability of UCRL2 and its variants is not a limitation of the optimistic approach but comes from the fact that UCRL2 relies on extended value iteration [JOA10] needed to deal with upper confidence bounds on the transition matrices. We show that MB-UCBVI, an optimistic algorithm that does not add any bonus on transition probabilities and hence does not rely on extended value iteration, does not suffer from the same problem. Its regret is sublinear in the number of episodes and arms (although larger than the regret of both MB-PSRL and MB-UCRL2), and its runtime is linear in the number of arms. This allows us to conclude that, on the one hand, if a weakly coupled MDP or a factored MDP can be solved efficiently when all the parameters are known, then the Bayesian approach is efficient both in terms of learning and computation time. On the other hand, knowing how to solve a weakly coupled MDP or a factored MDP efficiently is not sufficient for all optimistic algorithms to be computationally efficient.

We also conduct a series of numerical experiments to compare the performance of MB-PSRL, MB-UCRL2, and MB-UCBVI. They confirm the good behavior of MB-PSRL, both in terms of regret and computational complexity. Furthermore, these numerical experiments also show that the empirical regret of MB-UCBVI is larger than the regret of MB-PSRL and MB-UCRL2, confirming the comparisons between the upper bounds derived in Theorem 7.2. All this makes MB-PSRL the better choice between the three learning algorithms.

7.2 Related work

Markovian bandits have been applied to many problems such as single-machine scheduling, choosing a job in resource constraint problems as well as other industrial research problems. Many applications can be found in [Put14, Section 3.6] and [GGW11]. [Git79] shows that the discounted rested Markovian bandit can be solved linearly in the number of arms using Gittins index policy. Therefore, several papers are focused on the complexity of computing Gittins index (see, e.g., [GGK23; CM14]).

In this chapter, we focus on the rested Markovian bandit problems with a discount factor $\gamma < 1$ where all reward functions and transition matrices are unknown. A possible approach to learn under these conditions is to ignore the problem's structure and view the Markovian bandit problem as a generic MDP. As we have seen in Chapter 3, there are two main families of model-based general-purpose reinforcement learning (RL) algorithms with a regret guarantee. The first one uses the *optimism in face of uncertainty* (OFU) principle and the second uses a Bayesian approach, the posterior sampling method introduced by [Tho33]. Yet, applied as-is to Markovian bandit problems, the regret bound of these general-purpose algorithms grows exponentially with the number of arms.

Our work is not the first attempt to exploit the structure of an MDP to improve learning performance. Factored MDPs (the state space can be factored into $n \in \mathbb{N}^*$ components) are investigated in [Gue+03], where the asymptotic convergence to the optimal policy is proved to scale polynomially in the number of components. The regret of learning algorithms in factored MDPs with a factored action space is considered by [TQS20; RM20; XT20; OV14]. Our work differs substantially from these. First, the Markovian bandit problem is not a factored MDP because the action space is global and cannot be factored. Second, our reward is discounted over an infinite horizon while factored MDPs have been analyzed with no discount. Finally, and most importantly, the factored MDP framework assumes that the successive optimal policies are computed by an unspecified solver (oracle). There is no guarantee that the time complexity of this solver scales linearly with the number of components, especially for OFU-based algorithms. For rested bandits with discount, we get additional leverage: when all parameters are known, the Gittins index policy is optimal, and its computational complexity is linear in the number of arms. This reveals an interesting difference between Bayesian and extended value based algorithms (the former being scalable and not the latter in general). This difference is absent from the literature on factored MDPs because such papers do not consider the time complexity.

For rested bandits, [TL12] consider an average reward setting, $\gamma = 1$, and provide algorithms with a logarithmic regret guarantee for rested and restless settings. However, they consider a notion of regret known as *weak regret*, which measures how fast the learning algorithm identifies the best arm in the stationary regime. So, it ignores the learning behavior at the beginning of the learning process. In contrast, we consider a discounted rested bandit setting in which the regret of [TL12] makes no more sense due to the discount factor, and we propose a regret definition similar to what is frequently used in RL literature. This regret captures the learning algorithm's performance during the whole learning process. In addition, [Ort+12; JT19; WHL20] consider partially observed restless bandit setting in which the learner observe only the state of chosen arms. [Ort+12; WHL20] propose optimistic algorithms for the infinite-horizon setting and provide regret bounds that are sublinear in time. Again the discounted case is not considered in these papers, while it is particularly interesting because learning algorithms can leverage the optimal Gittins index policy. [JT19] propose a Bayesian algorithm in the episodic finite-horizon setting and also provide a regret bound that is sublinear in the number of episodes. However, the computational complexity is not studied in their work (the algorithm of [Ort+12] is intractable while the ones of [JT19; WHL20] rely on an unspecified problem solver called *oracle*). Contrarily, we provide both performance guarantee and computational complexity analysis of each algorithm we consider in this chapter. Finally, [KPT21] consider a more general setting of restless bandits in which each arm is an MDP with multiple actions. The learner has to decide which arms to activate and which action to execute on each activated arm under a global action constraint. The authors propose a suboptimal Lagrangian policy to solve the restless bandit problem with known parameters and a sampling algorithm to learn their Lagrangian policy when the parameters are unknown. Unfortunately, no performance guarantee is provided in their work.

Since index policies scale with the number of arms, using Q-learning approach to learn such a policy is also popular, see *e.g.*, [AB22; Fu+19; Duf95]. [Duf95] address the same Markovian bandit problem as we do: their algorithm learns the optimal value in the restart-in-state MDP [KV87] for each arm and uses Softmax exploration to solve the exploration-exploitation dilemma. As mentioned on page 250 of [ACF02], however, there exists no finite-time regret bound for this algorithm. Furthermore, tuning its hyperparameters (learning rate and temperature) is rather delicate and unstable in practice.

7.3 Rested Markovian bandit with discount

We recall from Chapter 4 that a rested Markovian bandit is a multi-armed bandit having $n \in \mathbb{N}^+$ arms. Each arm $\langle \mathcal{S}_a, \boldsymbol{r}_a, \boldsymbol{P}_a \rangle$ for $a \in \{1, \dots, n\} =: [n]$ is a Markov reward process with a finite state space $\mathcal{S}_a$ of size S. Without loss of generality, we assume that the state spaces of the arms are pairwise distinct: $\mathcal{S}_a \cap \mathcal{S}_b = \emptyset$ for $a \neq b$. In the following, the state of an arm a will always be denoted with an index a: we will denote such a state by s_a or s'_a. As state spaces are disjoint, this allows us to simplify the notation by dropping the index a from the reward and transition matrix: when convenient, we will denote them by $r(s_a)$ instead of $r_a(s_a)$ and by $P(s_a, s'_a)$ instead of $P_a(s_a, s'_a)$ since no confusion is possible.

At time 1, the global state $\boldsymbol{s}_1$ is distributed according to some initial distribution ρ over the global state space $\mathcal{X} := \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$. At time $t \geq 1$, the decision maker observes the states of all arms, $\boldsymbol{s}_t = (s_{t,1} \ \dots \ s_{t,n})$, and chooses which arm a_t to activate. This problem can be cast as an MDP – that we denote by M – with state space $\mathcal{X}$ and action space $[n]$. Let $a \in [n]$ and $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{X}$. If the state at time t is $\boldsymbol{s}_t = \boldsymbol{s}$, the chosen arm is $a_t = a$, then arm a incurs a random reward r_t drawn from some distribution on $[0, 1]$ with mean $r(s_a)$ and the MDP M transitions to a new state $\boldsymbol{s}_{t+1} = \boldsymbol{s}'$ with probability $p(\boldsymbol{s}' \mid \boldsymbol{s}, a)$ that satisfies:

$$p(\boldsymbol{s}' \mid \boldsymbol{s}, a) = \begin{cases} P(s_a, s'_a) & \text{if } s_b = s'_b \text{ for all } b \neq a; \\ 0 & \text{otherwise.} \end{cases} \tag{7.1}$$

That is, the activated arm makes a transition while the other arms remain in the same state.

Let Π be the set of deterministic policies, *i.e.,* the set of functions $\pi : \mathcal{X} \mapsto [n]$. For the MDP M, we denote by $v^{\pi}_{M}(\boldsymbol{s})$ the expected cumulative discounted reward of M under policy π starting from an initial state $\boldsymbol{s}$:

$$v^{\pi}_{M}(\boldsymbol{s}) = \mathbb{E}^{\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid \boldsymbol{s}_1 = \boldsymbol{s}\right].$$

An alternative definition of v is to consider a finite-horizon problem with a geometrically distributed length. Indeed, let H be a time-horizon geometrically distributed with parameter $1-\gamma > 0$. We have

$$v^{\pi}_{M}(\boldsymbol{s}) = \mathbb{E}\left[\mathbb{E}^{\pi}\left[\sum_{t=1}^{H} r_t \mid \boldsymbol{s}_1 = \boldsymbol{s}\right]\right]. \tag{7.2}$$

Problem 7.1
Given a rested bandit M with n arms, each is a Markov reward process $\langle \mathcal{S}_a, \boldsymbol{r}_a, \boldsymbol{P}_a \rangle$ with a finite state space $\mathcal{S}_a$, find a policy $\pi : \mathcal{S}_1 \times \ldots \times \mathcal{S}_n \mapsto [n]$ that maximizes $v^{\pi}_{M}(\boldsymbol{s})$ for any state $\boldsymbol{s}$ distributed according to initial global state distribution ρ.

By a small abuse of notation, we denote by $v^{\pi}_{M}(\rho)$ the expected reward when the initial state is randomly generated according to $\rho : v^{\pi}_{M}(\rho) = \sum_{\boldsymbol{s}} \rho(\boldsymbol{s}) v^{\pi}_{M}(\boldsymbol{s})$. A policy π^* is optimal for Problem 7.1 if $v^{\pi^*}_{M}(\boldsymbol{s}) \geq v^{\pi}_{M}(\boldsymbol{s})$ for all $\pi \in \Pi$ and $\boldsymbol{s} \in \mathcal{X}$. As we have seen in Chapter 2, such a policy exists and does not depend on $\boldsymbol{s}$ (or ρ). One well-known optimal policy is the Gittins index policy as presented in Chapter 4.

7.4 Online learning and episodic regret

We now consider an extension of Problem 7.1 in which the decision maker does not know the transition matrices nor the rewards. Our goal is to design a reinforcement learning algorithm that learns the optimal policy from past observations. Similarly to what is done for finite-horizon reinforcement learning with deterministic horizon – see Section 3.3.1 – we consider a decision maker that faces a sequence of independent replicas of the same rested bandit problem, where the transitions and the rewards are drawn independently for each episode. What is new here is that the time horizon H is random and has a geometric distribution with expected value $1/(1-\gamma)$. It is drawn independently for each episode. This implies that Gittins index policy is an optimal policy in this setting.

Let $H^1, \dots, H^k$ be the sequence of random episode lengths and let $t^k := 1+\sum_{i=1}^{k-1} H^i$ be the starting time of the kth episode. The observations made prior and up to episode k is denoted $o_{t^k} := \{\boldsymbol{s}_1, a_1, r_1, \dots, \boldsymbol{s}_{t^k-1}, a_{t^k-1}, r_{t^k-1}, s_{t^k}\}$. An *Episodic Learning Algorithm* $\mathcal{L}$ is a function that maps observations o_{t^k} to $\mathcal{L}(o_{t^k})$, a probability distribution whose support is a subset of Π. We recall from Algorithm 1 that at the beginning of episode k, the algorithm samples $\pi^k \sim \mathcal{L}(o_{t^k})$ and uses this policy during the whole kth episode. Note that one could also design algorithms where learning takes place inside each episode. We will see later that episodic learning as described here is enough to design algorithms that are essentially optimal, in the sense given by Theorem 7.2 and Theorem 7.3.

Similarly to Definition 3.1, for an instance M of a rested bandit problem and a total number of episodes K, the regret of a learning algorithm $\mathcal{L}$ is defined by

$$\mathrm{Regret}(\mathcal{L}, M, K) := \sum_{k=1}^{K} v_M^{\pi^*}(\boldsymbol{s}_{t^k}) - v_M^{\pi^k}(\boldsymbol{s}_{t^k}). \tag{7.3}$$

It is the sum over all episodes of the value of the optimal policy π^* minus the value obtained by applying the policy π^k chosen by the algorithm for episode k. In what follows, we will provide bounds on the expected regret.

Recall that a "good" algorithm $\mathcal{L}$ is such that its expected regret $\mathbb{E}\left[\mathrm{Regret}(\mathcal{L}, M, K)\right]$ grows sub-linearly in the number of episodes K. This implies that the expected regret over episode k converges to 0 as k goes to infinity. Such an algorithm learns an optimal policy of Problem 7.1.

Note that, for discounted MDPs, an alternative regret definition (used for instance by [HZG21b]) is to use the non-episodic version $\sum_{t=1}^{T}(v_M^{\pi^*}(\boldsymbol{s}_t) - v_M^{\pi_t}(\boldsymbol{s}_t))$. In our definition at Equation (7.3), we use an episodic approach where the process is restarted according to ρ after each episode of geometrically distributed length H^k.

7.5 Learning algorithms for rested Markovian bandits

In what follows, we present three algorithms having a regret that grows like $\tilde{O}(S\sqrt{nK})$, that we call MB-PSRL, MB-UCRL2 and MB-UCBVI. As their names suggest, these algorithms are adaptation of PSRL, UCRL2 and UCBVI to rested bandit problems that intend to overcome the exponentiality in n of their regret. The structure of the three MB-* algorithms are similar and is represented in Algorithm 6. All algorithms are episodic learning algorithms. At the beginning of each episode

k, an MB-* learning algorithm computes a new policy π^k that will be used during the episode of geometrically distributed length. The difference between the three algorithms lies in the way this new policy π^k is computed. MB-PSRL uses posterior sampling while MB-UCRL2 and MB-UCBVI use the optimism. We detail the three algorithms below.

Algorithm 6: Pseudocode of the three MB-* algorithms.

```
Input  : Discount factor $\gamma$, initial distribution $\rho$ (and a prior distribution
         $\{\phi_a\}_{a\in[n]}$ for MB-PSRL)
1 for episodes $k = 1, 2, \dots$ do
2     Compute a new policy $\pi^k$ (using posterior sampling or optimism). ;
3     Set $t^k \leftarrow 1 + \sum_{i=1}^{k-1} H^i$, sample $\boldsymbol{s}_{t^k} \sim \rho$ and $H^k \sim \mathrm{Geom}(1-\gamma)$.;
4     for $t \leftarrow t^k$ to $t^k + H^k$ do
5         Activate arm $a_t = \pi^k(\boldsymbol{s}_t)$. ;
6         Observe $r_t$ and $\boldsymbol{s}_{t+1}$. ;
```

7.5.1 MB-PSRL

MB-PSRL starts with a prior distribution ϕ_a over the parameters $(\boldsymbol{r}_a, \boldsymbol{P}_a)$ of all arm $a \in [n]$. At the start of each episode k, MB-PSRL computes a posterior distribution $\phi_a(\cdot \mid o_{t^k})$ of the parameters for each arm $a \in [n]$ and samples parameters $(\boldsymbol{r}_a^k, \boldsymbol{P}_a^k)$ from $\phi_a(\cdot \mid o_{t^k})$. Then, MB-PSRL uses $\{(\boldsymbol{r}_a^k, \boldsymbol{P}_a^k)\}_{a\in[n]}$ to compute the Gittins index policy π^k that is optimal for the sampled problem. The policy π^k is then used for the whole episode k. Note that as π^k is a Gittins index policy, it can be computed efficiently.

The difference between PSRL and MB-PSRL is mostly that MB-PSRL uses a prior distribution tailored to Markovian bandit. The only hyperparameter of MB-PSRL is the prior distribution ϕ. As we see in Appendix 7.E, MB-PSRL seems robust to the choice of the prior distribution, even if a coherent prior gives a better performance than a misspecified prior, similarly to what happens for Thompson's sampling [Rus+18].

7.5.2 MB-UCRL2

At the beginning of each episode k, MB-UCRL2 computes the following quantities: for each state $s_a \in \mathcal{S}_a$, $N^k(s_a)$ is the number of times that Arm a is activated before episode k while being in state s_a, and $\hat{r}^k(s_a)$, and $\hat{\boldsymbol{P}}^k(s_a, \cdot)$ are the empirical means of $r(x_a)$ and $\boldsymbol{P}(s_a, \cdot)$ respectively. We define the confidence bonuses $\beta_r^k(s_a) :=$

$\sqrt{\frac{\log(2SnKt^k)}{2\max\{1,N^k(s_a)\}}}$ and $\beta_P^k(s_a) := \sqrt{\frac{2\log(SnK2^S t^k)}{\max\{1,N^k(s_a)\}}}$. This defines a confidence set $\mathbb{M}^k$ as follows: a rested bandit M' is in $\mathbb{M}^k$ if for all $a \in [n]$ and $s_a \in \mathcal{S}_a$:

$$\left|r'(s_a) - \hat{r}^k(s_a)\right| \le \beta_r^k(s_a) \text{ and } \left\|\boldsymbol{P}'(s_a,\cdot) - \hat{\boldsymbol{P}}^k(s_a,\cdot)\right\|_{\ell_1} \le \beta_P^k(s_a). \tag{7.4}$$

MB-UCRL2 then chooses a policy π^k that is optimal for the optimistic problem $M^k \in \mathbb{M}^k$:

$$\pi^k \in \arg\max_{\pi} \max_{M' \in \mathbb{M}^k} v_{M'}^{\pi}(\rho). \tag{7.5}$$

Note that as we explain later in Section 7.7, we believe that there is no efficient algorithm to compute the optimistic policy π^k of Equation (7.5) unless there is an oracle that gives the optimistic problem M^k.

Compared to a vanilla implementation of UCRL2, MB-UCRL2 uses the structure of the Markovian bandit problem: The constraints Equation (7.4) are on $\boldsymbol{P}$ whereas vanilla UCRL2 uses constraints on the full transition p (defined in Equation (7.1)). This leads MB-UCRL2 to use the bonus term that scales as $\sqrt{S/N^k(s_a)}$ whereas vanilla UCRL2 would use the term in $\sqrt{S^n/N^k(\boldsymbol{s},a)}$.

7.5.3 MB-UCBVI

At the beginning of episode k, MB-UCBVI uses the same quantities $N^k(s_a)$, $\hat{r}^k(s_a)$, and $\hat{\boldsymbol{P}}^k(s_a,\cdot)$ as MB-UCRL2. The difference lies in the definition of the bonus terms. While MB-UCRL2 uses a bonus on the reward and on the transition matrices, MB-UCBVI defines a bonus $\beta^k(s_a) := \frac{1}{1-\gamma}\sqrt{\frac{\log(2SnKt^k)}{2\max\{1,N^k(s_a)\}}}$ that is used on the reward only. MB-UCBVI computes the Gittins index policy π^k that is optimal for the bandit problem $\{(\hat{\boldsymbol{r}}_a^k + \beta^k, \hat{\boldsymbol{P}}_a^k)\}_{a\in[n]}$.

Similarly to the case of UCRL2, a vanilla implementation of UCBVI would use a bonus that scales exponentially in the number of arms. MB-UCBVI makes an even better use of the structure of the learned problem because the optimistic MDP $\{(\hat{\boldsymbol{r}}_a^k + \beta^k, \hat{\boldsymbol{P}}_a^k)\}_{a\in[n]}$ is still a Markovian bandit problem. This implies that the optimistic policy π^k is a Gittins index policy, and that can therefore be computed efficiently.

7.6 Regret analysis

In this section, we first present upper bounds on the expected regret of the three learning algorithms. These bounds are sublinear in the number of episodes and sublinear in the number of arms. We then derive a minimax lower bound on the Bayesian regret of any learning algorithm in the rested Markovian bandit problem with discount.

7.6.1 Upper bounds on regret

The theorem below provides upper bounds on the regret of the three algorithms presented in Section 7.5.

Theorem 7.2 (Regret upper bounds)
Let $f(S,n,K,\gamma) = Sn\Big(\ln K/(1-\gamma)\Big)^2 + \sqrt{SnK}\Big(\ln K/(1-\gamma)\Big)^{3/2}$. *There exists universal constants* C, C' *and* C'' *independent of the model (i.e., that do not depend on* S, n, K *and* γ*) such that:*

- *For any prior distribution* ϕ*:*

$$\mathrm{BayesRegret}(\text{MB-PSRL}, \phi, K) \le C\left(\sqrt{S} + \ln\frac{SnK\ln K}{1-\gamma}\right) f(S,n,K,\gamma),$$

- *For any Markovian bandit model* M*:*

$$\mathbb{E}\left[\mathrm{Regret}(\text{MB-UCRL2}, M, K)\right] \le C'\left(\sqrt{S} + \ln\frac{SnK\ln K}{1-\gamma}\right) f(S,n,K,\gamma),$$

$$\mathbb{E}\left[\mathrm{Regret}(\text{MB-UCBVI}, M, K)\right] \le C''\left(\frac{\sqrt{S}}{1-\gamma}\right)\left(\ln\frac{SnK\ln K}{1-\gamma}\right) f(S,n,K,\gamma).$$

We provide a sketch of proof below. The detailed proof is provided in Appendix 7.A.

This theorem calls for several comments. First, it shows that when $K \ge Sn/(1-\gamma)$, the regret of MB-PSRL and MB-UCRL2 is smaller than

$$\tilde{O}\left(\frac{S\sqrt{nK}}{(1-\gamma)^{3/2}}\right), \tag{7.6}$$

where the notation $\tilde{O}$ means that all logarithmic terms are removed. The regret of MB-UCBVI has an extra $1/(1-\gamma)$ factor.

Hence, the regret of the three algorithms is sublinear in the number of episodes K which means that they all are no-regret algorithms. This regret bound is sublinear in the number of arms which is very significant in practice when facing a large number of arms. Note that directly applying PSRL, UCRL2 or UCBVI would lead to a regret in $\tilde{O}\left(S^n\sqrt{nK}\right)$ or $\tilde{O}\left(\sqrt{nS^nK}\right)$, which is exponential in n.

Second, the upper bound on the expected regret of MB-UCRL2 (and of MB-UCBVI) is a guarantee for a specific problem M while the bound on Bayesian regret of MB-PSRL is a guarantee in average overall the problems drawn from the prior ϕ. Hence, the bounds of MB-UCRL2 and MB-UCBVI are stronger guarantee compared to the one of MB-PSRL. Yet, as we will see later in the numerical experiments reported in Section 7.8, MB-PSRL seems to have a smaller regret in practice, even when the problem does not follow the correct prior. An interesting open question would be to find a prior that would guarantee that MB-PSRL has a good worst-case regret bound. We do not know if such a prior exists and to the best of our knowledge, this question is also open for the classical PSRL. Note that there exist Bayesian type algorithms with worst-case guarantees, see e.g., [Ish+21; ACJ21; WSY20; AJ17], but they contain an optimistic part, and it is not clear how to implement them in an efficient manner for Markovian bandits.

Third, the result of Theorem 7.2 is the statistical evaluation of the three learning algorithms and does not require them to use Gittins index policy (in particular, MB-UCRL2 does not use Gittins index policy). What is required is that policy π^k is optimal for the sampled problem M^k for MB-PSRL (so that Lemma 7.9 applies) or for the optimistic problem M^k for MB-UCBVI (so that (7.10) is valid). Indeed, instead of using Gittins index policy for MB-PSRL or MB-UCBVI, assume that we have access to an oracle that provides an optimal policy for any given Markovian bandit problem. Then, the upper bound on regret in Theorem 7.2 still holds when MB-PSRL and MB-UCBVI use the oracle to compute policy π^k. Gittins index policy is required only for the runtime evaluation as we will see in Section 7.8.

Finally, our bound in Equation (7.6) is linear in S, the state space size of each arm. Having a regret bound linear in the state space size is currently state-of-the-art for Bayesian algorithms, see e.g., [AJ17; Ouy+17] and our discussion in Appendix 7.A.3. For optimistic algorithms, the best regret bounds are linear in the square root of the state size because they use Bernstein's concentration bounds instead of Weissman's inequality [AOM17], yet this approach does not work in our setting due to the randomness of episode's length and the bound of MB-UCBVI depends linearly on S. We discuss more about this in Appendix 7.A.5. UCBVI has also been studied in the discounted case by [HZG21b]. They use, however, a different definition of regret, making their bound on the regret hardly comparable to ours.

Sketch of proof

A crucial ingredient of our proof is to work with the value function over a random finite time horizon (W defined below), instead of working directly with the discounted value function v. For a given model M, and a deterministic policy π, a horizon H and a time step $h \leq H$, we define by $W^{\pi}_{M,h:H}(\boldsymbol{s})$ the value function of policy π over the finite time horizon $H-h+1$ when starting in $\boldsymbol{s}$ at time h. It is defined by: for any state $\boldsymbol{s} \in \mathcal{X}$, $a = \pi(\boldsymbol{s})$,

$$W^{\pi}_{M,h:H}(\boldsymbol{s}) := r(s_a) + \sum_{\boldsymbol{s}' \in \mathcal{X}} p(\boldsymbol{s}' \mid \boldsymbol{s}, a) W^{\pi}_{M,h+1:H}(\boldsymbol{s}'), \tag{7.7}$$

with $W^{\pi}_{M,H:H}(\boldsymbol{s}) := r(s_a)$. We recall from (2.7) and (7.2) that $v^{\pi}_{M}(\boldsymbol{s}) = \mathbb{E}\left[W^{\pi}_{M,1:H}(\boldsymbol{s})\right]$.

This characterization is important in our proof. Since the episode length H^k is independent of the observations o_{t^k} available before episode k, for any policy π^k that is independent of H^k, one has

$$\mathbb{E}\left[v^{\pi^k}_{M}(\boldsymbol{s}_{t^k}) \mid o_{t^k}, \pi^k\right] = \mathbb{E}\left[W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k}) \mid o_{t^k}, \pi^k\right]. \tag{7.8}$$

In the above Equation (7.8), the expectation is taken over all initial state $\boldsymbol{s}_{t^k}$ and all possible horizon H^k.

Equation (7.8) will be very useful in our analysis as it allows us to work with either v or W interchangeably. While the proof of MB-PSRL could be done by only studying the function W, the proof of MB-UCRL2 and MB-UCBVI will use the expression of the regret as a function of v to deal with the non-determinism. Indeed, at episode k, all algorithms compare the optimal policy π^* (that is optimal for the true MDP M) and a policy π^k chosen by the algorithm (that is optimal for the MDP M^k that is either sampled by MB-PSRL or chosen by an optimistic principle). The quantity $\Delta^k := W^{\pi^*}_{M,1:H^k}(\boldsymbol{s}_{t^k}) - W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k})$ equals:

$$\underbrace{W^{\pi^*}_{M,1:H^k}(\boldsymbol{s}_{t^k}) - W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k})}_{(A)} + \underbrace{W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k}) - W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k})}_{(B)}. \tag{7.9}$$

The analysis of the term (B) is similar for the three algorithms: it is bounded by the distance between the sampled MDP M^k and the true MDP M which in turn can be bounded by using a concentration argument (Lemma 7.5) based on Hoeffding's and

Weissman's inequalities. Compared with the literature [AOM17; Ouy+17], our proof leverages on taking conditional expectations, making all terms whose conditional expectation is zero disappear. One of the main technical hurdle is to deal with the random episodes lengths $H^1, \dots, H^k$. This is required in our approach and is not needed in the classical analysis of finite-horizon problems.

The analysis of (A) depends heavily on the algorithm used. The easiest case is PSRL: As our setting is Bayesian, the expectation of the first term (A) with respect to the model is zero (see Lemma 7.9). The case of MB-UCRL2 and MB-UCBVI are harder. In fact, our bonus terms are specially designed so that $v^{\pi^k}_{M^k}(\boldsymbol{s})$ is an optimistic upper bound of the true value function with high probability, that is:

$$v^{\pi^k}_{M^k}(\boldsymbol{s}) = \max_{\pi} \max_{M' \in \mathbb{M}^k} v^{\pi}_{M'}(\boldsymbol{s}) \geq v^{\pi^*}_{M}(\boldsymbol{s}). \tag{7.10}$$

This requires the use of v and not W, and it is used to show that the expectation of the term (A) of Equation (7.9) cannot be positive.

7.6.2 Bayesian minimax lower bound

After obtaining upper bounds on the regret, a natural question is: can we do better? Or in other terms, does there exist a learning algorithm with a smaller regret? As we explained in Chapter 3, this question is addressed by the notion of minimax regret lower bound: for a given set of parameters (S, n, K, γ), a minimax lower bound is a lower bound on the quantity $\inf_{\mathcal{L}} \sup_M \mathrm{Regret}(\mathcal{L}, M, K)$, where the supremum is taken among all possible models that have parameters (S, n, K, γ) and the infimum is taken over all possible learning algorithms. The next theorem provides a lower bound on the Bayesian regret. It is therefore stronger than a minimax bound for two reasons: First, the Bayesian regret is an average over models, which means that there exists at least one model that has a larger regret than the Bayesian lower bound; And second, in Theorem 7.3, we allow the algorithm to depend on the prior distribution ϕ and to use this information.

Theorem 7.3 (Minimax Bayesian regret lower bound)
For any state size S, number of arms n, discount factor γ and number of

episodes $K \geq 16S$, there is a prior distribution ϕ on rested bandit problems with parameters (S, n, K, γ) such that, for any learning algorithm $\mathcal{L}$:

$$\text{BayesRegret}(\mathcal{L}, \phi, K) \geq \frac{1}{60}\sqrt{\frac{SnK}{(1-\gamma)}}. \tag{7.11}$$

The proof is given in Appendix 7.B and uses a counterexample inspired by the one of [JOA10]. Note that for general MDPs, the minimax regret lower bound in Proposition 3.4 says that a learning algorithm cannot have a regret smaller than $\Omega(\sqrt{\tilde{S}\tilde{A}\tilde{T}})$, where $\tilde{S}$ is the number of states of the MDP, $\tilde{A}$ is the number of actions and $\tilde{T}$ is the number of time steps. Yet, this lower bound is not directly applicable to rested bandit with $\tilde{S} = S^n$ because Markovian bandit problems are very specific instances of MDPs and this can be exploited by the learning algorithm. Also note that this lower bound on the Bayesian regret is also a lower bound on the expected regret of any non-Bayesian algorithm for any MDP model M.

Apart from the logarithmic terms, the lower bound provided by Theorem 7.3 differs from the bound of Theorem 7.2 by a factor $\sqrt{S}/(1-\gamma)$. This factor is similar to the one observed for PSRL and UCRL2 [ORV13; JOA10]. There are various factors that could explain this. We believe that the extra factor $1/(1-\gamma)$ might be half due to the episodic nature of MB-PSRL and MB-UCRL2 (when $1/(1-\gamma)$ is large, algorithms with internal episodic updates might have smaller regret) and half due to the fact that the lower bound of Theorem 7.3 is not optimal and could include a term $1/\sqrt{1-\gamma}$ (similar to the term $O(\sqrt{D})$ of the lower bound of [OV16; JOA10]). The factor $\sqrt{S}$ between our two bounds comes from our use of Weissman's inequality. It might be possible that our regret bounds are not optimal with respect to this term, although such an improvement cannot be obtained using the same approach of [AOM17].

7.7 Scalability of learning algorithms for Markovian bandits

Historically, Problem 7.1 was considered unresolved until [Git79] proposed Gittins index. This is because previous solutions were based on Dynamic Programming in the global MDP which are computationally expensive. Hence, after establishing regret guarantees, we are now interested in the computational complexity of our learning algorithms, which is often disregarded in the learning literature.

7.7.1 MB-PSRL and MB-UCBVI are scalable

If one excludes the simulation of the MDP, the computational cost of MB-PSRL and MB-UCBVI of each episode is low. For MB-PSRL, its cost is essentially due to three components: Updating the observations, sampling from the posterior distribution and computing the optimal policy. The first two are relatively fast when the conjugate posterior has a closed form: updating the observation takes $O(1)$ at each time, and sampling from the posterior can be done in $O(nS^2)$ – more details on posterior distributions are given in Appendix 7.D. When the conjugate posterior is implicit (*i.e.*, under the integral form), the computation can be higher but remains linear in the number of arms. For MB-UCBVI, the cost is due to two components: computing the bonus terms and computing the Gittins policy for the optimistic MDP. Computing the bonus is linear in the number of arms and the length of the episode. As explained in Section 4.2.2 and Chapter 6, the computation of the Gittins index policy for a given rested bandit can be done in $O(nS^3)$. Hence, MB-PSRL and MB-UCBVI have a regret and a runtime both scale with the number of arms.

7.7.2 MB-UCRL2 is not scalable because it cannot use an index policy

While MB-UCRL2 has a regret equivalent to the one of MB-PSRL, its computational complexity, and in particular the complexity of computing an *optimistic* policy that maximizes Equation (7.5) does not scale with n. Such a policy can be computed by using *extended value iteration* [JOA10]. This computation is polynomial in the number of states of the global MDP and is therefore exponential in the number of arms, precisely $O(nS^{2n})$. For MB-PSRL (or MB-UCBVI), the computation is easier because the sampled (or optimistic) MDP is a rested bandit problem. Hence, using Proposition 4.1, computing the optimal policy can be done by computing local indices. In the following, we show that it is not possible to solve Equation (7.5) by using local indices. This suggests that MB-UCRL2 (nor any of the modifications of UCRL2's variants that would use extended value iteration) cannot be implemented efficiently.

More precisely, to find an optimistic policy (that satisfies Equation (7.10)), UCRL2 and its variants, e.g., KL-UCRL [FCG10], compute a policy π^k that is optimal for the optimistic MDP in $\mathbb{M}^k$. This is done by using extended value iteration. We now show that this cannot be replaced by the computation of local indices.

Let us consider that the estimates and confidence bonus for a given arm a are $\hat{\mathcal{B}}_a := (\hat{\boldsymbol{r}}_a, \hat{\boldsymbol{P}}_a, \beta_r, \beta_P)$. We say that an algorithm computes indices locally for Arm a if for each $s_a \in \mathcal{S}_a$, it computes an index $I^{\hat{\mathcal{B}}_a}(s_a)$ by using only $\hat{\mathcal{B}}_a$ but not $\hat{\mathcal{B}}_b$ for any $b \neq a$. We denote by $\pi^{I(\hat{\mathcal{B}})}$ the index policy that uses index $I^{\hat{\mathcal{B}}_a}$ for arm a and by $\mathbb{M}(\hat{\mathcal{B}})$ the set of rested bandits M' that satisfy Equation (7.4).

Theorem 7.4 (Optimistic bandit requires knowledge of all arms' bonuses) *For any algorithm that computes indices locally, there exists a rested bandit M, an initial state $\boldsymbol{s}$ and estimates $\hat{\mathcal{B}}_a := (\hat{\boldsymbol{r}}_a, \hat{\boldsymbol{P}}_a, \beta_r, \beta_P)$ such that $M \in \mathbb{M}(\hat{\mathcal{B}})$ and*

$$\sup_{M' \in \mathbb{M}(\hat{\mathcal{B}})} v_{M'}^{\pi^{I(\hat{\mathcal{B}})}}(\boldsymbol{s}) < \sup_{\pi} v_M^{\pi}(\boldsymbol{s}).$$

Proof. The proof presented in Appendix 7.C is obtained by constructing a set $\mathbb{M}$ and two MDPs M_1 and M_2 in $\mathbb{M}$ such that Equation (7.10) cannot hold simultaneously for both M_1 and M_2. □

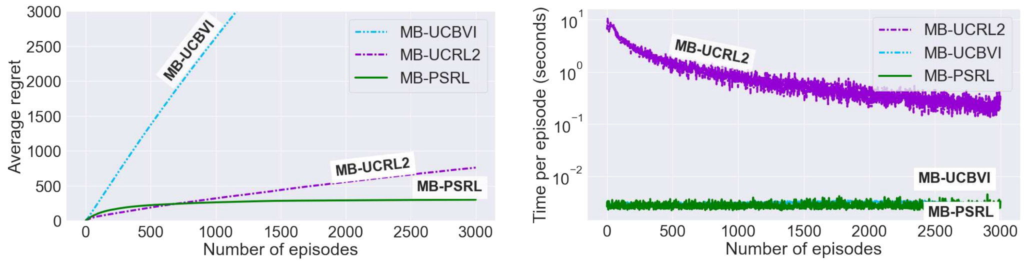
This theorem implies that one cannot define local indices such that Equation (7.10) holds for all rested bandits $M' \in \mathbb{M}^k$. Yet, the use of this inequality is central in the regret analysis of UCRL2 (see the proof of UCRL2 [JOA10]). This implies that the current methodology to obtain regret bounds for UCRL2 and its variants, e.g., [BMT20; Fru+18; TM18; FCG10], that use extended value iteration is not applicable to bound the regret of their modified version that computes indices locally.

Therefore, we believe that UCRL2 and its variants cannot compute optimistic policy locally: they should all require the joint knowledge of all $\{\hat{\mathcal{B}}_a\}_{a\in[n]}$.

7.8 Numerical experiments

In complement to our theoretical analysis, we report, in this section, the performance of our three algorithms in a model taken from the literature. The model is an environment with 3 arms, all following a Markov chain that is obtained by applying the optimal policy on the river swim MDP. A detailed description is given in Appendix 7.D, along with all hyperparameters that we used. Our numerical experiments suggest that MB-PSRL outperforms other algorithms in terms of average regret and is computationally less expensive than other algorithms. To ensure reproducibility, the code and data of our experiments are available at `https://gitlab.inria.fr/kkhun/learning-in-rested-markovian-bandit`.

Performance result We investigate the average regret and policy computation time of each algorithm. To do so, we run each algorithm for 80 simulations and for $K = 3000$ episodes per simulation. We arbitrarily choose the discount factor $\gamma = 0.99$. In Figure 7.1a, we show the average cumulative regret of the 3 algorithms. We observe that the average regret of MB-UCBVI is larger than those of MB-PSRL and MB-UCRL2. Moreover, we observe that MB-PSRL obtains the best performance and that its regret seems to grow slower than $\mathcal{O}(\sqrt{K})$. This is in accordance to what was observed for PSRL [ORV13]. Note that the expected number of time steps after K episodes is $K/(1-\gamma)$ which means that in our setting with $K = 3000$ episodes there are $300\,000$ time steps in average. In Figure 7.1b, we compare the computation time of the various algorithms. We observe that the computation time (the y-axis is in log-scale) of MB-PSRL and MB-UCBVI, the index-based algorithms, are the fastest by far. Moreover, the computation time of these algorithms seem to be independent of the number of episodes. These two figures show that MB-PSRL has the smallest regret and computation time among all compared algorithms.



(a) Average cumulative regret in function of the number of episodes.

(b) Average runtime per episode. The vertical axis is in log-scale.

Figure 7.1.: Experimental result for the three 4-state random walk arms given in Table 7.1. The x-axis is the number of episodes. Each algorithm is identified by a unique color for all figures.

Robustness (larger models and different priors) To test the robustness of MB-PSRL, we conduct two more sets of experiments that are reported in Appendix 7.E. They confirm the superiority of MB-PSRL. The first experiment is an example from [Duf95] with 9 arms each having 11 states. This model illustrates the effect of the curse of dimensionality: the global MDP has 11^9 states which implies that the runtime of MB-UCRL2 makes it impossible to use, while MB-PSRL and MB-UCBVI take a few minutes to complete 3000 episodes. Also in this example, MB-PSRL seems to converge faster to the optimal policy than MB-UCBVI. The second experiment tests the robustness of MB-PSRL to the choice of prior distribution. We provide numerical evidences that show that, even when MB-PSRL is run with a prior ϕ that is not the one from which M is drawn, the regret of MB-PSRL remains acceptable (around twice the regret obtained with a correct prior).

7.9 Conclusion

In this chapter, we present MB-PSRL, a modification of PSRL to rested Markovian bandit problems with discount. We show that its regret is close to the lower bound that we derive for this problem while its runtime scales linearly with the number of arms. Furthermore, and unlike what is usually the case, MB-PSRL does not have an optimistic counterpart that scales well: we prove that MB-UCRL2 also has a sublinear regret but has a computational complexity exponential in the number of arms. This result generalizes to all the variants of UCRL2 that rely on extended value iteration. We nevertheless show that OFU approach may still be pertinent for Markovian bandit problems: MB-UCBVI, a version of UCBVI can use Gittins indices and does not suffer from the dimensionality curse: it has a sublinear regret in terms of the number of episodes and arms as well as a linear time complexity. However, its regret bound remains larger than the one of MB-PSRL.

The broad implication of this work is that, on the one hand, if a weakly coupled MDP or factored MDP can be solved efficiently when all the parameters are known, then PSRL can be adapted to have efficient regret and runtime. On the other hand, solving weakly coupled MDP or factored MDP efficiently when all the parameters are known does not imply that all optimistic algorithms are computationally efficient. This is a major difference between the Bayesian and the optimistic approach.

Appendix of the chapter

The appendix is organized as follows:

- In Appendix 7.A, we prove Theorem 7.2.
- In Appendix 7.B, we obtain a Bayesian minimax regret lower bound for any reinforcement learning algorithm in rested bandits (Theorem 7.3).
- In Appendix 7.C, we show that Equation (7.5) cannot be solved by local indices (Theorem 7.4).
- In Appendix 7.D, we provide a detailed description of the algorithms that we use in our numerical comparisons.
- In Appendix 7.E, we provide additional numerical experiments that show the good behavior of MB-PSRL.
- In Appendix 7.F, we provide details about the experimental environment and the computation time needed.

7.A Proof of Theorem 7.2

The proof of the regret bounds for our three algorithms share a common structure but with different technical details. In this section, we do a detailed proof of the three algorithms by factorizing as much as possible what can be factorized in the different proofs. This proof is organized as follows:

- In Section 7.A.1, we give an overview of the proof that is common to all algorithms.
- In Section 7.A.2, we provide technical lemmas that are used in the detailed proofs of each algorithm.
- In Section 7.A.3, 7.A.4 and 7.A.5, we provide detailed analysis of MB-PSRL, MB-UCRL2, and MB-UCBVI.

7.A.1 Overview of the Proof

Let π^* be the optimal policy of the true MDP M and π^k the optimal policy for M^k, the sampled MDP at episode k. Recall that the expected regret is $\sum_{k=1}^{K} \mathbb{E}\left[\Delta^k\right]$, where $\Delta^k {=} W^{\pi^*}_{M,1:H^k}(\boldsymbol{s}_{t^k}) {-} W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k})$. For each of the three algorithms, we will define an event $\mathcal{E}^k_{\text{Algo}}$ that is o_{t^k}-measurable. $\mathcal{E}^k_{\text{Algo}}$ is true with high probability and guarantees that M and M^k are close. We have:

$$\begin{aligned}\mathbb{E}\left[\Delta^k\right] &= \mathbb{E}\left[\Delta^k \mathbb{I}_{\{\neg\mathcal{E}^k_{\text{Algo}}\}} + \Delta^k \mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right] \\ &\leq \mathbb{E}\left[H^k\right] \mathbb{P}\left(\neg\mathcal{E}^k_{\text{Algo}}\right) + \mathbb{E}\left[\Delta^k \mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right] \end{aligned} \tag{7.12}$$

because $\Delta^k \leq H^k$ and the random variables H^k and $\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}$ are independent. For each of the three algorithms, the policy π^k used at episode k is optimal for a model M^k, that is either sampled from the posterior distribution for MB-PSRL, or computed by extended value iteration for MB-UCRL2, or equal to the model with the bonus for MB-UCBVI. We have

$$\Delta^k = \underbrace{W^{\pi^*}_{M,1:H^k}(\boldsymbol{s}_{t^k}) - W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k})}_{:=\Delta^k_{model}} + \underbrace{W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k}) - W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k})}_{:=\Delta^k_{conc}}.$$

As we deal with the expected regret and H^k is independent of the model M^k and of the policy π^k, we have:

$$\mathbb{E}\left[\Delta^k_{model}\right] = v^{\pi^*}_{M}(\boldsymbol{s}_{t^k}) - v^{\pi^k}_{M^k}(\boldsymbol{s}_{t^k}) \tag{7.13}$$

As we see later, the above equation can be used to show that $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right]$ is either 0 (for MB-PSRL) or non-positive (for MB-UCRL2 or MB-UCBVI).

We are then left with $\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right]$. To do so, we use Lemma 7.6 to show that there exists a constant B^k (equal to H^k for MB-PSRL and MB-UCRL2, and $H^k L^k/(2(1-\gamma))$ for MB-UCBVI) such that

$$\begin{aligned}\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right] &= \mathbb{E}\left[\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\left(W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k}) {-} W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}_{t^k})\right)\right] \\ &\leq \mathbb{E}\left[\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}} \sum_{t=t^k}^{t^{k+1}-1} \left|r^k(s_{t,a_t}) {-} r(s_{t,a_t})\right| + B^k \left\|\boldsymbol{P}^k(s_{t,a_t},\cdot) {-} \boldsymbol{P}(s_{t,a_t},\cdot)\right\|_{\ell_1}\right]\end{aligned} \tag{7.14}$$

where $\left\|\boldsymbol{P}^k(s_a,\cdot)-\boldsymbol{P}(s_a,\cdot)\right\|_{\ell_1}=\sum_{s'_a}\left|P^k(s_a,s'_a)-P(s_a,s'_a)\right|$. For an arm a and a state $s_a\in\mathcal{S}_a$, we denote[1] by $N^k(s_a)=\sum_{t=1}^{t^k-1}\mathbb{I}_{\{s_{t,a_t}=s_a\}}$ the number of times that Arm a is activated before episode k while being in state s_a. Equation (7.14) relates the performance gap to the distance between the reward functions and transition matrices of the MDPs M and M^k. With $L^K=\sqrt{2\ln\frac{2SnK^2\ln K^2}{1-\gamma}}$, the event $\mathcal{E}^k_{\text{Algo}}$ guarantees that for all a, s_a and $k\geq 1$,

$$\left|r^k(s_a)-r(s_a)\right|\leq\frac{L^K}{\sqrt{\max\{1,N^k(s_a)\}}}\text{ and }\left\|\boldsymbol{P}^k(s_a,\cdot)-\boldsymbol{P}(s_a,\cdot)\right\|_{\ell_1}\leq\frac{2L^K+3\sqrt{S}}{\sqrt{\max\{1,N^k(s_a)\}}}\tag{7.15}$$

We use this with Equation (7.14) to show that:

$$\sum_{k=1}^{K}\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{Algo}}\}}\right]\leq\mathbb{E}\left[C^K_{\text{Algo}}\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{1}{\sqrt{\max\{1,N^k(s_{t,a_t})\}}}\right],\tag{7.16}$$

where C^K_{Algo} is a random variable that depends on the algorithm studied.

The final analysis takes care of the right term of Equation (7.16) and is more technical. It uses the fact that there cannot be too many large terms in this sum because if an arm is activated many times, then $1/\sqrt{N^k(s_{t,a_t})}$ is small. The main technical hurdle here is to deal with the K random episodes $H^1,\dots,H^K$. This is specific to our approach compared to the analysis of finite horizons. To bound this, one needs to bound terms of the form $\mathbb{E}\left[\max_{1\leq k\leq K}(H^k)^\alpha\right]$ with $\alpha\in\{1.5,2\}$ (see Equation (7.32)). To bound this, we use the geometric distribution of H^k to show that $\mathbb{E}\left[\max_{1\leq k\leq K}(H^k)^\alpha\right]=\mathcal{O}((\frac{\ln K}{1-\gamma})^\alpha)$ (see Lemma 7.8).

7.A.2 Technical lemmas common to the three algorithms

In this section, we establish a series of lemmas that are true for any learning algorithm used. They show that:

- The estimates $\hat{r}$ and $\hat{P}$ concentrates on their true values (Lemma 7.5);
- One can transform Δ^k_{conc} into Equation (7.14) (Lemma 7.6);
- The sum Equation (7.16) can be analyzed (Lemma 7.7).

[1]We use the notation $\mathbb{I}_{\{E\}}$ to denote a random variable that equals 1 if E is true and 0 otherwise. For instance, $\mathbb{I}_{\{Y_i=y\}}=1$ if $Y_i=y$ and 0 otherwise.

High Probability Events

Recall that o_{t^k} are the observations collected by the decision maker before episode k. Based on o_{t^k}, we compute the empirical estimators of reward vector and transition matrix as the following: For all $a \in [n]$ and any $s_a \in \mathcal{S}_a$, let $N^k(s_a) = \sum_{t=1}^{t^k-1} \mathbb{I}_{\{s_{t,a_t}=s_a\}}$ be the number of times so far that an arm a was activated in state s_a (at episode 1, we have $N^1(s_a) = 0$). Recall that $t^k{:=}1{+}\sum_{i=1}^{k-1} H^i$, and that $\hat{\boldsymbol{r}}^k$ and $\hat{\boldsymbol{P}}^k$ are the empirical mean reward vector and transition matrix. More precisely, $\hat{r}^k(s_a)$ is the empirical mean reward earned when arm a is chosen while being in state s_a:

$$\hat{r}^k(s_a) = \frac{1}{N^k(s_a)} \sum_{t=1}^{t^k-1} r_t \mathbb{I}_{\{a_t=a \wedge s_{t,a_t}=s_a\}},$$

and $\hat{P}^k(s_a, s'_a)$ is the fraction of times that arm a moved from s_a to s'_a:

$$\hat{P}^k(s_a, s'_a) = \frac{1}{N^k(s_a)} \sum_{t=1}^{t^k-1} \mathbb{I}_{\{a_t=a \wedge s_{t,a_t}=s_a \wedge s_{t+1,a_t}=s'_a\}}.$$

We design confidence sets similar to [JOA10; BT12].

Lemma 7.5 (High probability events based on concentration argument)
For any $k \le K$, let $L^k = \sqrt{2\ln\left(2SnKk\frac{\ln(Kk)}{1-\gamma}\right)}$. Let

$$\mathcal{E}_H^k := \left\{\forall k' \le k{:}H^{k'} \le \frac{\ln(Kk)}{1-\gamma}\right\} \tag{7.17}$$

$$\mathcal{E}_r^k := \left\{\forall a \in [n], s_a \in \mathcal{S}_a, k' \le k{:} \left|\hat{r}^{k'}(s_a){-}r(s_a)\right| \le \frac{L^k}{2\sqrt{\max\{1, N^{k'}(s_a)\}}}\right\} \tag{7.18}$$

$$\mathcal{E}_P^k := \left\{\forall a \in [n], s_a \in \mathcal{S}_a, k' \le k{:} \left\|\hat{\boldsymbol{P}}^{k'}(s_a, \cdot){-}\boldsymbol{P}(s_a, \cdot)\right\|_{\ell_1} \le \frac{L^k{+}1.5\sqrt{S}}{\sqrt{\max\{1, N^{k'}(s_a)\}}}\right\} \tag{7.19}$$

$$\mathcal{E}_v^k := \left\{\forall a \in [n], \boldsymbol{s} \in \mathcal{X}, k' \le k{:}\left|\hat{r}^{k'}(s_a){-}r(s_a) + \gamma \sum_{\boldsymbol{s}' \in \mathcal{X}} \left(\hat{p}^{k'}(\boldsymbol{s}' \mid \boldsymbol{s}, a){-}p(\boldsymbol{s}' \mid \boldsymbol{s}, a)\right) v_M^{\pi^*}(\boldsymbol{s}')\right| \le \frac{L^k}{2(1-\gamma)\sqrt{\max\{1, N^{k'}(s_a)\}}}\right\} \tag{7.20}$$

Then, the above events are all o_{t^k}-measurable. Moreover:

$$\mathbb{P}\left(\neg\mathcal{E}_H^k\right) \leq 1/K$$
$$\mathbb{P}\left(\neg\mathcal{E}_r^k\right) \leq 2/K$$
$$\mathbb{P}\left(\neg\mathcal{E}_P^k\right) \leq 2/K$$
$$\mathbb{P}\left(\neg\mathcal{E}_v^k\right) \leq 2/K.$$

Proof. For event $\mathcal{E}_H^k$, since $\{H^{k'}\}_{k'\leq k}$ are i.i.d. and geometrically distributed with parameter $(1-\gamma)$, we have that

$$\mathbb{P}\left(\exists k' \leq k : H^{k'} > \epsilon\right) \leq \sum_{k'=1}^{k} \mathbb{P}\left(H^{k'} > \epsilon\right) = k\gamma^{\lfloor \epsilon \rfloor}.$$

Then, with $\epsilon = \frac{\ln(1/(Kk))}{\ln(\gamma)}$, we get $\mathbb{P}\left(\exists k' \leq k : H^{k'} > \epsilon\right) \leq 1/K$. Moreover,

$$\epsilon = \frac{\ln(1/(Kk))}{\ln(\gamma)} = \frac{\ln(Kk)}{\ln(1/\gamma)} < \frac{\ln(Kk)}{1-\gamma}.$$

Then, $\mathbb{P}\left(\exists k' \leq k : H^{k'} > \frac{\ln(Kk)}{1-\gamma}\right) \leq 1/K$.

Let $\tau^k = k\frac{\ln(Kk)}{1-\gamma}$. Under event $\mathcal{E}_H^k$, the random variable t^k is upper bounded by the deterministic quantity τ^k. In what follows, we assume that event $\mathcal{E}_H^k$ holds.

For event $\mathcal{E}_r^k$, let $\tilde{r}_\ell(s_a)$ be a random variable that is the empirical mean of ℓ i.i.d. realizations of the reward when the arm in state s_a is chosen. In particular, $\hat{r}^k(s_a) = \tilde{r}_{N^k(s_a)}(s_a)$. By Hoeffding's inequality, for any $\epsilon > 0$, one has:

$$\mathbb{P}\left(|\tilde{r}_\ell(s_a) - r(s_a)| \geq \epsilon\right) \leq 2e^{-2\ell\epsilon^2}.$$

In particular, this holds for $\epsilon = \sqrt{\frac{\ln(2SnK\tau^k)}{2\ell}}$. As $N^k(s_a) < \tau^k$, by using the union-bound, this implies that:

$$\mathbb{P}\left(\mathcal{E}_H^k \wedge \exists a, s_a, k' \leq k : \left|\hat{r}^{k'}(s_a) - r(s_a)\right| \geq \sqrt{\frac{\ln(2SnK\tau^k)}{2N^{k'}(s_a)}}\right) \tag{7.21}$$
$$\leq \sum_a \sum_{s_a} \mathbb{P}\left(\exists \ell \in \{1, \dots, \tau^k - 1\} : |\tilde{r}_\ell(s_a) - r(s_a)| \geq \sqrt{\frac{\ln(2SnK\tau^k)}{2\ell}}\right)$$
$$\leq \sum_{\ell=1}^{\tau^k} \sum_a \sum_{s_a} \mathbb{P}\left(|\tilde{r}_\ell(s_a) - r(s_a)| \geq \sqrt{\frac{\ln(2SnK\tau^k)}{2\ell}}\right)$$
$$\leq nS \sum_{\ell=1}^{\tau^k} 2e^{-2\ell\frac{\ln(2SnK\tau^k)}{2\ell}} = 1/K,$$

where the second and third line is the union on all possible events $N^{k'}(s_a){=}\ell$ for all $\ell{\in}\{1, \dots, \tau^k - 1\}$. In total this says $\mathbb{P}\left(\mathcal{E}_H^k \wedge \neg\mathcal{E}_r^k\right) \leq 1/K$. Now, $\neg\mathcal{E}_r^k{=}(\mathcal{E}_H^k \wedge \neg\mathcal{E}_r^k) \vee (\neg\mathcal{E}_H^k \wedge \neg\mathcal{E}_r^k)$. Then, using union bound,

$$\mathbb{P}\left(\neg\mathcal{E}_r^k\right) \leq \mathbb{P}\left(\neg\mathcal{E}_r^k \wedge \mathcal{E}_H^k\right) + \mathbb{P}\left(\neg\mathcal{E}_r^k \wedge \neg\mathcal{E}_H^k\right)$$
$$\leq \mathbb{P}\left(\neg\mathcal{E}_r^k \wedge \mathcal{E}_H^k\right) + \mathbb{P}\left(\neg\mathcal{E}_H^k\right) \leq 2/K$$

The event $\mathcal{E}_P^k$ is similar but by using Weissman's inequality [Wei+03] instead of Hoeffding's bound. Indeed, by using Equation (8) in Theorem 2.1 of [Wei+03], if $N^k(s_a)$ was not a random variable, one would have

$$\mathbb{P}\left(\left\|\hat{\boldsymbol{P}}^k(s_a, \cdot){-}\boldsymbol{P}(s_a, \cdot)\right\|_{\ell_1} \geq \epsilon\right) \leq 2^S e^{-N^k(s_a)\epsilon^2/2}.$$

Following the same approach as for Equation (7.21) with $\epsilon = \sqrt{2\ln(SnK\tau^k 2^S)/N^k(s_a)}$, we use the union-bound to show that:

$$\mathbb{P}\left(\mathcal{E}_H^k \wedge \exists a, x_a, k' \leq k : \left\|\hat{\boldsymbol{P}}^{k'}(s_a, \cdot){-}\boldsymbol{P}(s_a, \cdot)\right\|_{\ell_1} \geq \sqrt{\frac{2\ln(SnK\tau^k 2^S)}{N^{k'}(s_a)}}\right)$$
$$\leq \tau^k nS2^S e^{-N^{k'}(s_a)\frac{2\ln(SnK\tau^k 2^S)}{2N^{k'}(s_a)}} = 1/K.$$

By definition of $L^k = \sqrt{2\ln(2SnK\tau^k)}$ and since $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$, we have

$$\sqrt{2\ln(SnK\tau^k 2^S)} = \sqrt{2\ln(2SnK\tau^k){+}2(S-1)\ln 2}$$
$$\leq L^k + \sqrt{2(S-1)\ln 2} \leq L^k + 1.5\sqrt{S}.$$

Hence:

$$\mathbb{P}\left(\mathcal{E}_H^k \wedge \exists a, s_a, k' \leq k : \left\|\hat{\boldsymbol{P}}^{k'}(s_a,\cdot) - \boldsymbol{P}(s_a,\cdot)\right\|_{\ell_1} \geq \frac{L^k + 1.5\sqrt{S}}{\sqrt{N^{k'}(s_a)}}\right) \leq 1/K.$$

As done for $\mathcal{E}_r^k$, we have $\neg\mathcal{E}_P^k{=}(\mathcal{E}_H^k \wedge \neg\mathcal{E}_P^k) \vee (\neg\mathcal{E}_H^k \wedge \neg\mathcal{E}_P^k)$. With the same process, we get $\mathbb{P}\left(\neg\mathcal{E}_P^k\right) \leq 2/K$.

For event $\mathcal{E}_v^k$, we have that $\hat{r}^k + \gamma\hat{p}^k v_M^{\pi^*}$ is the empirical mean of $r + \gamma p v_M^{\pi^*}$. This is because $v_M^{\pi^*}$ is deterministic and $\hat{r}^k$ and $\hat{p}^k$ are empirical mean of r and p respectively. Using Hoeffding's inequality and following the same approach above, we have $\mathbb{P}\left(\neg\mathcal{E}_v^k\right) \leq 2/K$. □

Note that Lemma 7.5 is about the statistical properties of the observations o_{t^k} in the observation space. These properties are true for any learning algorithms. In fact, we will combine different events of this lemma to bound the regret of our algorithm accordingly.

Concentration Gap

At episode k, our algorithms believe that the unknown MDP M is the MDP M^k. For Bayesian algorithms, M^k is sampled from posterior distribution while for optimistic algorithms, M^k is chosen with respect to optimism principle. The algorithms follow the policy π^k that is optimal for M^k. Recall that $W_{M,1:H^k}^{\pi^k}(\boldsymbol{s})$ is the expected reward of the MDP M under policy π^k, starts in state $\boldsymbol{s}$ and lasts for H^k time steps and the expected cumulative discounted reward in M starting from state $\boldsymbol{s}$ under policy π^k is $v_M^{\pi^k}(\boldsymbol{s}){=}\mathbb{E}[W_{M,1:H^k}^{\pi^k}(\boldsymbol{s})]$ where $H^k \sim \mathrm{Geom}(1-\gamma)$ is the horizon of episode k.

Lemma 7.6 (Gap of concentration)
For episode k, let $B^k \in \mathbb{R}^+$ be an upper bound[a] *of $W_{M^k,1:H^k}^{\pi^k}(\boldsymbol{s})$, i.e., a constant B^k such that for any $\boldsymbol{s} \in \mathcal{X}$, $W_{M^k,1:H^k}^{\pi^k}(\boldsymbol{s}) \leq B^k$. We have,*

$$\begin{aligned}\mathbb{E}\left[\Delta_{conc}^k | o_{t^k}, H^k, M^k, M\right] &{=}\mathbb{E}\left[W_{M^k,1:H^k}^{\pi^k}(\boldsymbol{s}_{t^k}){-}W_{M,1:H^k}^{\pi^k}(\boldsymbol{s}_{t^k}) | o_{t^k}, H^k, M^k, M\right] \\ &\leq\mathbb{E}\left[\sum_{t=t^k}^{t^{k+1}-1} \left|r^k(s_{t,a_t}){-}r(s_{t,a_t})\right| + B^k \left\|\boldsymbol{P}^k(s_{t,a_t},\cdot){-}\boldsymbol{P}(s_{t,a_t},\cdot)\right\|_{\ell_1} | o_{t^k}, H^k, M^k, M\right]\end{aligned} \tag{7.22}$$

[a] We will use $B^k = H^k$ for MB-PSRL and MB-UCRL2, and $B^k = H^k L^k/(2(1-\gamma))$ for MB-UCBVI.

Proof. From Equation (7.7) with $a = \pi^k(\boldsymbol{s})$,

$$W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}) = r(s_a) + \sum_{\boldsymbol{s}'} p(\boldsymbol{s}' \mid \boldsymbol{s}, a) W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}'). \tag{7.23}$$

Comparing the sampled MDP M^k with the original M and using Equation (7.23), one has

$$\begin{aligned} W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}) - W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}) &= r^k(s_a) - r(s_a) \\ &\quad + \sum_{\boldsymbol{s}'} p^k(\boldsymbol{s}' \mid \boldsymbol{s}, a) W^{\pi^k}_{M^k,2:H^k}(\boldsymbol{s}') - \sum_{\boldsymbol{s}'} p(\boldsymbol{s}' \mid \boldsymbol{s}, a) W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}'). \end{aligned} \tag{7.24}$$

Note that in the above equation, the last term is of the form $p^k W^{\pi^k}_{M^k} - p W^{\pi^k}_{M}$, which is equal to $(p^k - p) W^{\pi^k}_{M^k} + p(W^{\pi^k}_{M^k} - W^{\pi^k}_{M})$. Moreover, $W^{\pi_k}_{M_k}$ is less than B_k. Plugging this to the above equation shows that:

$$\begin{aligned} &W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}) - W^{\pi^k}_{M,1:H^k}(\boldsymbol{s}) \\ &\quad \le \left| r^k(s_a) - r(s_a) \right| + B^k \sum_{\boldsymbol{s}'} \left| p^k(\boldsymbol{s}' \mid \boldsymbol{s}, a) - p(\boldsymbol{s}' \mid \boldsymbol{s}, a) \right| \\ &\qquad + \sum_{\boldsymbol{s}'} p(\boldsymbol{s}' \mid \boldsymbol{s}, a) \Big(W^{\pi^k}_{M^k,2:H^k}(\boldsymbol{s}') - W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}') \Big) \\ &\quad = \left| r^k(s_a) - r(s_a) \right| + B^k \left\| \boldsymbol{p}^k(\cdot \mid \boldsymbol{s}, a) - \boldsymbol{p}(\cdot \mid \boldsymbol{s}, a) \right\|_{\ell_1} + D^{M^k,M}_{H^k}(\boldsymbol{s}) \\ &\qquad + W^{\pi^k}_{M^k,2:H^k}(\boldsymbol{s}_2) - W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}_2) \end{aligned}$$

where $D^{M^k,M}_{H^k}(\boldsymbol{s}) := \sum_{\boldsymbol{s}'} p(\boldsymbol{s}' \mid \boldsymbol{s}, a) \Big(W^{\pi^k}_{M^k,2:H^k}(\boldsymbol{s}') - W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}') \Big) - (W^{\pi^k}_{M^k,2:H^k}(\boldsymbol{s}_2) - W^{\pi^k}_{M,2:H^k}(\boldsymbol{s}_2))$. Note that in the equation above, $D^{M^k,M}_{H^k}(\boldsymbol{s})$ is a martingale difference term with $\boldsymbol{s}_2 \sim \boldsymbol{p}(\cdot \mid \boldsymbol{s}, a)$. Hence, the expected value of the martingale difference term is zero. As only arm a makes a transition, we have $\left\| \boldsymbol{p}^k(\cdot \mid \boldsymbol{s}, a) - \boldsymbol{p}(\cdot \mid \boldsymbol{s}, a) \right\|_{\ell_1} = \left\| \boldsymbol{P}^k(s_a, \cdot) - \boldsymbol{P}(s_a, \cdot) \right\|_{\ell_1}$. Hence, a direct induction shows that Equation (7.22) holds. □

Bound on the double sum

Recall that for $k \le K$, any $a \in [n]$ and any $s_a \in \mathcal{S}_a$, $N^k(s_a) = \sum_{t=1}^{t^k - 1} \mathbb{I}_{\{s_{t,a_t} = s_a\}}$ is the number of times so far that an arm a was activated in state s_a (at episode 1, we have $N^1(s_a) = 0$) and $\{H^k\}_{k \le K}$ be the sequence of episode horizons.

Lemma 7.7

For any learning algorithms, we have

$$\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{1}{\sqrt{\max\{1,N^k(s_{t,a_t})\}}}\le Sn\max_{k\le K}H^k+2\sqrt{SnK\max_{k\le K}H^k}$$

Proof. Let $\tilde{N}_t(s_a)$ be the number of times that arm a has been activated before time t while being in state s_a. By definition, $\tilde{N}_{t^k}(s_a)=N^k(s_a)$. Moreover, if $t\in\{t^k,\dots,t^{k+1}-1\}$, then $\tilde{N}_t(s_a)\le N^k(s_a)+H^k$. This shows that

$$\begin{aligned}\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{1}{\sqrt{\max\{1,N^k(s_{t,a_t})\}}}&\le\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{1}{\sqrt{\max\{1,\tilde{N}_t(s_{t,a_t})-H^k\}}}\\&\le\sum_{t=1}^{t^{K+1}-1}\frac{1}{\sqrt{\max\{1,\tilde{N}_t(s_{t,a_t})-\max_k H^k\}}}.\end{aligned}$$

The above sum can be reordered to group terms by state: The above sum equals

$$\begin{aligned}\sum_{a,s_a}\sum_{m=1}^{\tilde{N}_{t^{K+1}}(s_a)}\frac{1}{\sqrt{\max\{1,m-\max_k H^k\}}}&\le\sum_{a,s_a}\left[\max_k H^k+\sum_{m=1}^{\max\{1,\tilde{N}_{t^{K+1}}(s_a)-\max_k H^k\}}\frac{1}{\sqrt{m}}\right],\\&\le Sn\max_k H^k+\sum_{a,s_a}\sum_{m=1}^{\tilde{N}_{t^{K+1}}(s_a)}\frac{1}{\sqrt{m}},\\&\le Sn\max_k H^k+2\sum_{a,s_a}\sqrt{\tilde{N}_{t^{K+1}}(s_a)},\end{aligned}$$

where the last inequality holds because $\sum_{m=1}^{t^{K+1}}1/\sqrt{m}\le\int_1^{t^{K+1}}1/\sqrt{x}dx\le2\sqrt{t^{K+1}}$.

Now, by Cauchy-Schwartz inequality, and because $\sum_{a,s_a}\tilde{N}_{t^{K+1}}(s_a)=t^{K+1}-1=\sum_{k=1}^{K}H^k$, we have:

$$\sum_{a,s_a}\sqrt{\tilde{N}_{t^{K+1}}(s_a)}\le\Big(\sum_{a,s_a}\tilde{N}_{t^{K+1}}(s_a)\Big)^{1/2}\Big(\sum_{a,s_a}1\Big)^{1/2}=\sqrt{Sn\sum_{k=1}^{K}H^k}\le\sqrt{SnK\max_{k\le K}H^k}.$$

□

Bound on the expectation of $\mathbb{E}\left[\max_{k\le K} H^k\right]$

Lemma 7.8 (Bound on the expectation of the maximum random horizon)
Let $\alpha \in [1, 2.5]$. Then,

$$\mathbb{E}\left[\max_{k\le K}(H^k)^\alpha\right] \le 5 + 5\left(\frac{\ln K}{1-\gamma}\right)^\alpha. \tag{7.25}$$

Proof. By definition, we have

$$\begin{aligned}\mathbb{E}\left[\max_{k\le K}(H^k)^\alpha\right] &= \sum_{i=1}^{\infty}\mathbb{P}\left(\max_{k\le K}(H^k)^\alpha \ge i\right)\\ &\le \sum_{i=1}^{\infty}\min(1, K\mathbb{P}\left((H^k)^\alpha \ge i\right))\\ &= \sum_{i=1}^{\infty}\min(1, K\gamma^{i^{1/\alpha}}),\end{aligned}$$

where the inequality comes from the union bound, and the last equality is because the random variables H^k are geometrically distributed.

Let $A = \min\{i : K\gamma^{i^{1/\alpha}} \le 1\}$. Decomposing the above sum by group of size A, we have

$$\begin{aligned}\sum_{i=1}^{\infty}\min(1, K\gamma^{i^{1/\alpha}}) &= \sum_{j=0}^{\infty}\sum_{i=Aj+1}^{A(j+1)}\min(1, K\gamma^{i^{1/\alpha}})\\ &\le \sum_{j=0}^{\infty}A\min(1, K\gamma^{(Aj)^{1/\alpha}})\\ &= A + A\sum_{j=1}^{\infty}K(\gamma^{A^{1/\alpha}})^{j^{1/\alpha}},\end{aligned} \tag{7.26}$$

where the inequality holds because $\gamma^{i^{1/\alpha}}$ is decreasing in i.

By definition of A, we have $\gamma^{A^{1/\alpha}} \le 1/K$. This implies that the second term of Equation (7.26) is smaller than $\sum_{j=1}^{\infty}K(1/K)^{j^{1/\alpha}} = \sum_{j=1}^{\infty}K^{1-j^{1/\alpha}}$. As $\alpha \le 2.5$, if $K \ge 5$, this is smaller than $\sum_{j=1}^{\infty}5^{1-j^{1/2.5}} \approx 3.92 < 4$.

This shows that for $K \ge 5$, we have:

$$\mathbb{E}\left[\max_{k\le K}(H^k)^\alpha\right] \le 5A,$$

where $A = \lceil(-\ln K/\ln\gamma)^\alpha\rceil \le 1 + (\ln K/(1-\gamma))^\alpha$.

As for the case where $K \leq 4$, we have $\mathbb{E}\left[\max_{k\leq K}(H^k)^\alpha\right] \leq K\mathbb{E}\left[(H^1)^\alpha\right] \leq \frac{K}{(1-\gamma)^\alpha}$. This term is smaller than Equation (7.25) for $K \leq 4$. □

7.A.3 Detailed analysis of MB-PSRL

We decompose the analysis of PSRL in three steps:

- We define the high-probability event $\mathcal{E}^k_{\text{PSRL}}$.
- We analyze $\sum_{k=1}^K \mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right]$ (which equals 0 here because of posterior sampling).
- We analyze $\sum_{k=1}^K \mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right]$.

We will use the same proof structure for MB-UCRL2 and MB-UCBVI.

Before doing the proof, we start by a first lemma that essentially formalizes the fact that the distribution of M given o_{t^k} is the same as the distribution of the sampled MDP M^k conditioned on o_{t^k}.

Lemma 7.9 (Expectation identity)
Assume that the MDP M is drawn according to the prior ϕ and that M^k is drawn according to the posterior $\phi(\cdot \mid o_{t^k})$. Then, for any o_{t^k}-measurable function g, one has:

$$\mathbb{E}\left[g(M)\right] = \mathbb{E}\left[g(M^k)\right]. \tag{7.27}$$

Proof. At the start of each episode k, MB-PSRL computes the posterior distribution of M conditioned on the observations o_{t^k}, and draws M^k from it. This implies that M and M^k are identically distributed conditioned on o_{t^k}. Consequently, if g is an o_{t^k}-measurable function, one has:

$$\mathbb{E}\left[g(M) \mid o_{t^k}\right] = \mathbb{E}\left[g(M^k) \mid o_{t^k}\right].$$

Equation (7.27) then follows from the tower rule. □

Definition of the high probability event $\mathcal{E}^k_{\text{PSRL}}$

Lemma 7.10 (High probability event for MB-PSRL)
At episode k, the event

$$\mathcal{E}^k_{\text{PSRL}} = \Bigg\{\forall a \in [n], s_a \in \mathcal{S}_a, k' \leq k\colon \left|r^k(s_a) - r(s_a)\right| \leq \frac{L^k}{\sqrt{\max\{1, N^{k'}(s_a)\}}},$$
$$\left\|\boldsymbol{P}^k(s_a,\cdot) - \boldsymbol{P}(s_a,\cdot)\right\|_{\ell_1} \leq \frac{2L^k + 3\sqrt{S}}{\sqrt{\max\{1, N^{k'}(s_a)\}}}, \text{ and } H^{k'} \leq \frac{\ln(Kk)}{1-\gamma}\Bigg\}$$

is o_{t^k}-measurable and true with probability at least $1 - 9/K$.

Proof. Recall that for MB-PSRL, at the beginning of episode k, we sample an MDP M^k. We define the two events that are the analogue of the events Equation (7.18) and Equation (7.19) of Lemma 7.5 but replacing the true MDP M by the sampled MDP M^k:

$$\tilde{\mathcal{E}}^k_r := \left\{\forall a \in [n], s_a \in \mathcal{S}_a, k' \leq k\colon \left|\hat{r}^{k'}(s_a) - r^k(s_a)\right| \leq \frac{L^k}{2\sqrt{\max\{1, N^{k'}(s_a)\}}}\right\}$$
$$\tilde{\mathcal{E}}^k_P := \left\{\forall a \in [n], s_a \in \mathcal{S}_a, k' \leq k\colon \left\|\hat{\boldsymbol{P}}^{k'}(s_a,\cdot) - \boldsymbol{P}^k(s_a,\cdot)\right\|_{\ell_1} \leq \frac{L^k + 1.5\sqrt{S}}{\sqrt{\max\{1, N^{k'}(s_a)\}}}\right\}$$

These events are o_{t^k}-measurable. Hence, Lemma 7.9, combined with Lemma 7.5 implies that $\mathbb{P}\left(\neg\tilde{\mathcal{E}}^k_r\right) = \mathbb{P}\left(\neg\mathcal{E}^k_r\right) \leq 2/K$ and $\mathbb{P}\left(\neg\tilde{\mathcal{E}}^k_P\right) = \mathbb{P}\left(\neg\mathcal{E}^k_P\right) \leq 2/K$. Since the complement of $\mathcal{E}^k_{\text{PSRL}}$ is the union of $\neg\mathcal{E}^k_r, \neg\tilde{\mathcal{E}}^k_r, \neg\mathcal{E}^k_P, \neg\tilde{\mathcal{E}}^k_P$ and $\neg\mathcal{E}^k_H$, the union bound implies that $\mathbb{P}\left(\mathcal{E}^k_{\text{PSRL}}\right) \geq 1 - 9/K$. □

Analysis of $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right]$ for MB-PSRL

Lemma 7.9 implies that for MB-PSRL, $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right] = 0$ because $\mathcal{E}^k_{\text{PSRL}}$, π^k and M^k are o_{t^k}-measurable.

Analysis of $\mathbb{E}\left[\Delta^k_{conc}\right]$ for MB-PSRL

Following Equation (7.12), the Bayesian regret can be written as:

$$\begin{aligned}\mathrm{BayesRegret}(\text{MB-PSRL}, \phi, K) = \sum_{k=1}^{K} \mathbb{E}\left[\Delta^k\right] &\leq \sum_{k=1}^{K} \mathbb{E}\left[H^k\right] \mathbb{P}\left(\neg\mathcal{E}^k_{\text{PSRL}}\right) + \mathbb{E}\left[\Delta^k \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right] \\ &\leq \frac{9}{(1-\gamma)} + \sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{model} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right] + \mathbb{E}\left[\Delta^k_{conc} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right]\end{aligned} \tag{7.28}$$

where the last inequality holds due to Lemma 7.10. By the previous section, the second term of Equation (7.28) is zero. As all rewards are bounded by 1, $W^{\pi^k}_{M^k,1:H^k}(\boldsymbol{s}_{t^k}) \leq H^k$. Hence, by applying Lemma 7.6 with the upper bound $B^k = H^k$, and because $\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}$ is deterministic given o_{t^k}, we have

$$\begin{aligned}\mathbb{E}\left[\Delta^k_{conc} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right] &= \mathbb{E}\left[\mathbb{E}\left[\Delta^k_{conc} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}} \mid o_{t^k}, H^k, M^k, M\right]\right] \\ &\leq \mathbb{E}\Bigg[\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}} \sum_{t=t^k}^{t^{k+1}-1} \left|r^k(s_{t,a_t}) - r(s_{t,a_t})\right| \\ &\qquad + H^k \left\|\boldsymbol{P}^k(s_{t,a_t}, \cdot) - \boldsymbol{P}(s_{t,a_t}, \cdot)\right\|_{\ell_1}\Bigg].\end{aligned} \tag{7.29}$$

Let $\mathcal{R}^k := \sum_{t=t^k}^{t^{k+1}-1} \left|r^k(s_{t,a_t}) - r(s_{t,a_t})\right| + H^k \left\|\boldsymbol{P}^k(s_{t,a_t}, \cdot) - \boldsymbol{P}(s_{t,a_t}, \cdot)\right\|_{\ell_1}$. By using the definition of $\mathcal{E}^k_{\text{PSRL}}$, we have:

$$\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}} \mathcal{R}^k \leq \sum_{t=t^k}^{t^{k+1}-1} \frac{L^k + (2L^k + 3\sqrt{S})H^k}{\sqrt{\max\{1, N^k(s_{t,a_t})\}}} \tag{7.30}$$

Hence, summing over all K episodes gives us:

$$\begin{aligned}\sum_{k=1}^{K} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}} \mathcal{R}^k &\leq (L^K + (2L^K + 3\sqrt{S}) \max_{k \leq K} H^k) \sum_{k=1}^{K} \sum_{t=t^k}^{t^{k+1}-1} \frac{1}{\sqrt{\max\{1, N^k(s_{t,a_t})\}}} \\ &\leq 3(L^K + \sqrt{S}) \max_{k \leq K} H^k \sum_{k=1}^{K} \sum_{t=t^k}^{t^{k+1}-1} \frac{1}{\sqrt{\max\{1, N^k(s_{t,a_t})\}}},\end{aligned} \tag{7.31}$$

where the first inequality holds because $L^k \leq L^K$ and $\max_{k \leq K} H^k \geq 1$.

Note that the last inequality leads to a slightly worst bound but simplifies the expression. By Lemma 7.7, we get

$$\begin{aligned}\sum_{k=1}^{K} \mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}} \mathcal{R}^k &\le 3(L^K + \sqrt{S}) \max_{k\le K} H^k (Sn \max_{k\le K} H^k + 2\sqrt{SnK \max_{k\le K} H^k}) \\ &= 3(L^K + \sqrt{S})(Sn \max_{k\le K}(H^k)^2 + 2\sqrt{SnK} \max_{k\le K}(H^k)^{3/2})\end{aligned}$$

Then,

$$\begin{aligned}\sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{PSRL}}\}}\right] &\le 3(L^K+\sqrt{S})\Big(Sn\mathbb{E}\left[\max_{k\le K}(H^k)^2\right] + 2\sqrt{SnK}\mathbb{E}\left[\max_{k\le K}(H^k)^{3/2}\right]\Big) \qquad (7.32)\\ &\le 3(L^K+\sqrt{S})\Big(Sn\Big(5+5\Big(\frac{\ln K}{1-\gamma}\Big)\Big)^2 + \sqrt{SnK}\Big(5+5\Big(\frac{\ln K}{1-\gamma}\Big)\Big)^{3/2}\Big)\end{aligned}$$

where the last inequality is true due to Lemma 7.8. With $L^K = \sqrt{2\ln\Big(2SnK^2\frac{\ln K^2}{1-\gamma}\Big)}$, this implies that there exists a constant C independent of all problem's parameters such that:

$$\text{BayesRegret}(\text{MB-PSRL}, \phi, K) \le C\Big(\sqrt{S} + \ln\frac{SnK\ln K}{1-\gamma}\Big)\Big(Sn\Big(\frac{\ln K}{1-\gamma}\Big)^2 + \sqrt{SnK}\Big(\frac{\ln K}{1-\gamma}\Big)^{3/2}\Big).$$

Remark on the dependence on S

Our bound is linear in S, the state size of each arm, because our proof follows the approach used in [ORV13]. Using another proof methodology, it is argued in [OV17] that the regret of PSRL grows as the square root of the state space size and not linearly. In our paper, we choose to use the more conservative approach of [ORV13] because we believe that the proof used in [OV17] is not correct (in particular the use of a deterministic v in Equation (16) of the proof of Lemma 3 in Appendix A in the arXiv version of [OV17] seems incompatible with the use of Lemma 4 of the same paper). In fact, when considering the worst case realization of v, the concentration bound in Equation (16) of the paper is equivalent to the (scaled) L1 norm of transition concentration. We are not alone to point out this error. Effectively, [AJ17] used Lemma C.1 and Lemma C.3 (equivalence of Lemma 3 of [OV17]) to get a bound in square root of the state space size. But both lemmas are erroneous as mentioned in the latest arXiv version of [AJ17]. The validity of Lemma 3 is also questioned on page 87 of [Fru19]. While it is informal, the recent work of [Qia+20] also theoretically contradicts the lemma.

7.A.4 Case of MB-UCRL2

The proof follows the same steps as for MB-PSRL. While the high probability event is simpler, the additional complication is to show that $\sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{model}\right] \leq 0$ by using the optimism principle.

Definition of the high probability event

Lemma 7.11 (High probability event for MB-UCRL2)
At episode k, the event

$$\mathcal{E}^k_{\text{UCRL2}} = \Bigg\{ \forall a \in [n], s_a \in \mathcal{S}_a, k' \leq k \colon \left|\hat{r}^{k'}(s_a) - r(s_a)\right| \leq \frac{L^k}{2\sqrt{\max\{1, N^{k'}(s_a)\}}}, \\ \left\|\hat{\boldsymbol{P}}^{k'}(s_a, \cdot) - \boldsymbol{P}(s_a, \cdot)\right\|_{\ell_1} \leq \frac{L^k + 1.5\sqrt{S}}{\sqrt{\max\{1, N^{k'}(s_a)\}}}, \text{ and } H^{k'} \leq \frac{\ln(Kk)}{1-\gamma} \Bigg\}$$

is o_{t^k}-measurable and true with probability at least $1 - 5/K$.

Proof. The complement of $\mathcal{E}^k_{\text{UCRL2}}$ is the union of $\neg\mathcal{E}^k_r, \neg\mathcal{E}^k_P$ and $\neg\mathcal{E}^k_H$. We conclude the proof by using the union bound and $\mathbb{P}\left(\neg\mathcal{E}^k_r\right) \leq 2/K$, $\mathbb{P}\left(\neg\mathcal{E}^k_P\right) \leq 2/K$ and $\mathbb{P}\left(\neg\mathcal{E}^k_H\right) \leq 1/K$. $\square$

Analysis of $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right]$ – Optimism of MB-UCRL2

Recall that π^* is the optimal policy of the unknown MDP M and that π^k is the policy used in episode k. π^k is optimal for the optimistic MDP that is chosen from the plausible MDP set $\mathbb{M}^k$:

$$\pi^k \in \arg\max_{\pi} \max_{M' \in \mathbb{M}^k} v^{\pi}_{M'}.$$

For each episode k, the plausible MDP set $\mathbb{M}^k$ is defined by

$$\mathbb{M}^k = \Bigg\{ (\boldsymbol{r}', \boldsymbol{P}') : \forall a, s_a, \left|r'(x_a) - \hat{r}^k(s_a)\right| \leq \frac{L^k}{2\sqrt{\max\{1, N^k(s_a)\}}}, \text{ and} \\ \left\|\boldsymbol{P}'(s_a, \cdot) - \hat{\boldsymbol{P}}^k(s_a, .)\right\|_{\ell_1} \leq \frac{L^k + 1.5\sqrt{S}}{\sqrt{\max\{1, N^k(s_a)\}}} \Bigg\}. \tag{7.33}$$

As [JOA10], we argue that there exists an MDP $M^k \in \mathbb{M}^k$ such that π^k is an optimal policy for M^k. Moreover, under event $\mathcal{E}^k_{\text{UCRL2}}$, one has $M \in \mathbb{M}^k$, which implies that $\max_\pi \max_{M' \in \mathbb{M}^k} v^\pi_{M'}(\boldsymbol{s}) \geq v^{\pi^*}_M(\boldsymbol{s})$. By Equation (7.13), we get $\mathbb{E}\left[\Delta^k_{model}\right] \leq 0$. If $\mathcal{E}^k_{\text{UCRL2}}$ does not hold, we simply have $\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}} = 0$. We conclude that: $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right] \leq 0$.

Analysis of $\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right]$ for MB-UCRL2

Following Equation (7.12), the expected regret can be written as:

$$\begin{aligned}\mathbb{E}\left[\text{Regret}(\text{MB-UCRL2}, M, K)\right] &= \sum_{k=1}^{K} \mathbb{E}\left[\Delta^k\right] \leq \sum_{k=1}^{K} \mathbb{E}\left[H^k\right]\mathbb{P}\left(\neg\mathcal{E}^k_{\text{UCRL2}}\right) + \mathbb{E}\left[\Delta^k \mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right] \\ &\leq \frac{5}{1-\gamma} + \sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\neg\mathcal{E}^k_{\text{UCRL2}}\}}\right] + \mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\neg\mathcal{E}^k_{\text{UCRL2}}\}}\right]\end{aligned} \tag{7.34}$$

where the last inequality holds due to Lemma 7.11. By the previous section, the second term of Equation (7.34) is non-positive. In the following, we therefore analyze the last term whose analysis is then similar to the one for MB-PSRL. Indeed, with $B^k = H^k$ and definition of $\mathcal{E}^k_{\text{UCRL2}}$, the use of Lemma 7.6 shows that one has

$$\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right] \leq \mathbb{E}\left[\frac{1}{2}\sum_{t=t^k}^{t^{k+1}-1} \frac{L^k + (2L^k + 3\sqrt{S})H^k}{\sqrt{\max\{1, N^k(s_{t,a_t})\}}}\right].$$

Up to a factor $1/2$, the expression inside the expectation is the same as Equation (7.30) of MB-PSRL. Hence, one can use Lemma 7.7 the same way to show that

$$\sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCRL2}}\}}\right] \leq \frac{3}{2}(L^K + \sqrt{S})\left(Sn\mathbb{E}\left[\max_{k\leq K}(H^k)^2\right] + 2\sqrt{SnK}\mathbb{E}\left[\max_{k\leq K}(H^k)^{3/2}\right]\right).$$

Up to a factor $1/2$, the right term of the above equation is equal to the right term of Equation (7.32). Following the same process done for the latter, we can conclude that there exists a constant C' independent of all problem's parameters such that:

$$\text{Regret}(\text{MB-UCRL2}, M, K) \leq C'\left(\sqrt{S} + \ln\frac{SnK\ln K}{1-\gamma}\right)\left(Sn\left(\frac{\ln K}{1-\gamma}\right)^2 + \sqrt{SnK}\left(\frac{\ln K}{1-\gamma}\right)^{3/2}\right).$$

7.A.5 Case of MB-UCBVI

We start by defining the high probability event. Then, we prove the optimistic property of MB-UCBVI. Finally, we bound its expected regret.

Definition of the high-probability event

Lemma 7.12 (High probability event for MB-UCBVI)
The event

$$\mathcal{E}^k_{\text{UCBVI}}=\Bigg\{\forall a\in[n], \boldsymbol{s}\in\mathcal{X}, k'\leq k:$$

$$\left|\hat{r}^{k'}(s_a)-r(s_a)\right| \leq \frac{L^k}{2\sqrt{\max\{1, N^{k'}(s_a)\}}},$$

$$\left\|\hat{\boldsymbol{P}}^{k'}(s_a,\cdot)-\boldsymbol{P}(s_a,\cdot)\right\|_{\ell_1} \leq \frac{L^k+1.5\sqrt{S}}{\sqrt{\max\{1, N^{k'}(s_a)\}}}, H^{k'} \leq \frac{\ln(Kk)}{1-\gamma}, \text{ and}$$

$$\left|\hat{r}^{k'}(s_a)-r(s_a)+\gamma\sum_{\boldsymbol{s}'}(\hat{p}^{k'}(\boldsymbol{s}'\mid\boldsymbol{s},a)-p(\boldsymbol{s}'\mid\boldsymbol{s},a))v^{\pi^*}_M(\boldsymbol{s}')\right| \leq \frac{L^k}{2(1-\gamma)\sqrt{\max\{1, N^{k'}(s_a)\}}}$$

$$\Bigg\}$$

is o_{t^k}*-measurable and true with probability at least* $1-7/K$.

Proof. The complement of $\mathcal{E}^k_{\text{UCBVI}}$ is the union of $\neg\mathcal{E}^k_r, \neg\mathcal{E}^k_P, \neg\mathcal{E}^k_H$ and $\neg\mathcal{E}^k_v$. We conclude the proof by using the union bound and $\mathbb{P}\left(\neg\mathcal{E}^k_r\right)\leq 2/K$, $\mathbb{P}\left(\neg\mathcal{E}^k_P\right)\leq 2/K$, $\mathbb{P}\left(\neg\mathcal{E}^k_H\right)\leq 1/K$, and $\mathbb{P}\left(\neg\mathcal{E}^k_v\right)\leq 2/K$. □

Analysis of $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right]$ – Optimism of MB-UCBVI

The following lemma guarantees that $\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right]\leq 0$. Indeed, as $\mathcal{E}^k_{\text{UCBVI}}$ is o_{t^k}-measurable, one has

$$\begin{aligned}\mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right] &= \mathbb{E}\left[\mathbb{E}\left[\Delta^k_{model}\mid o_{t^k}\right]\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right] \\ &= \mathbb{E}\left[(v^{\pi^*}_M(\boldsymbol{s}_{t^k})-v^{\pi^k}_{M^k}(\boldsymbol{s}_{t^k}))\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right]\leq 0.\end{aligned}$$

Lemma 7.13 (Optimism of MB-UCBVI)
If $\mathcal{E}^k_{\text{UCBVI}}$ holds, then for any $\boldsymbol{s} \in \mathcal{X}$, we have

$$v^{\pi^k}_{M^k}(\boldsymbol{s}) \geq v^{\pi^*}_{M}(\boldsymbol{s})$$

Proof. Recall that at episode k, we define the optimistic MDP of MB-UCBVI by M^k in which the parameters of any arm $a \in [n]$ are $(\hat{\boldsymbol{r}}^k_a + \beta^k, \hat{\boldsymbol{P}}^k_a)$ with $\beta^k(s_a) = \frac{L^k}{2(1-\gamma)\sqrt{\max\{1, N^k(s_a)\}}}$ for any $s_a \in \mathcal{S}_a$. The Gittins index policy π^k is optimal for MDP M^k. For any state $\boldsymbol{s}$, let $a = \pi^k(\boldsymbol{s})$ and $a^* = \pi^*(\boldsymbol{s})$. Then,

$$\begin{aligned}
v^{\pi^k}_{M^k}(\boldsymbol{s}) - v^{\pi^*}_{M}(\boldsymbol{s}) &= \beta^k(s_a) + \hat{r}^k(s_a) + \gamma \sum_{\boldsymbol{s}'} \hat{p}^k(\boldsymbol{s}' \mid \boldsymbol{s}, a) v^{\pi^k}_{M^k}(\boldsymbol{s}') - v^{\pi^*}_{M}(\boldsymbol{s}) \\
&\geq \beta^k(s_{a^*}) + \hat{r}^k(s_{a^*}) + \gamma \sum_{\boldsymbol{s}'} \hat{p}^k(\boldsymbol{s}' \mid \boldsymbol{s}, a^*) v^{\pi^k}_{M^k}(\boldsymbol{s}') \\
&\quad - r(s_{a^*}) - \gamma \sum_{\boldsymbol{s}'} p(\boldsymbol{s}' \mid \boldsymbol{s}, a^*) v^{\pi^*}_{M}(\boldsymbol{s}') \\
&= \beta^k(s_{a^*}) + \hat{r}^k(s_{a^*}) - r(s_{a^*}) + \gamma \sum_{\boldsymbol{s}'} \Big(\hat{p}^k(\boldsymbol{s}' \mid \boldsymbol{s}, a^*) - p(\boldsymbol{s}' \mid \boldsymbol{s}, a^*) \Big) v^{\pi^*}_{M}(\boldsymbol{s}') \\
&\qquad + \gamma \sum_{\boldsymbol{s}'} \hat{p}^k(\boldsymbol{s}' \mid \boldsymbol{s}, a^*) \Big(v^{\pi^k}_{M^k}(\boldsymbol{s}') - v^{\pi^*}_{M}(\boldsymbol{s}') \Big)
\end{aligned}$$

Let (r^π, P^π) be the reward vector and transition matrix under policy π (i.e. $\forall \boldsymbol{s}, \boldsymbol{s}' \in \mathcal{X}, r^\pi(\boldsymbol{s}) = r(s_{\pi(\boldsymbol{s})}), P^\pi(\boldsymbol{s}, \boldsymbol{s}') = p\Big(\boldsymbol{s}' \mid \boldsymbol{s}, \pi(\boldsymbol{s})\Big)$ as defined in Equation (7.1)). In matrix form, the above equations

$$v^{\pi^k}_{M^k} - v^{\pi^*}_{M} \geq (\beta^k)^{\pi^*} + (\hat{r}^k)^{\pi^*} - r^{\pi^*} + \gamma \Big((\hat{P}^k)^{\pi^*} - P^{\pi^*} \Big) v^{\pi^*}_{M} + \gamma (\hat{P}^k)^{\pi^*} (v^{\pi^k}_{M^k} - v^{\pi^*}_{M}).$$

Under event $\mathcal{E}^k_{\text{UCBVI}}$, $(\beta^k)^{\pi^*} + (\hat{r}^k)^{\pi^*} - r^{\pi^*} + \gamma \Big((\hat{P}^k)^{\pi^*} - P^{\pi^*} \Big) v^{\pi^*}_{M} \geq 0$. This implies that:

$$\Big(I - \gamma (\hat{P}^k)^{\pi^*} \Big) (v^{\pi^k}_{M^k} - v^{\pi^*}_{M}) \geq 0.$$

As $\Big(I - \gamma (\hat{P}^k)^{\pi^*} \Big)^{-1} = I + \Big(I - \gamma (\hat{P}^k)^{\pi^*} \Big) + \Big(I - \gamma (\hat{P}^k)^{\pi^*} \Big)^2 + \ldots$ is a matrix whose coefficients are all non-negative, this implies that $v^{\pi^k}_{M^k} - v^{\pi^*}_{M} \geq 0$. $\square$

Analysis of $\mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right]$ for MB-UCBVI

Following Equation (7.12), the expected regret can be written similarly to Equation (7.34) for MB-UCRL2, one can write that

$$\mathbb{E}\left[\text{Regret}(\text{MB-UCBVI}, M, K)\right] \leq \frac{7}{1-\gamma} + \sum_{k=1}^{K} \mathbb{E}\left[\Delta^k_{model}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right] + \mathbb{E}\left[\Delta^k_{conc}\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\right].$$

The same as MB-UCRL2, the second term is non-positive. We are therefore left with the last term. Using Lemma 7.6 with $B^k = \frac{H^kL^k}{2(1-\gamma)}$ and the definition of M^k for MB-UCBVI, we have:

$$\begin{aligned}
\sum_{k=1}^{K}\mathbb{E}\left[\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\Delta^k_{\text{conc}}\right] &\leq \sum_{k=1}^{K}\mathbb{E}\Bigg[\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\sum_{t=t^k}^{t^{k+1}-1}\beta^k(s_{t,a_t}) + \left|\hat{r}^k(s_{t,a_t}) - r(s_{t,a_t})\right| \\
&\qquad + \frac{H^kL^k}{2(1-\gamma)}\left\|\hat{\boldsymbol{P}}^k(s_{t,a_t},\cdot) - \boldsymbol{P}(s_{t,a_t},\cdot)\right\|_{\ell_1}\Bigg] \\
&\leq \mathbb{E}\left[\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{(2-\gamma)L^k + H^kL^k(L^k + 1.5\sqrt{S})}{2(1-\gamma)\sqrt{\max\{1, N^k(s_{t,a_t})\}}}\right] \\
&\leq \mathbb{E}\left[\frac{2L^K(L^K + \sqrt{S})\max_{k\leq K} H^k}{1-\gamma}\sum_{k=1}^{K}\sum_{t=t^k}^{t^{k+1}-1}\frac{1}{\sqrt{\max\{1, N^k(s_{t,a_t})\}}}\right] \\
&\leq \mathbb{E}\left[\frac{2L^K(L^K + \sqrt{S})\max_{k\leq K} H^k}{1-\gamma}\left(Sn\max_{k\leq K} H^k + 2\sqrt{SnK\max_{k\leq K} H^k}\right)\right]
\end{aligned}$$

where the second inequality holds due to the definition of $\mathcal{E}^k_{\text{UCBVI}}$ and the last one holds due to Lemma 7.7. With $L^K = \sqrt{2\ln\left(\frac{2SnK^2\ln K^2}{1-\gamma}\right)}$, we have

$$\begin{aligned}
\sum_{k=1}^{K}\mathbb{E}\left[\mathbb{I}_{\{\mathcal{E}^k_{\text{UCBVI}}\}}\Delta^k_{conc}\right] \leq \frac{4(1+\sqrt{S})}{1-\gamma}\ln\Big(\frac{4SnK^2\ln K}{1-\gamma}\Big)\Big(Sn\mathbb{E}\left[\max_{k\leq K}(H^k)^2\right] \\
+2\sqrt{SnK}\mathbb{E}\left[\max_{k\leq K}(H^k)^{3/2}\right]\Big)
\end{aligned}$$

The last term of the right side above can be analyzed exactly the same as what is done for Equation (7.32) using Lemma 7.8. This concludes the proof.

Remark on the dependence on S

In the analysis of UCBVI and in our analysis, one need to bound the last term of Equation (7.24) which is of the form $\boldsymbol{p}^kW^{\pi^k}_{M^k} - \boldsymbol{p}W^{\pi^k}_{M}$. [AOM17] rewrite this term as $(\boldsymbol{p}^k - \boldsymbol{p})W^{\pi^*}_{M} + (\boldsymbol{p}^k - \boldsymbol{p})(W^{\pi^k}_{M^k} - W^{\pi^*}_{M}) + \boldsymbol{p}(W^{\pi^k}_{M^k} - W^{\pi^k}_{M})$. They then bound the first term by Chernoff-Hoeffding's inequality because the optimal value of the unknown MDP is deterministic. The second term, called the "correction" term, is bounded by using Bernstein's inequality and the optimism [AOM17, Step 1, page 6]. The third term is bounded by using the inequalities for martingale difference sequences. This allows the authors to have a factor $\sqrt{S}$ (instead of the classical S) in the regret bound.

If the technique to deal with the first and third terms would also apply to our case, the analysis of the second term uses heavily the optimism and the fact that $W_{M^k}^{\pi^k} \geq W_M^{\pi^*}$ in their setting. This cannot be adapted for MB-UCBVI because the optimism implies that $v_{M^k}^{\pi^k} \geq v_M^{\pi^*}$ but we do not necessarily have $W_{M^k,h:H^k}^{\pi^k}(\boldsymbol{s}') \geq W_{M,h:H^k}^{\pi^*}(\boldsymbol{s}')$ for all $\boldsymbol{s}' \in \mathcal{X}$, all $h \in [H^k]$ and all k. This is because we might lose the optimism when working with the value function over random episode length. That is why we need to use Lemma 7.6 which is more conservative.

7.B Proof of Theorem 7.3

To prove the lower bound, we consider a specific Markovian bandit problem that is composed of S independent *stochastic bandit problems*. This allows us to reuse the existing minimax lower bound for stochastic bandit problems. This existing result can be stated as follows: let $\mathcal{L}^{\text{stoc.pb}}$ be a learning algorithm for the stochastic bandit problem. It is shown in Theorem 3.1 of [BC12] that for any number of arms n and any number of time steps τ, there exists parameters for a stochastic bandit problem $M^{\text{stoc.pb}}$ with n arms such that the regret of the learning algorithm over τ time steps is at least $(1/20)\sqrt{n\tau}$.

$$\text{Regret}^{\text{stoc.pb}}(\tau, \mathcal{L}^{\text{stoc.pb}}, M^{\text{stoc.pb}}) \geq \frac{1}{20}\sqrt{n\tau}. \tag{7.35}$$

This lower bound (Theorem 3.1 of [BC12]) is constructed by considering n stochastic bandit problems $M^{\text{stoc.pb},j}$ for $j \in [n]$ with parameters that depend on τ and n. In the problem $M^{\text{stoc.pb},j}$, all arms have a reward $\theta(\tau, n)$ except arm j that has a reward $\theta'(\tau, n) > \theta(\tau, n)$. It is shown in Theorem 3.1 of [BC12] that a learning algorithm cannot perform uniformly well on all problems because it is impossible to distinguish them *a priori*. More precisely, in the proof of Lemma 3.2 of [BC12], it is shown that if the best arm is chosen at random, then the expected (Bayesian) regret of any learning algorithm is at least $(1/20)\sqrt{n\tau}$.

As for our problem, let K be a number of episodes, γ a discount factor, n a number of arms, S a number of states per arm and set $\tau = K/(2S(1-\gamma))$. We consider a random Markovian bandit model M constructed as follows. Each arm a has S states with the state space $\mathcal{S}_a = \{1_a, 2_a, \dots, S_a\}$. The transition matrix $\boldsymbol{P}_a$ is the identity matrix. For each state $i \in \{1, \dots, S\}$, we choose the best arm a_i^* uniformly at random among the n arms, independently for each i. The rewards of a state i_a are *i.i.d.* Bernoulli rewards with mean $\theta(\tau, n)$ if $a \neq a_i^*$ and $\theta'(\tau, n)$ if $a = a_i^*$. The initial distribution ρ couples the initial states of all arms for all $i \in \{1, \dots, S\}$,

$$\mathbb{P}\left(\forall a \in [n] : s_{1,a} = i_a\right) = \frac{1}{S}.$$

In this case, the Markovian bandit problem becomes a combination of S independent stochastic bandit problems with n arms each. We denote by $M_i^{\text{stoc.pb}}$ the random stochastic bandit problem for the initial state $\boldsymbol{i} = (i_a)_{a\in[n]}$. As the best arm a_i^* are chosen independently for each i, a learning algorithm $\mathcal{L}$ cannot use the information from $M_i^{\text{stoc.pb}}$ to perform better on $M_j^{\text{stoc.pb}}$, $j \neq i$.

Let ϕ be the distribution of the random Markovian bandit model M defined above and let T_i be the number of time steps spent in state $\boldsymbol{i}$ by the learning algorithm $\mathcal{L}$.

$$\begin{aligned}
\text{BayesRegret}(\mathcal{L}, \phi, K) &\geq \sum_{i=1}^{S} \mathbb{E}\left[\text{Regret}^{\text{stoc.pb}}(T_i, \mathcal{L}_i^{\text{stoc.pb}}, M_i^{\text{stoc.pb}})\right] \\
&\geq \sum_{i=1}^{S} \mathbb{E}\left[\text{Regret}^{\text{stoc.pb}}(\tau, \mathcal{L}_i^{\text{stoc.pb}}, M_i^{\text{stoc.pb}})\mathbb{I}_{\{T_i \geq \tau\}}\right] && (7.36) \\
&\geq \frac{S}{20}\sqrt{n\tau}\,\mathbb{P}\left(T_i \geq \tau\right) && (7.37) \\
&= \frac{1}{20}\sqrt{\frac{SnK}{2(1-\gamma)}}\,\mathbb{P}\left(T_i \geq \tau\right), && (7.38)
\end{aligned}$$

where Equation (7.36) is true because the expected regret is non-decreasing function of the number of episodes, Equation (7.37) comes from Equation (7.35) and Equation (7.38) from the definition of τ.

We show in the Lemma 7.14 below that $\mathbb{P}\left(T_i \leq K/(2S(1-\beta))\right) \leq 8S/K$. This shows that for $K \geq 16S$, one has $\mathbb{P}\left(T_i \geq \tau\right) \geq 1/2$. This concludes the proof as $40\sqrt{2} \leq 60$.

Lemma 7.14

Recall that T_i is the number of time steps that the MDP is in state $\boldsymbol{i}$ for the MDP model above. Let G_k be a sequence of i.i.d. *Bernoulli random variable of mean*

$1/S$ and let H_k be an independent i.i.d. *sequence of geometric random variable of parameter $1-\beta$. Then:*

(i) $T_i \sim \sum_{k=1}^{K} G_k H_k$,

(ii) $\mathbb{E}[T_i] = K/(S(1-\beta))$,

(iii) $\mathbb{P}(T_i \geq \mathbb{E}[T_i]/2) \geq 1 - 8S/K$.

Proof. Let G_k be a random variable that equals 1 if the initial state $\boldsymbol{i}$ is chosen at the beginning of episode k and recall that H_k is the episode length. By definition, the variables G_k and H_k are independent and follow respectively Bernoulli and geometric distribution. This shows *(i)*.

Let $W_k = G_k H_k$. As the W_k are *i.i.d.* and G_k and H_k are independent, we have:

$$\mathbb{E}[T_i] = K\mathbb{E}[H_1 G_1] = \frac{K}{S(1-\beta)}.$$

This shows (ii).

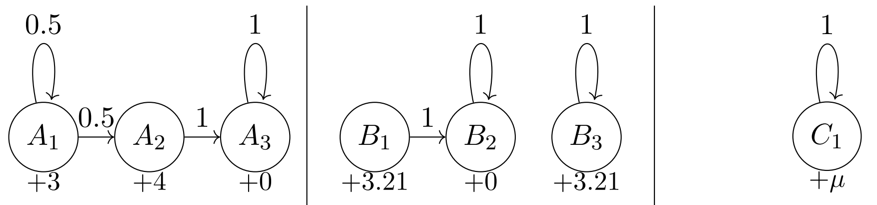
Moreover, $\mathbb{V}[T_i] = K\mathbb{V}[H_1 G_1]$. Hence, by using Chebyshev's inequality, one has:

$$\begin{aligned}\mathbb{P}\left(T_i \leq \frac{\mathbb{E}[T_i]}{2}\right) &\leq \mathbb{P}\left(\|T_i - \mathbb{E}[T_i]\| \geq \frac{\mathbb{E}[T_i]}{2}\right)\\ &\leq \frac{4\mathbb{V}[T_i]}{(\mathbb{E}[T_i])^2}\\ &= \frac{4}{K}\frac{\mathbb{V}[H_1G_1]}{(\mathbb{E}[H_1G_1])^2}.\end{aligned}$$

Concerning the variance, the second moment of a geometric random variable of parameter $1-\beta$ is $(1+\beta)/(1-\beta)^2$. This shows that $\mathbb{E}[(H_1G_1)^2] = (1+\beta)/(S(1-\beta)^2) \leq 2S(\mathbb{E}[H_1G_1])^2$. This implies:

$$\mathbb{V}[H_1G_1] \leq (2S-1)(\mathbb{E}[H_1G_1])^2 \leq 2S(\mathbb{E}[H_1G_1])^2.$$

This implies (iii). ☐



(a) $\hat{\boldsymbol{P}}_a$ and $\hat{\boldsymbol{r}}_a = \boldsymbol{r}_a$. (b) $\hat{\boldsymbol{P}}_b = \boldsymbol{P}_b$ and $\hat{\boldsymbol{r}}_b = \boldsymbol{r}_b$. (c) $\hat{\boldsymbol{P}}_c = \boldsymbol{P}_c$ and $\hat{\boldsymbol{r}}_c = \boldsymbol{r}_c$.

Figure 7.2.: Counterexample for OFU indices: $\hat{\mathcal{B}}_a$, $\hat{\mathcal{B}}_b = \mathcal{B}_b$, $\hat{\mathcal{B}}_c = \mathcal{B}_c$.

7.C Proof of Theorem 7.4

In this proof, we reason by contradiction and assume that there exists a procedure that computes local indices such that the obtained policy is such that for any estimate $\hat{\mathcal{B}}$ and any initial condition ρ, then if $M \in \mathbb{M}(\hat{\mathcal{B}})$, one has

$$\sup_{M \in \mathbb{M}(\hat{\mathcal{B}})} v_M^{\pi^{I(\hat{\mathcal{B}})}}(\rho) \geq \sup_{\pi} v_M^{\pi}(\rho). \tag{7.39}$$

In the remaining of this section, we set the discount factor to $\gamma = 0.5$. For a given state s_a, we denote by $I(s_a)$ the local index of state s_a computed by this hypothetically optimal algorithm.

We first consider a Markovian bandit problem with two arms $\{b, c\}$. We consider that these two arms are perfectly estimated (i.e., $\beta_r(s_b) = \beta_P(s_b) = \beta_r(s_c) = \beta_P(s_c) = 0$ for any s_b, s_c). The Markov chains for these arms are depicted in Figure 7.2. Their transitions matrices and rewards are

$$\boldsymbol{P}_b = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \boldsymbol{r}_b = [3.21, 0, 3.21]; \qquad \boldsymbol{P}_c = [1] \text{ and } \boldsymbol{r}_c = [\mu].$$

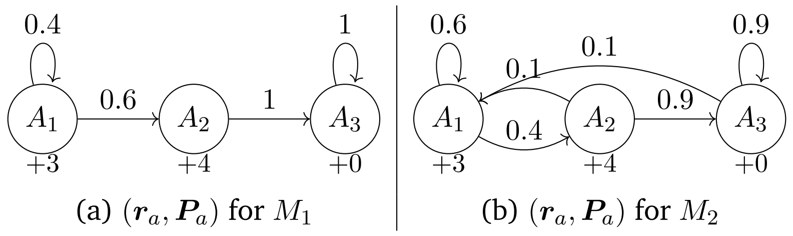
As the Markovian bandit are perfectly known, the indices $I(B_1)$, $I(B_2)$, $I(B_3)$ and $I(C_1)$ must be such that the obtained priority policy is optimal for the true MDP, that is: states B_1 and B_3 should have priority over C_1 (i.e., $I(B_1) > I(C_1)$ and $I(B_3) > I(C_1)$) if and only if $\mu < 3.21$, and state B_2 should have priority over C_1 (i.e., $I(B_2) > I(C_1)$) if and only if $\mu < 0$ where μ is the reward incurred in state C_1. This implies that the local indices defined by our hypothetically optimal algorithm must satisfy

$$I(B_1) = I(B_3) > I(B_2).$$

Now, we consider Markovian bandit problems with two arms $\{a, b\}$, where Arm b is as before. For Arm a, we consider a confidence set for $\hat{\mathcal{B}}_a := (\hat{\boldsymbol{r}}_a, \hat{\boldsymbol{P}}_a, \beta_r, \beta_P)$ where $(\hat{\boldsymbol{r}}_a, \hat{\boldsymbol{P}}_a)$ are depicted in Figure 7.2(a) and where $\beta_r(s_a) = 0$ and $\beta_P(s_a) = 0.1$:

$$\hat{\boldsymbol{P}}_a = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \text{ and } \hat{\boldsymbol{r}}_a = \boldsymbol{r}_a = [3, 4, 0] \quad \beta_P = [0.1, 0.1, 0.1] \text{ and } \beta_r = [0, 0, 0].$$

We consider two possible instances of the "true" Markovian bandit problem, denoted M_1 and M_2. For M_1, the transition matrix and reward function of the first arm are depicted in Figure 7.3(a). For M_2, they are depicted in Figure 7.3(b). In both cases, $(\boldsymbol{r}_b, \boldsymbol{P}_b)$ are as in Figure 7.2(b). It should be clear that $M_1 \in \mathbb{M}$ and $M_2 \in \mathbb{M}$.



0.4
1
0.6
1
A_1
A_2
A_3
+3
+4
+0
(a) $(\boldsymbol{r}_a, \boldsymbol{P}_a)$ for M_1
0.6
0.1
0.9
0.1
0.9
0.4
A_1
A_2
A_3
+3
+4
+0
(b) $(\boldsymbol{r}_a, \boldsymbol{P}_a)$ for M_2

Figure 7.3.: The two instances $\mathcal{B}_{a^1}$ and $\mathcal{B}_{a^2}$ of $\mathcal{B}_a$.

If there exist indices that can be computed locally, then the indices for an arm should not depend on the confidence that one has on the other arms. The indices $I(A_1)$, $I(A_2)$ and $I(A_3)$ must satisfy the following facts:

- $I(A_3) \in (I(B_2), I(B_3))$ because for all Markovian bandit $M \in \mathbb{M}$, state A_3 should have priority over state B_2 and should not have priority over state B_3 (because of the discount factor $\gamma = 1/2$).

- $I(A_2) > I(B_1) = I(B_3)$ because for all Markovian bandit $M \in \mathbb{M}$, state A_2 will give a higher instantaneous reward than state B_1 or B_3. It should therefore have a higher priority.

This leaves two possibilities for $I(A_1)$:

- If $I(A_1) > I(B_1) = I(B_3)$, then state A_1 has priority over both B_1 and B_3. We denote the corresponding priority policy π_1.

- If $I(A_1) < I(B_1) = I(B_3)$, then state B_1 and B_3 have a higher priority than state A_1. We denote the corresponding priority policy by π_2.

We use a numerical implementation of extended value iteration[2] to find that:

$$\sup_{M\in\mathbb{M}} V_M^{\pi_2}(A_1,B_3) \approx 6.42 < \sup_{\pi} V_{M_1}^{\pi}(A_1,B_3) \approx 6.47$$
$$\sup_{M\in\mathbb{M}} V_M^{\pi_1}(A_1,B_1) \approx 5.96 < \sup_{\pi} V_{M_2}^{\pi}(A_1,B_1) \approx 6.00$$

This implies that there does not exist any definition of indices such that Equation (7.10) holds regardless of M and $\boldsymbol{s}$.

7.D Description of the Algorithms and Choice of Hyperparameter

In this section, we provide a detailed description of the simulation environment used in the paper. We first describe the Markov chain used in our example. Then, we describe all algorithms that we compare in the paper. For each algorithm, we give some details about our choice of hyperparameters. Last, we also describe the experimental methodology that we used in our simulations.

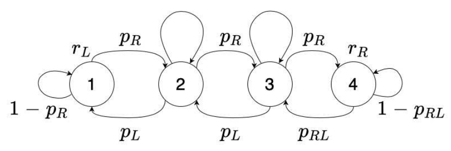
7.D.1 Description of the example

We design an environment with 3 arms, all following a Markov chain represented in Table 7.1. This Markov chain is obtained by applying the optimal policy on the river swim MDP of [FCG10]. In each chain, there are 2 rewarding states: state 1 with low mean reward r_L, and state 4) with high mean reward r_R, both with Bernoulli distributions. At the beginning of each episode, all chains start in their state 1. Each chain is parametrized by the values of $p_L, p_R, p_{RL}, r_L, r_R$ that are given in Table 7.1 along with the corresponding Gittins indices of each chain.

7.D.2 MB-PSRL

MB-PSRL, the adaption from PSRL, puts prior distribution on the parameters $(\boldsymbol{r}_a, \boldsymbol{P}_a)$ of each Arm a, draws a sample from the posterior distribution and uses it to compute the Gittins indices at the start of each episode. We implement two posterior updates for the mean reward vector $\boldsymbol{r}_a$: Beta and Gaussian-Gamma. The second posterior,

[2]available at `https://gitlab.inria.fr/kkhun/learning-in-rested-markovian-bandit`



					Gittins index for each state			
p_L	p_R	p_{RL}	r_L	r_R	1	2	3	4
0.1	0.2	0.3	0.2	1.0	0.276	0.2894	0.392	1.0
0.1	0.5	0.7	0.35	0.7	0.35	0.256	0.2892	0.7
0.1	0.4	0.5	0.4	0.65	0.4	0.250	0.286	0.65

Table 7.1.: The random walk chain with 4 states. In state 4, the chain has an average reward r_R. For state 2 and 3, the chain gives zero reward. In state 1, the mean reward is r_L. This chain is obtained by applying the optimal policy on the 4-state river swim MDP of [FCG10]. The table contains the parameters that we used, along with Gittins indices of all states when the discount factor is $\gamma = 0.99$.

Gaussian-Gamma, will be used in prior choice sensitivity tests. For the transition matrix $\boldsymbol{P}_a$, we implemented Dirichlet posterior update because Dirichlet distribution is the only natural conjugate prior for categorical distribution. Beta, Gaussian-Gamma and Dirichlet distributions can be easily sampled using the `numpy` package of Python. This greatly contributes to the computational efficiency of MB-PSRL.

We give more details on this prior distribution and their conjugate posterior in the subsections below.

Bayesian Updates: Conjugate Prior and Posterior Distributions

MB-PSRL is a Bayesian learning algorithm. As such, it samples reward vectors and transition matrices at the start each episode. We would like to emphasize that neither the definition of the algorithm nor its performance guarantees that we prove in Theorem 7.2 depend on a specific form of the prior distribution ϕ. Yet, in practice, some prior distributions are more preferable because their conjugate distributions are easy to implement. In the following, we give concrete examples on how to update the conjugate distribution given the observations.

For $a \in [n]$ and $s_a \in \mathcal{S}_a$, let $N^k(s_a)$ be the number of activations of arm a while in state s_a up to episode k. For this state s_a, the number of samples of the reward and of transitions from s_a are equal to $N^k(s_a)$. To ease the exposition, we drop the label a and assume that we are given:

- $N^k(s)$ *i.i.d.* samples $\{Y_1, \dots, Y_{N^k(s)}\}$ of next states to which the arm transitioned from s.

- $N^k(s)$ *i.i.d.* samples $\{R_1, \dots, R_{N^k(s)}\}$ of random immediate rewards earned while the arm was activated in state s

Each Y_i is such that $\mathbb{P}(Y_i = s') = P(s, s')$ and each R_i is such that $\mathbb{E}[R_i] = r(s)$. In what follows, we describe natural priors that can be used to estimate the transition matrix and the reward vector.

Transition Matrix

If no information is known about the arm, the natural prior distribution is to consider the lines $P(s, \cdot)$ of the matrix as independent multivariate random variables uniformly distributed among all non-negative vectors of length S that sum to 1. This corresponds to a Dirichlet distribution of parameters $\alpha = (1 \ \dots \ 1)$. For a given s, the variables $\{Y_1, \dots, Y_{N^k(s)}\}$ are generated according to a categorical distribution $\boldsymbol{P}(s, \cdot)$. The Dirichlet distribution is self-conjugate with respect to the likelihood of a categorical distribution. So, the posterior distribution $\phi(\boldsymbol{P}(s, \cdot) | Y_1, \dots, Y_{N^k(s)})$ is a Dirichlet distribution with parameters $\boldsymbol{c} = (c_1 \ \dots \ c_S)$ where $c'_s = 1 + \sum_{i=1}^{N^k(s)} \mathbb{I}_{\{Y_i = s'\}}$.

Reward Distribution

As for the reward vector, the choice of a good prior depends on the distribution of rewards. We consider two classical examples: Bernoulli and Gaussian.

Bernoulli distribution A classical case is to assume that the reward distribution of a state s is Bernoulli with mean value $r(s)$. A classical prior in this case is to consider that $\{r(s)\}_{\{s \in \mathcal{S}\}}$ are *i.i.d.* random variables following a uniform distribution whose support is $[0, 1]$. The posterior distribution of $r(s)$ at time t is the distribution of $r(s)$ conditional to the reward observations from state s gathered up to time t. The posterior distribution $\phi(r(s) \mid R_1, \dots, R_{N^k(s)})$ is then a Beta distribution with parameters $(1 + \sum_{i=1}^{N^k(s)} \mathbb{I}_{\{R_i = 1\}}, 1 + \sum_{i=1}^{N^k(s)} \mathbb{I}_{\{R_i = 0\}})$. Recall that the Beta distribution is a special case of the Dirichlet distribution in the same way as the Bernoulli distribution is a special case of the Categorical distribution.

Gaussian distribution We now consider the case of Gaussian rewards, and we assume that the immediate rewards earned in state s are *i.i.d.* Gaussian random variables of mean and variance $(r(s), \sigma^2(s))$. A natural prior for Gaussian rewards is to consider that $(r(s), \frac{1}{\sigma^2(s)})$ are *i.i.d.* bivariate random variables where the marginal distribution of each $\frac{1}{\sigma^2(s)}$ is a Gamma distribution (it is a natural belief since the empirical variance of Gaussian has a chi-square distribution which is a special case of Gamma distribution). Conditioned on $\frac{1}{\sigma^2(s)}$, $r(s)$ follows a Gaussian distribution of variance $\sigma^2(s)$. We say that $(r(s), \frac{1}{\sigma^2(s)})$ has a Gaussian-Gamma distribution, which is self-conjugate with respect to a Gaussian likelihood (*i.e.,* the likelihood of Gaussian rewards). So, given the reward observations, the marginal distribution of $\frac{1}{\sigma^2(s)}$ is still a Gamma distribution. $r(s)$ has Gaussian distribution conditioned on the reward observations and $\frac{1}{\sigma^2(s)}$. Indeed, let $\hat{r}(s) = \frac{1}{N^k(s)} \sum_{i=1}^{N^k(s)} R_i$ and $\hat{\sigma}^2(s) = \frac{1}{N^k(s)} \sum_{i=1}^{N^k(s)} (R_i - \hat{r}(s))^2$ be the empirical mean and empirical variance of R_i. Then it can be shown that the posterior distribution of $\frac{1}{\sigma^2(s)}$ and $r(s)$ are:

$$\frac{1}{\sigma^2(s)} \mid R_1, \dots, R_{N^k(s)} \sim \text{Gamma}\left(\frac{N^k(s)+1}{2}, \frac{1}{2} + \frac{N^k(s)\hat{\sigma}^2(s)}{2} + \frac{N^k(s)\hat{r}^2(s)}{2(N^k(s)+1)}\right)$$
$$r(x) \mid \frac{1}{\sigma^2(x)}, R_1, \dots, R_{N^k(x)} \sim \mathcal{N}\left(\frac{N^k(s)\hat{r}(s)}{N^k(s)+1}, \frac{\sigma^2(s)}{N^k(s)+1}\right).$$

For more details about the analysis of conjugate prior and posterior presented above as well as more conjugate distributions, we refer the reader to [Fin97; Mur07].

Notice that a reward that has a Gaussian distribution violates the property that all rewards are in $[0, 1]$. This could invalidate the bound on the regret of our algorithm proven in Theorem 7.2. Actually, it is possible to correct the proof to cover the Gaussian case by replacing the Hoeffding's inequality used in Lemma 7.5 by a similar inequality, also valid for sub-Gaussian random variables, see [Ver18]. In the experimental section (see 7.E.3), we also show that a bad choice for the prior distribution of the reward (assuming a Gaussian distribution while the rewards are actually Bernoulli) does not alter too much the performance of the learning algorithm.

7.D.3 Experimental Methodology

In our numerical experiment, we did 3 scenarios to evaluate the algorithms (scenario 2 and 3 are given in Appendix 7.E). In each scenario, we choose the discount factor $\gamma = 0.99$ (which is classical) and we compute the regret over $K = 3000$ episodes. The number of simulations varies over scenario depending on how the regret is computed. For each run, we draw a sequence of horizons $\{H^k\}_{k\in[3000]}$ from a geometric distribution of parameter 0.01, and we run all algorithms for this sequence of time-horizons to remove a source of noise in the comparisons.

For a given sequence of policies π^k, following Equation (7.3), the expected regret is $\mathbb{E}\left[\sum_{k=1}^{K}\Delta^k(\boldsymbol{s}_{t^k})\right]$ where $\Delta^k(\boldsymbol{s}_{t^k})$ is the expected regret over episode k. To reduce the variance in the numerical experiment, we compute $\Delta^k(\boldsymbol{s}_{t^k}) = v_M^{\pi_*}(\boldsymbol{s}_{t^k}) - v_M^{\pi^k}(\boldsymbol{s}_{t^k})$. For a given Markovian bandit problem and state $\boldsymbol{s}$, the value $v_M^{\pi^*}(\boldsymbol{s})$ can be computed by using the retirement evaluation presented in Page 272 of [Whi96]. It seems, however, that the same methodology is not applicable to compute the value function of an index policy that is not the Gittins policy. This means that while the policy π^k is easily computable, we do not know of an efficient algorithm to compute its value $v_M^{\pi^k}(\boldsymbol{s})$. Hence, in our simulations, we will use two methods to compute the regret, depending on the problem size:

1. (Exact method) Let (r^π, P^π) be the reward vector and transition matrix under policy π (i.e. $\forall \boldsymbol{s}, \boldsymbol{s}' \in \mathcal{X}, r^\pi(\boldsymbol{s}) = r(\boldsymbol{s}, \pi(\boldsymbol{s})), P^\pi(\boldsymbol{s}, \boldsymbol{s}') = P^{\pi(\boldsymbol{s})}(\boldsymbol{s}, \boldsymbol{s}')$ as defined in Equation (7.1)). Using the Bellman equation, the value function under policy π is computed by

$$v_M^\pi = (I - \gamma P^\pi)^{-1} r^\pi. \tag{7.40}$$

   The matrix inversion can be done efficiently with the `numpy` package of Python. However, this takes $S^{2n} + 2S^n$ of memory storage. Hence, when the number of states and arms are too large, the exact computation method cannot be performed.

2. (Monte Carlo method) In Scenario 2, the model has $n = 9$ arms with $S = 11$ states each, which makes the exact method inapplicable. In this case, it is still possible to compute the optimal policy and to apply Gittins index based algorithms but computing their value is intractable. In such a case, to measure the performance, we do 240 simulations for each algorithm and try to approximate Δ^k by

$$\hat{\Delta}^k = \frac{1}{\#\text{replicas}} \sum_{j=1}^{\#\text{replicas}} \sum_{t=1}^{H^{k,(j)}} \left[r(s^{*,(j)}_{t,a_t^{*,(j)}}) - r(s^{(j)}_{t,a_t^{(j)}}) \right], \tag{7.41}$$

   where $H^{k,(j)}$ is the horizon of the kth episode of the jth simulation and $\{s^{*,(j)}_{t,a_t^{*,(j)}}\}$ and $\{s^{(j)}_{t,a_t^{(j)}}\}$ are the trajectories of the oracle and the agent respectively. The term oracle refers to the agent that knows the optimal policy π^*.

Note that the expectation of Equation (7.41) is equal to the value given in Equation (7.40) but Equation (7.41) has a high variance. Hence, when applicable (Scenario 1 and 3) we use Equation (7.40) to compute the expected regret.

7.E Additional Numerical Experiments

7.E.1 Scenario 1: Small Dimensional Example (Random Walk chain)

This scenario is explained in Appendix 7.D.1 and the main numerical results are presented in Section 6.7. Here, we provide the result with error bars with respect to the random seed. In Figure 7.4, the error bar size equals twice the standard deviation over 80 samples (each sample is a simulation with a given random seed and the random seeds are different for different simulations).

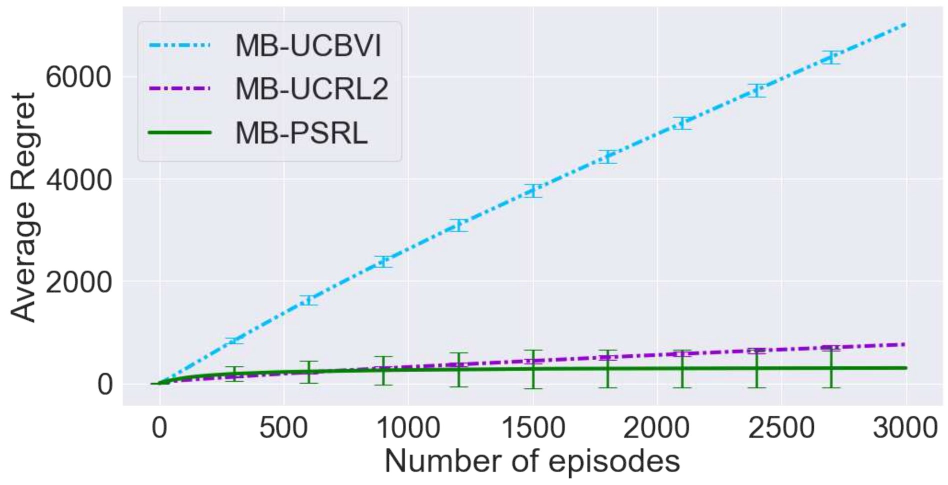


Figure 7.4.: Average cumulative regret in function of the number of episodes. Result from 80 simulations in a Markovian bandit problem with three 4-state random walk chains given in Table 7.1. The horizontal axis is the number of episodes. The size of the error bar equals twice the standard deviation over 80 simulations.

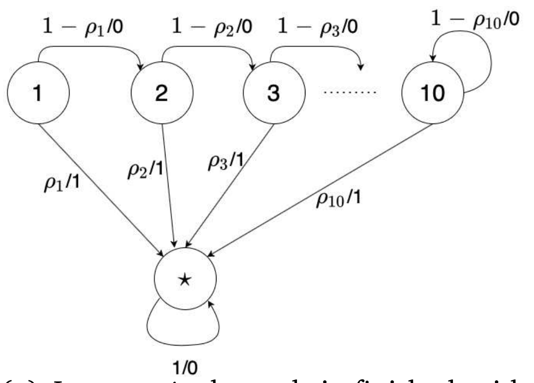
7.E.2 Scenario 2: Higher Dimensional Example (Task Scheduling)

We now study an example that is too large to apply MB-UCRL2 . Hence, here we only compare MB-PSRL and MB-UCBVI.

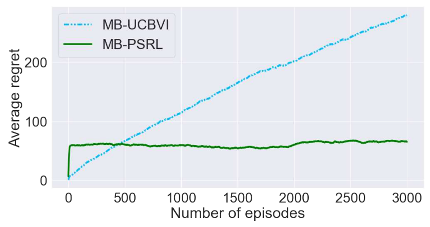
We implement the environment proposed on page 19 of [Duf95] that was used as a benchmark for the algorithm in the cited paper. Each chain represents a task that needs to be executed, and is represented in Figure 7.5(a). Each task has 11 states (including finished state $\star$ that is absorbing). For a given chain $a \in \{1, \dots, 9\}$ and a state $i \in \{1, \dots, 10\}$, the probability that a task a ends at state i is $\rho_i^{(a)} = \mathbb{P}\left(\tau^{(a)} = i \mid \tau^{(a)} \geq i\right)$ where $\tau^{(a)}$ is the execution time of task a. We choose the same values of the parameters as in [Duf95]: $\rho_1^{(a)} = 0.1a$ for $a \in \{1, \dots, 9\}$, $\lambda = 0.8$, $\gamma = 0.99$ and for $i \geq 2$,

$$\mathbb{P}\{s_a = i\} = \left[1 - [1 - \rho_1^{(a)}]\lambda^{i-1}\right] [1 - \rho_1^{(a)}]^{i-1} \lambda^{\frac{(i-1)(i-2)}{2}}.$$

Hence, the hazard rate $\rho_i^{(a)}$ is increasing with i. The reward in this scenario is deterministic: the agent receives 1 if the task is finished (*i.e.,* under the transition from any state i to state $\star$) and 0 otherwise (*i.e.,* any other transitions including the one from state $\star$ to itself). For MB-PSRL, we use a uniform prior for the expected rewards and consider that the rewards are Bernoulli distributed.



(a) In state i, the task is finished with probability ρ_i or transitions to state $i+1$ with probability $1-\rho_i$. For $i = 1, \dots, 10$, the transition from state i to state $\star$ provides 1 as the immediate reward. Otherwise, the agent always receives 0 reward.



(b) Average cumulative regret over 240 simulations.

Figure 7.5.: Task Scheduling with 11 states including the absorbing state (finished state).

The average regret of the two algorithms is displayed in Figure 7.5(b). As before, MB-PSRL outperforms MB-UCBVI. Note that we also studied the time to run one simulation for 3000 episodes. This time is around 1 min for MB-PSRL and MB-UCBVI.

7.E.3 Scenario 3: Bayesian Regret and Sensitivity to the Prior

In this section, we study how robust the two implementations of PSRL are, namely MB-PSRL and vanilla PSRL (to simplify, we will just call the latter PSRL), to a choice of prior distributions. As explained in Appendix 7.D.2, the natural conjugate prior for Bernoulli reward is the Beta distribution. In this section, we simulate MB-PSRL and PSRL in which the rewards are Bernoulli but the conjugate prior used for the rewards are Gaussian-Gamma which is incorrect for Bernoulli random reward. In other words, MB-PSRL and PSRL have Gaussian-Gamma prior belief while the real rewards are Bernoulli random variables.

To conduct our experiments, we use a Markovian bandit problem with three 4-state random walk chains represented in Table 7.1. We draw 16 models by generating 16 pairs of (r_L, r_R) from $U[0, 1]$, 16 pairs of (p_L, p_R) from Dirichlet(3,(1,1,1)) and 16 values of p_{RL} from Dirichlet(2, (1,1)) for each chain. Each model is an unknown MDP that will be learned by MB-PSRL or PSRL. For each of these 16 models, we simulate MB-PSRL and PSRL 5 times with correct priors and 5 times with incorrect priors. The result can be found in Figure 7.6 which suggests that MB-PSRL performs better when the prior is correct and is relatively robust to the choice of priors in terms of Bayesian regret. This figure also shows that PSRL seems more sensitive to the choice of prior distribution. Also note that for both MB-PSRL and PSRL, some trajectories deviate a lot from the mean, under correct priors but even more so with incorrect priors. This illustrates the general fact that learning can go wrong, but with a small probability.

7.F Experimental environment

The code of all experiments is given in a separated zip file that contains all necessary material to reproduce the simulations and the figures.

Our experiments were run on HPC platform with 1 node of 16 cores of Xeon E5. The experiments were made using Python 3 and Nix and submitted as supplementary material and will be made publicly available with the full release of the paper. The package requirement is detailed in README.md. Using only *1 core* of Xeon E5, the Table 7.2 gives some orders of duration taken by each experiment (with discount factor $\gamma = 0.99$, and 3000 episodes per simulation). We would like to draw two remarks. First, the duration reported in Figure 7.1b is the time for policy computation (algorithm's parameters update and policy computation). The duration

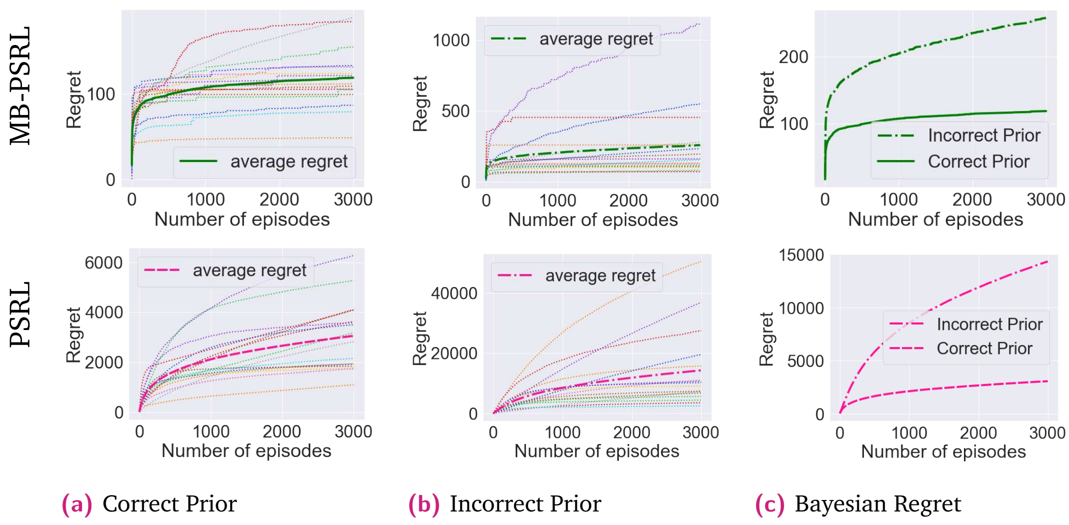


Figure 7.6.: Bayesian regret of MB-PSRL and vanilla PSRL in 3 4-state Random Walk chains. For each chain, we draw 16 random models and run the algorithms for 5 simulations in each model (there are 80 simulations in total). In panels (a) and (b), we plot 16 dotted lines that correspond to the average cumulative regret over 5 simulations in the 16 samples. The solid and dash-dot lines are the average regret each over 80 simulations (the estimated Bayesian regret). Figure 7.6a shows the performance when reward prior is well-chosen (namely, $U([1,1])$). Figure 7.6b is when the reward prior is incorrectly chosen (namely Gaussian-Gamma distribution). Figure 7.6c compares the Bayesian regret of the correct prior with the incorrect one (dash-dot line). In both case, the prior of next state transition is well-chosen (namely, Dirichlet distribution). Y-axis range changes for each figure.

reported in Table 7.2 includes this plus the computation time for oracle (because we track the regret), the state transition time along the trajectories of oracle and of each algorithm, resetting time... This explains why the duration reported in Table 7.2 cannot be compared to the duration reported in Figure 7.1b. Second, the duration shown in Table 7.2 are meant to be a rough estimation of the computation time (we only ran the simulation once and the average duration might fluctuate).

Experiment	MB-PSRL	PSRL	MB-UCRL2	MB-UCBVI	Total
Scenario 1	40 min	-	6 days	50 min	6 days
Scenario 2	200 min	-	-	200 min	400 min
Scenario 3	90 min	260 min	-	-	350 min

Table 7.2.: Approximative execution time for simulating each algorithm and tracking its regret in each scenario. This time includes the time given in Figure 7.1b and the computation time needed by oracle (because we track the regret), the state transition time along the trajectories of oracle and each algorithm, etc. In each scenario, we set the discount factor $\gamma = 0.99$ and run the algorithms for 3000 episodes per simulation.

8 Learning in Average Reward Restless Markovian Bandits

In the previous chapter, we adapted a few learning algorithms to discounted rested Markovian bandits and showed that their regrets, which were upper bounded sublinearly in the number of arms, matched the minimax Bayesian regret we derived in the chapter. We also showed that there was no index definition such that, relying on the confidence bonuses on arms' transition, computing the optimistic index on each arm independently of the other arms guaranteed the optimism in face of uncertainty (OFU) principle in general discounted rested bandits. This chapter considers the learning problem in restless Markovian bandits with average reward criterion. Such a bandit is a Markov decision process (MDP) that suffers from the curse of dimensionality, and general-purpose learning algorithms are not efficient when directly applied. Recently, a few learning algorithms have been specifically designed for restless bandits. Yet, they most often work only for very particular subclasses of restless bandits or require conditions that are computationally hard to verify. Hence, in this chapter, we provide some arguments to explain why those conditions are needed when learning in restless Markovian bandits. We show that the properties of the local arms (like ergodic or small diameter) do not generally imply similar properties for the bandit. Therefore, defining a subclass of restless bandits with desirable properties (like small diameter) by only making assumptions about arms is difficult. Finally, we discuss a few issues when learning in the general class of restless bandits and present RB-TSDE [AM22] along with its regret analysis under a few technical assumptions.

Section 8.1 provides the context of learning in restless bandit problems and presents our contributions. We discuss several existing works that design reinforcement learning (RL) algorithms for restless bandit problems in Section 8.2. We then recall the notations and the problem formulation of the average reward restless bandit in Section 8.3. Section 8.4 accumulates examples and counter-examples that respectively show "desirable" and "non-desirable" properties of restless bandit when its arms all have "desirable" properties. Section 8.5 discusses the requirements when using Whittle index policy as the baseline policy in regret definition. We also present an overview of regret analysis of RB-TSDE [AM22], a modified version of TSDE [Ouy+17] for learning in restless bandits. Section 8.6 concludes this chapter.

8.1 Contributions

Restless Markovian bandits are the structured MDPs that manifest the curse of dimensionality. The apparent effect of the curse is that the bandit's state size is exponential in the number of arms.

For learning generic MDPs with average reward criterion, the current best algorithms have a regret over T time steps bounded by $\tilde{O}(\sqrt{HSAT})$ in the unknown weakly communicating MDP with state size S, action size A, and an upper bound on the span of the optimal bias function H (see Table 3.2 for more algorithms with regret guarantee). From what we have observed, the MDP properties that appear in regret upper bound are:

- the diameter D (defined by Definition 3.3) used in e.g., [JOA10; FPL20];
- the upper bound on the span of the optimal bias function $H \geq sp(\boldsymbol{h}^*)$ used in e.g., [BT12; Ouy+17; Fru+18; ZJ19];
- the mixing time t_{mix} (defined by Definition 8.1) used in e.g., [Ort20].

So, applying these algorithms to learn restless Markovian bandits with average reward criterion provokes a few critical issues:

- The state size of restless bandit is exponential in the number of arms.
- The diameter D (if defined), the mixing time t_{mix} (if defined), or the upper bound on the span of the optimal bias H may also be exponential in the number of arms.
- Compute an optimal policy in restless bandit is PSPACE-hard [PT94], let alone the computation of optimistic policy.

So, the regret bound of those general-purpose RL algorithms is not scalable with the number of arms. A few algorithms specifically designed for restless bandits successfully eliminate the exponentiality in the number of arms from the state size of the bandit in their regret bound (see e.g., [Ort+12; JAT19; AM22]). Yet, no explicit dependency between the diameter, the mixing time, or the upper bound on the span of the optimal bias function that appears in their regret bound and the number of arms is given. This calls for definitions of subclasses of restless bandits whose MDP properties can be expressed polynomially in the number of arms.

In this chapter, we provide a few results that support this argument and study how the assumptions on arms' structure translate into the structure of the bandit. First, we show that no matter how good a learning algorithm is, there is a restless bandit whose arms are unichain that makes the algorithm suffer an expected regret linear in the total steps regardless of how many total steps the algorithm wants to learn. This implies that no RL algorithms can perform uniformly well over the general class of restless bandits whose arms are unichain.

Furthermore, we study the restless bandits having richer arm structures. We show that a restless bandit can be multichain even though its arms are ergodic. Moreover, we also show that:

- for restless bandits whose arms are ergodic,
  - there is an ergodic bandit whose diameter is exponential in the number of arms;
  - there is an ergodic bandit whose mixing time can be as big as we want;
- for restless bandits that are communicating, there is a bandit whose arms have a bounded span of local bias function that has a span of global bias function as large as we want.

We also provide a positive result related to the ergodicity coefficient of the bandit. Finally, we discuss model-based RL algorithms that use Whittle index policy as the baseline policy in regret definition. We also provide an overview of regret analysis of RB-TSDE [AM22] when learning in restless bandits. All of our results suggest that the MDP properties of restless bandit can be exponential in the number of arms in the general class of restless bandits. This calls for work to design subclasses of restless bandits whose MDP properties can be expressed polynomially in the number of arms. Another research direction is to develop algorithms that directly learn the Whittle index of the unknown restless bandit in model-free style.

8.2 Related work

We believe that there are at least two directions of research for learning in an unknown restless Markovian bandit: (1) model-free algorithms that directly estimate the Whittle index and (2) model-based algorithms that estimate the parameters of the unknown restless bandit.

Since the Whitle index policy is asymptotically optimal when the restless bandit satisfies some conditions [WW90] and performs exceptionally well in practice (see e.g., [GM02; Ans+03; GRK06]), the first direction consists in learning the Whittle index of the unknown fully observed restless bandit. The celebrated Q-learning (QL) algorithm [Wat89] is one of the most popular approaches. For the discrete state-space restless arm, the work of [GJN21] learns the Whittle index by maintaining two Q-functions, updating them using QL algorithm, and deducing the Whittle index from them when needed. Meanwhile, the work of [Fu+19] learns the Whittle index by maintaining only one Q-function, updating it using QL algorithm, and deducing the Whittle index as the smallest critical penalty that equalizes the estimated Q-values of action activate and action rest. Yet both works [GJN21; Fu+19] only provide a numerical proof of convergence to the correct Whittle index, and no theoretical guarantee is given. Using the ideas from two-timescale stochastic approximation (see e.g., [ABB01; Sch93]), the work of [AB22] proposes a two-timescale QL algorithm for learning the Whittle index of any unknown indexable restless arms. Their algorithm maintains a Q-function and a vector of penalty separately. It updates Q-function using QL algorithm and the vector of penalty using a stochastic iterative algorithm with a slower timescale update compared to the QL algorithm. The theoretical and numerical proofs of convergence are provided in their work [AB22]. For continuous state-space restless arm, the work of [Nak+21] uses a deep reinforcement learning framework to estimate the Whittle indices of the arms with large state space or convoluted transition kernel, assuming a notion of strong indexability.

The second direction is to design a model-based learning algorithm with a theoretical performance guarantee. For partially observed restless bandit, the work of [Ort+12] derives colored-UCRL algorithm, a modified version of UCRL2, that achieves a regret bounded proportionally to $\sqrt{T}$ where T is the total steps. In its regret bound, colored-UCRL successfully removes the exponentiality in the number of arms from the state size of the bandit. However, the mixing time parameter appears in the bound, and the dependency between the mixing time and the number of arms is unclear. The same discussion goes to the work of [JAT19], which adapts TSDE [Ouy+17] to the learning setting of restless bandit and assumes a condition similar to [Ort+12]. To the best of our knowledge, Restless-UCB of [WHL20] is the first algorithm that has a regret bounded explicitly linearly in the number of arms for partially observed restless bandits. Yet, we must mention that their regret bound is proportional to $T^{2/3}$, and their result is valid for a very restrictive class of restless bandits in which each arm is a birth-death Markov reward process whose state transition satisfies a few technical constraints. Also, Restless-UCB requires a generative model to uniformly explore the dynamic of each arm before committing to its optimistic planning. All of these works also assume an oracle that knows how to compute an optimal policy given a restless bandit. Finally, the work of [AM22] adapts TSDE [Ouy+17] to the fully observed restless bandit problems. It successfully removes the exponentiality in the number of arms from the state size of the bandit in the Bayesian regret bound of RB-TSDE [AM22], the modified version of TSDE [Ouy+17]. However, the ergodicity coefficient of the bandit appears in the regret bound, and the dependency between such a coefficient and the number of arms is unclear. It may be useful to note that the ergodicity coefficient provides an upper bound on the MDP's mixing time and the span of the bias function.

8.3 Restless bandit with average reward criterion

We recall from Chapter 4 that a restless Markovian bandit is multi-armed bandit having $n \in \mathbb{N}^+$ arms. Each arm $\langle \mathcal{S}_i, \{0,1\}, \{\boldsymbol{r}_i^0, \boldsymbol{r}_i^1\}, \{\boldsymbol{P}_i^0, \boldsymbol{P}_i^1\}\rangle$ is an MDP with a finite state space $\mathcal{S}_i$ of size S and a binary action space $\{0,1\}$, where 0 denotes the action "rest", and 1 denotes the action "activate". If arm i is in state s_i, and the decision maker executes $a_i \in \{0,1\}$, the arm incurs a random reward with the expected value $r_i^{a_i}(s_i)$ and transitions to state $s_i' \in \mathcal{S}_i$ with a probability $P_i^{a_i}(s_i, s_i')$. Similarly to what is done in Chapter 7, we assume that the state space of the arms are pairwise distinct: $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for any $i \neq j$. So, we will drop the index i from the expected reward and transition if no confusion is possible: we denote them by $r^{a_i}(s_i)$ instead of $r_i^{a_i}(s_i)$ and by $P^{a_i}(s_i, s_i')$ instead of $P_i^{a_i}(s_i, s_i')$.

At time step 1, the state of all arms denoted by $\boldsymbol{s}_1 := (s_{1,1}, \dots, s_{1,n})$ is sampled according to some initial distribution ρ over the state space $\mathcal{X} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. At time step $t \geq 1$, the decision maker observes the current state of all arms denoted by $\boldsymbol{s}_t := (s_{t,1}, \dots, s_{t,n})$ and activates exactly m arms encoded by action $\boldsymbol{a}_t := (a_{t,1}, \dots, a_{t,n})$ such that $\boldsymbol{a}_t \in \{0,1\}^n$ and $\sum_{i=1}^n a_{t,i} = m$, and $m \in [n]$ is constant over time. Each arm i incurs then a random reward $r_{t,i}$ and makes a transition to the next state $s_{t+1,i}$ in function of $s_{t,i}$ and $a_{t,i}$ but independently of the other arms. So, the restless Markovian bandit is a specific MDP – that we denote by M – whose state space is $\mathcal{X}$, and action space is $\mathcal{A}(m) := \{\boldsymbol{a} \in \{0,1\}^n : \sum_{i=1}^n a_i = m\}$. We will say *global* to refer to the quantities related to the bandit M and *local* to refer to the quantities related to the arms. Without loss of generality, we assume that for any global state $\boldsymbol{s} \in \mathcal{X}$ and global action $\boldsymbol{a} \in \mathcal{A}(m)$, the expected reward is bounded like $\sum_{i=1}^n r^{a_i}(s_i) \in [0, r_{max}]$.

The decision maker wants to compute a policy $\pi : \mathcal{X} \mapsto \mathcal{A}(m)$ that maximizes the gain as defined in Section 2.4: for any global state $\boldsymbol{s} \in \mathcal{X}$,

$$g_M^\pi(\boldsymbol{s}) := \lim_{T\to+\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T}\sum_{i=1}^{n} r_{t,i} \mid \boldsymbol{s}_1 = \boldsymbol{s}\right]. \tag{8.1}$$

As presented in Chapter 2, if the MDP M has finite state and action spaces, then an optimal policy π^* such that for all $\boldsymbol{s} \in \mathcal{X}, g^{\pi^*}(\boldsymbol{s}) = g^*(\boldsymbol{s}) := \max_\pi g^\pi(\boldsymbol{s})$ exists and is deterministic. However, it is shown in [PT94, Theorem 4] that computing an optimal policy in restless bandit with average reward is PSPACE-hard.

We recall that the diameter of an MDP (see Definition 3.3) is finite if and only if the MDP is communicating. By [JOA10, Appendix A], the diameter of bandit M is lower bounded by

$$\log_{|\mathcal{A}(m)|}|\mathcal{X}| - 3 = \sum_{i=1}^{n} \log_{|\mathcal{A}(m)|}|\mathcal{S}_i| - 3 = n\log_{\binom{n}{m}} S - 3.$$

This lower bound is not exponential in n, which inspires us to study the restless bandit's diameter and gives us hope to remove the exponentiality in n from the regret bound of general-purpose algorithms.

8.3.1 Additional structural properties of MDP

We presented the classification of MDPs in Definition 2.2 of Chapter 2. We also divided the MDP space as shown in Figure 2.1. In addition, we say that an MDP is *recurrent* if, for all policy π, the transition matrix $\boldsymbol{P}^\pi$ defines a Markov chain where all states are recurrent (equivalently, for all states s and s', a Markov chain starting in s will visit s' with probability 1). A recurrent or unichain MDP is called *aperiodic* if all matrices $\boldsymbol{P}^\pi$ are aperiodic. An equivalence of the definition of ergodic MDP in Definition 2.2 is that an MDP is *ergodic* if it is recurrent and aperiodic.

As mentioned in the beginning, there exist learning algorithms that rely on the mixing time of the MDPs. So, we define the mixing time as the following. Following [Wei+20, Definition 5.1],

Definition 8.1 (The mixing time of ergodic MDP)
The mixing time of an ergodic MDP with state space $\mathcal{S}$ is defined as

$$t_{mix} := \max_{\pi} \min \left\{ t \geq 1 : \left\| (\boldsymbol{P}^{\pi})^{t}(s, \cdot) - \mu^{\pi} \right\|_{\ell_1} \leq \frac{1}{4}, \forall s \in \mathcal{S} \right\}.$$

By Definition 8.1, the mixing time is the maximum time required for any policy starting at any initial state to make the state distribution $\frac{1}{4}$-close (in ℓ_1 norm) to the stationary distribution, the state distribution in steady regime.

8.4 Local and global structures in restless bandits

In the following, we study the structure of the bandit when all the local arms are well-structured. To do so, we will provide a few simple examples and counter-examples. In those examples, the bandit always has two arms, and exactly one arm is activated at each decision time. To ease the exposition, Arm 1 is drawn in green and Arm 2 in blue. In each arm, the state transitions of action "activate" are drawn in black and those of action "rest" are drawn in dashed red. In the global MDP, the state transitions when activating Arm 1 are drawn in green arrows, and those when activating Arm 2 are drawn in blue. If applicable, the expected reward and transition probability are noted along the transition arrows. However, if the transition is deterministic, i.e., the transition probability is 1, we only note the expected reward along the transition arrows.

8.4.1 Negative results

In general, the expected regret is frequently bounded based on concentration inequalities. These inequalities encode how well an unknown parameter is estimated. So, it should be safe to say that "if the unknown parameters of the MDP are well estimated, then the regret is bounded sub-linearly in the total steps T". Meanwhile, for a restless Markovian arm having S states, there are at most $2S + 2S^2$ parameters. This means that a restless bandit M described above is a specific MDP with $n(2S + 2S^2)$ parameters instead of $\mathcal{O}(S^n)$. Intuitively, if each arm visits all of its states frequently, then the $2S + 2S^2$ unknown parameters should be well estimated as well as the $n(2S + 2S^2)$ unknown parameters of the bandit. Then, the expected regret in learning such a restless bandit should be bounded not only linearly in n

but also sub-linearly in T. Unfortunately, we show in this section that this reasoning is not always applicable.

Non-learnable restless bandit whose arms are unichain

The following theorem shows that the assumption that each arm is unichain and has a bounded span of local bias function under any policy is not enough to make a restless bandit learnable.

Theorem 8.2 (Non-learnable restless bandit)

(i) For any learning algorithm $\mathcal{L}$, there is a restless bandit whose arms are unichain and have a bounded span of local bias function under any policy such that for any total steps $T \geq 2$, the expected regret of $\mathcal{L}$ at least $\frac{T}{4}$.

(ii) For any learning algorithm $\mathcal{L}$, any total steps $T \geq 2$, there is a weakly communicating restless bandit whose arms are unichain and have a bounded span of local bias function under any policy such that the expected regret of $\mathcal{L}$ is at least $\frac{T}{5}$.

Proof. **Proof of (i)** – consider the two restless bandit problems in Figure 8.2. The arms of both bandits are given in Figure 8.1. All arms are unichain and have a bounded span of local bias under any policy. Both bandits M^a and M^b have two recurrent classes: $\mathcal{X}^1$ whose optimal gain is 0.5, and $\mathcal{X}^2$ whose optimal gain is 1. So, the regret gap when ending up in $\mathcal{X}^1$ instead of $\mathcal{X}^2$ is $\frac{1}{2}$. At time step 1, both M^a and M^b are in state $(3,1)$ with probability 0.5 and state $(3,2)$ with probability 0.5. Consider any learning algorithm $\mathcal{L}$ that knows the parameters of both M^a and M^b but does not know with which bandit it is interacting. In states $(3,1)$, $\mathcal{L}$ activates Arm 1 with probability θ_1, and in state $(3,2)$, $\mathcal{L}$ rests Arm 1 with probability θ_2, where θ_1 and θ_2 are its degree of freedoms to minimize the expected regret.

Facing M^a, $\mathcal{L}$ ends up in $\mathcal{X}^1$ with probability $(1-\theta_1)$ when starting in $(3,1)$ and with probability $(1-\theta_2)$ when starting in $(3,2)$. Then, its expected regret is

$$\begin{aligned}\mathbb{E}\left[\mathrm{Regret}(\mathcal{L}, M^a, T)\right] &= 0.5\Big((1-\theta_1)\frac{T}{2} + \theta_1 \times 0\Big) + 0.5\Big((1-\theta_2)\frac{T}{2} + \theta_2 \times 0\Big) \\ &= \frac{T}{2}\left(1 - \frac{\theta_1+\theta_2}{2}\right).\end{aligned}$$

Facing M^b, $\mathcal{L}$ ends up in $\mathcal{X}^1$ with probability θ_1 when starting in $(3,1)$ and with probability θ_2 when starting in $(3,2)$ and its expected regret is

$$\begin{aligned}\mathbb{E}\left[\text{Regret}(\mathcal{L}, M^b, T)\right] &= 0.5\Big((1-\theta_1)\times 0 + \theta_1\frac{T}{2}\Big) + 0.5\Big((1-\theta_2)\times 0 + \theta_2\frac{T}{2}\Big)\\ &= \frac{T}{2}\times\frac{\theta_1+\theta_2}{2}.\end{aligned}$$

Then $\mathcal{L}$ wants to adjust θ_1 and θ_2 such that

$$\min_{\theta_1,\theta_2\in[0,1]}\left(\max\left\{\frac{T}{2}\left(1-\frac{\theta_1+\theta_2}{2}\right), \frac{T}{2}\times\frac{\theta_1+\theta_2}{2}\right\}\right). \tag{8.2}$$

For the best possible choice of (θ_1,θ_2), the minimum value of (8.2) is $\dfrac{T}{4}$. This means that no learning algorithms can achieve an expected regret smaller than $\frac{T}{4}$ on both models simultaneously. That concludes the proof.

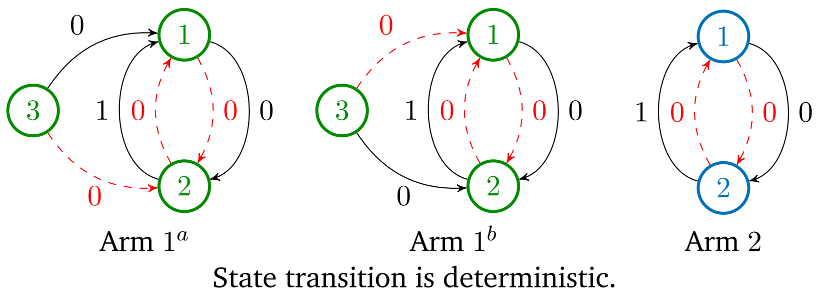


Figure 8.1.: Restless Markovian arms. Arm 1^a and Arm 1^b has 3 states, and Arm 2 has 2 states. The black arrows show state transition of action activate, and the dashed red one for action rest. The numbers along the arrows show the expected reward when executing the actions. The span of bias in both arms are bounded in $[0,1]$. The two corresponding global MDPs is given in Figure 8.2.

Proof of (ii) – It is derived from (i) by slightly modifying the transition of state 1 of Arm 1 as the following: if Arm 1 is activated in state 1, it transitions to state 3 with probability ε or to state 2 with probability $1-\varepsilon$. Then, the modified bandits $M^a(\varepsilon)$ and $M^b(\varepsilon)$ are both weakly communicating. Specifying a very small ε, for example, $\varepsilon = 1/T^2$ and following the proof of (i) conclude the proof. □

The difference between (i) and (ii) of Theorem 8.2 is that the bandits in (i) do not depend on T, while the ones in (ii) do. So, the result of Theorem 8.2 (i) is stronger. Yet, the reason that (ii) is proposed is that the bandits in (i) are not weakly communicating (see Figure 2.1 for the MDP classification). According to Definition 3.2, the regret is not defined in MDPs that are not weakly communicating.

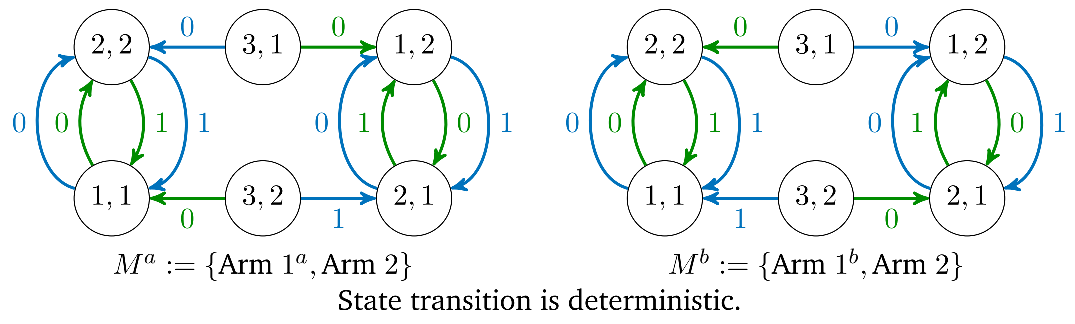

2, 2
3, 1
1, 2
1, 1
3, 2
2, 1
0
1
$M^a := \{\text{Arm } 1^a, \text{Arm } 2\}$
$M^b := \{\text{Arm } 1^b, \text{Arm } 2\}$
State transition is deterministic.

Figure 8.2.: Two restless bandits, each with two arms given in Figure 8.1. Exactly one arm is activated at each decision time, and the initial global state is either $(3,1)$ or $(3,2)$. The global state is denoted by (s_1, s_2), where s_1 is the state of Arm 1 and s_2 of Arm 2. The green arrows show the state transition when activating Arm 1. The blue arrows show the state transition when activating Arm 2. The numbers along the arrows show the expected reward when executing the actions. The global MDP M^a is formed by Arm 1^a and Arm 2, and M^b by Arm 1^b and Arm 2. Both MDPs are multichain and not weakly communicating because state $(3,1)$ and $(3,2)$ are transient, and there are two recurrent classes under any policy: $\mathcal{X}^1 := \{(1,1),(2,2)\}$ and $\mathcal{X}^2 := \{(1,2),(2,1)\}$. The optimal gain in class $\mathcal{X}^1$ is 0.5 and in class $\mathcal{X}^2$ is 1. This shows that the actions in states $(3,1)$ and $(3,2)$ are decisive. In M^a, the optimal action in state $(3,1)$ is to rest Arm 2, and in state $(3,2)$ is to activate Arm 2. In contrast, in M^b, the optimal action in state $(3,1)$ is to activate Arm 2, and in state $(3,2)$ is to rest Arm 2. Starting by either $(3,1)$ or $(3,2)$, no learning algorithms have a sublinear expected regret in both examples.

So, the regret in (i) should be understood as the difference between the cumulative reward of an optimal policy and the cumulative reward of the learner. For (ii), the bandits are weakly communicating, and Definition 3.2 is applied.

Theorem 8.2 shows that no RL algorithms can perform uniformly well over the general class of restless bandits whose arms are unichain. We prove this theorem by constructing a bandit such that one of its arms admits a local transient state (the state 3 of Arm 1 in Figure 8.1). The constructed bandit then has two global transient states (the states $(3,1)$ and $(3,2)$ in Figure 8.2). In fact, if a restless bandit has n arms, each arm has S states, and there exists one arm that admits one local transient state, then the bandit has S^{n-1} global transient states. This inspires us to study the bandits whose arms have no local transient state.

Local recurrence does not imply global recurrence

The following theorem shows that a restless bandit whose arms are ergodic or recurrent is not necessarily unichain.

Theorem 8.3 (Multichain restless bandit)

(i) Among restless bandits whose arms are recurrent, there is a bandit that is not weakly communicating.

(ii) Among restless bandits whose arms are ergodic, there is a bandit that is multichain.

Proof. **Proof of (i)** – It is given by the counter-example in which the restless bandit has two arms shown in Figure 8.3, and exactly one arm is activated at each decision time. For each arm, both "activate" and "rest" actions induce the same state transition. In such example, the restless bandit is a multichain and not weakly communicating MDP (see Figure 2.1 for MDP classification) because there are two recurrent classes under any policy: one composed of $\{(1,1),(2,2)\}$ and the other one is $\{(1,2),(2,1)\}$.

Proof of (ii) – It is given by the following counter-example. Consider a restless bandit with two 8-state arms presented in Figure 8.4. Both arms are identical in terms of state transition, which is summarized in Table 8.1. We observe that when rest, state 1 and 2 have the same possible next states 2 and 3. Similarly, the pair $\{3,4\}$ has $\{4,5\}$, $\{5,6\}$ has $\{6,7\}$, and $\{7,8\}$ has $\{8,1\}$. Similar pattern is observed under action activate: $\{2,3\}$ has $\{3,4\}$, $\{4,5\}$ has $\{5,6\}$, $\{6,7\}$ has $\{7,8\}$, and $\{8,1\}$

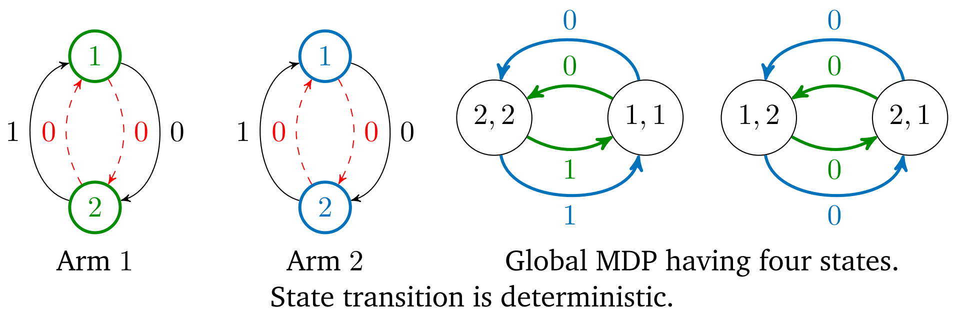

1
2
1
0
0
0
Arm 1
1
2
1
0
0
0
Arm 2
2, 2
1, 1
0
0
1
1
1, 2
2, 1
0
0
0
0
Global MDP having four states.
State transition is deterministic.

Figure 8.3.: A restless bandit with two arms and exactly one arm is activated at each decision time. The global state is denoted by (s_1, s_2), where s_1 is the state of Arm 1 and s_2 of Arm 2. The green arrows show the state transition when activating Arm 1. The blue arrows show the state transition when activating Arm 2. The numbers along the green and blue arrows show the expected reward when executing the actions.
For both arms, the black arrows show state transition of action activate, and the dashed red ones for action rest. The numbers along the black and dashed red arrows show the expected reward when executing the actions.

has $\{1,2\}$. This means that we can construct a cycle of transition by alternating between rest (R) and activate (A):

$$\{1,2\} \overset{R}{\to} \{2,3\} \overset{A}{\to} \{3,4\} \overset{R}{\to} \{4,5\} \overset{A}{\to} \{5,6\} \overset{R}{\to} \{6,7\} \overset{A}{\to} \{7,8\} \overset{R}{\to} \{8,1\} \overset{A}{\to} \{1,2\}.$$

With a proper synchronization between the states of Arm 1 and Arm 2, we can construct policies that induce two recurrent classes as presented in Figure 8.5. Indeed, consider the following arrangement.

$$\begin{aligned}
&\text{Arm 1}: \{1,2\} \overset{R}{\to} \{2,3\} \overset{A}{\to} \{3,4\} \overset{R}{\to} \dots \overset{A}{\to} \{1,2\}\\
&\text{Arm 2}: \{4,5\} \overset{A}{\to} \{5,6\} \overset{R}{\to} \{6,7\} \overset{A}{\to} \dots \overset{R}{\to} \{4,5\}\\
&\text{Bandit}: \begin{pmatrix}1,5\\2,4\\2,5\end{pmatrix} \overset{2}{\to} \begin{pmatrix}2,6\\3,5\\3,6\end{pmatrix} \overset{1}{\to} \begin{pmatrix}3,7\\4,6\\4,7\end{pmatrix} \overset{2}{\to} \dots \overset{1}{\to} \begin{pmatrix}1,5\\2,4\\2,5\end{pmatrix},
\end{aligned} \tag{8.3}$$

where the symbol $\overset{i}{\to}$ in Equation (8.3) means the transition when Arm i is activated. It means that any policies that activate Arm 2 for any odd $s \in [8]$, and rest Arm 2 for any even $s \in [8]$ when the bandit is in state $(s, s+4)$, $(s+1, s+3)$, and $(s+1, s+4)$ induce a recurrent class $\mathcal{X}^1$ as given in Figure 8.5 (note that for s big enough, state 9 is state 1, 10 is 2, etc.). The recurrent class $\mathcal{X}^2$ in Figure 8.5 is constructed by simply changing the sequence of Arm 2's state in the arrangement above and adapting the state of the bandit accordingly:

$$\text{Arm 1}: \{1,2\} \overset{R}{\to} \{2,3\} \overset{A}{\to} \{3,4\} \overset{R}{\to} \dots \overset{A}{\to} \{1,2\}$$

$$\text{Arm 2}: \{8,1\} \overset{A}{\to} \{1,2\} \overset{R}{\to} \{2,3\} \overset{A}{\to} \dots \overset{R}{\to} \{8,1\}$$

$$\text{Bandit}: \begin{pmatrix} 1,1 \\ 2,8 \\ 2,1 \end{pmatrix} \overset{2}{\to} \begin{pmatrix} 2,2 \\ 3,1 \\ 3,2 \end{pmatrix} \overset{1}{\to} \begin{pmatrix} 3,3 \\ 4,2 \\ 4,3 \end{pmatrix} \overset{2}{\to} \dots \overset{1}{\to} \begin{pmatrix} 1,1 \\ 2,8 \\ 2,1 \end{pmatrix}$$

So, any policies that activate Arm 2 for any odd $s \in [8]$, and rest Arm 2 for any even $s \in [8]$ when the bandit is in state $(s,s), (s+1,s+7)$, and $(s+1,s)$ induce the class $\mathcal{X}^2$. All in all, any policies that activate Arm 2 for any odd $s \in [8]$, and rest Arm 2 for any even $s \in [8]$ when the bandit is in state $(s,s), (s+1,s+7), (s+1,s), (s,s+4), (s+1,s+3)$ and $(s+1,s+4)$ induce two recurrent classes $\mathcal{X}^1$ and $\mathcal{X}^2$. That concludes the proof.

Current state	Next state when rest	Next state when activate
1	2 or 3	1 or 2
2	2 or 3	3 or 4
3	4 or 5	3 or 4
4	4 or 5	5 or 6
5	6 or 7	5 or 6
6	6 or 7	7 or 8
7	8 or 1	7 or 8
8	8 or 1	1 or 2

Table 8.1.: State transition of arms in Figure 8.4. Each transition happens with probability 0.5

$\square$

Theorem 8.3 means that even though all arms are ergodic, the corresponding bandit is not necessarily ergodic, and its mixing time can be infinite. This result is "not very intuitive", and it is crucial because assuming that all arms are ergodic is a popular condition in learning restless bandits (see e.g., [Ort+12; JAT19]).

To go further, the following theorem will show that even with an additional assumption that the bandit is also ergodic, the global mixing time can still be as large as we want. That is, we show that the MDP properties of a bandit, such as the diameter, the mixing time, and the span of the global bias function can be arbitrarily large.

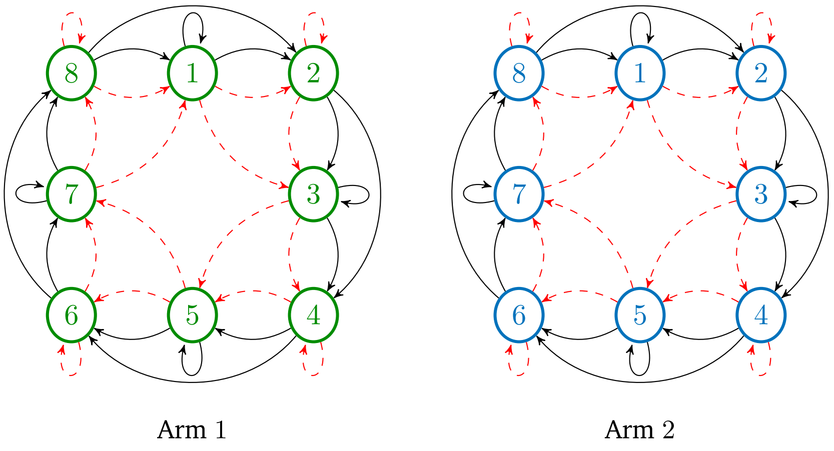


Figure 8.4.: Two restless Markovian arms, each having 8 states. Both arms are identical in terms of transition structure. The black arrows show state transition of action activate, and the dashed red ones for action rest. To ease the exposition, the expected reward is not shown because it is not relevant to the analysis. Also, each state transition happens with probability 0.5. We provide a list of state transitions in Table 8.1. The restless bandit having these two arms and exactly one arm is activated at each decision time is multichain because some policies induce two classes of recurrent states as shown in Figure 8.5.

Theorem 8.4 (MDP properties of restless bandit)

(i) *There is a restless bandit having n 2-state arms, all of whom have a diameter of size 2, that is ergodic and has a diameter of size 2^n.*

(ii) *There is a constant $c > 1$ such that for each constant $C > 1$, there is a restless bandit whose arms are ergodic and have a local mixing time smaller than c that is ergodic and has a global mixing time larger than C.*

(iii) *There is a constant $c > 0$ such that for each constant $C > 0$, there is a restless bandit whose arms have a span of local bias function under any policy upper bounded by c, that is communicating and has a span of global bias function lower bounded by C.*

Proof. **Proof of (i)** – consider a bandit having n arms, each arm is an MDP with state space $\{1, 2\}$ and has 0.5 probability of changing state under both actions. It should be clear that the bandit is ergodic. Indeed, any state of bandit can be reached from any other global states with probability $(1/2)^n$. Hence, the diameter of the bandit is 2^n.

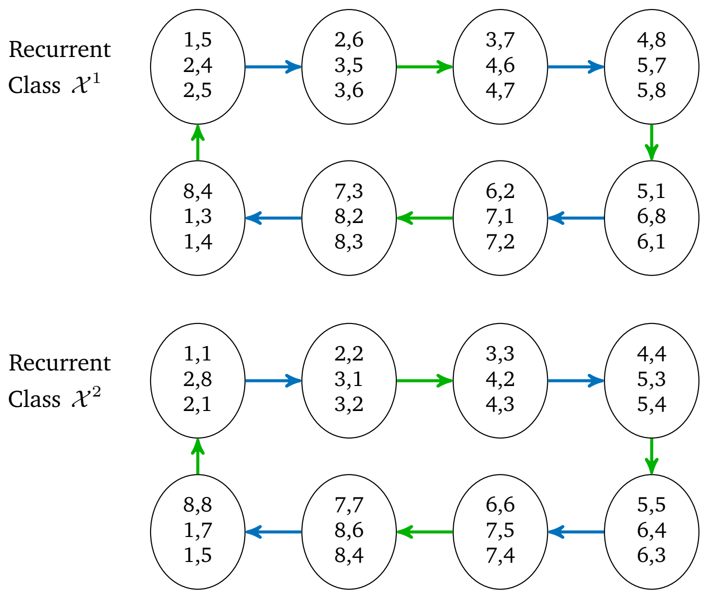


Figure 8.5.: State transition within two recurrent classes of a restless bandit with 2 arms presented in Figure 8.4. The first recurrent class is $\mathcal{X}^1 := \{(s, s{+}4), (s{+}1, s{+}3), (s{+}1, s{+}4) : s \in [8]\}$, and the second one is $\mathcal{X}^2 := \{(s, s), (s{+}1, s), (s{+}1, s{+}7) : s \in [8]\}$. These are two recurrent classes under any policies that activate Arm 2 for any odd $s \in [8]$, and rest Arm 2 for any even $s \in [8]$ when the bandit is in states $(s, s), (s, s+4), (s+1, s), (s+1, s+3), (s+1, s+4), (s+1, s+7)$. The blue arrows show the state transitions when Arm 2 is activated, and the green ones show the one when Arm 1 is activated. Each ellipse regroups the states having the same state transition under the same action. Note that each ellipse can transition to itself like in $\mathcal{X}^1$ state $(2, 4)$ can transition to state $(2, 5)$, but it is not shown. The rest of the states $\mathcal{X} \setminus (\mathcal{X}^1 \cup \mathcal{X}^2)$ are transient.

Proof of (ii) – Let p_1 and p_2 be the state transition functions of Arm 1 and Arm 2 given in Figure 8.4. p_1 and p_2 are both ergodic and induce the same state transition, which is given in Table 8.1. Suppose that the local mixing time of p_1 and p_2 is $t_{mix} > 1$. Note that we can specify p_1 and p_2 such that t_{mix} is small. Let's denote the bandit having two arms p_1 and p_2 by M. By Theorem 8.3, M is multichain. So, by Definition 8.1, the mixing time of M is infinite.

Now, consider two 8-state restless arms having transition functions u_1 and u_2, where u_1 and u_2 induce the same state transition: any local state $s \in [8]$ transitions to any other local state $s' \in [8]$ with probability $\frac{1}{8}$ under either action rest or activate. It is then clear that u_1 and u_2 are both ergodic and have a local mixing time of size 1. Any bandit having two arms u_1 and u_2 is an ergodic MDP.

For any $\varepsilon \in (0, 1)$, consider two 8-state restless arms having transition function $p'_1 = (1-\varepsilon)p_1 + \varepsilon u_1$ and $p'_2 = (1-\varepsilon)p_2 + \varepsilon u_2$. That is, at each state transition, a two-face coin whose head probability is ε is tossed. If the coin heads up, the arm evolves like u. Otherwise, the arm evolves like p. So, p'_1 and p'_2 are both ergodic. We know that if $\varepsilon = 0$, then the local mixing time of p'_1 and p'_2 is t_{mix}, and if $\varepsilon = 1$, then the mixing time is 1. Then, there exists $\varepsilon_0 > 0$ such that the local mixing time of p'_1 and p'_2 is $t_{mix}/2$. For any $\varepsilon \in (0, \varepsilon_0]$, any bandit having exactly two arms p'_1 and p'_2 is an ergodic MDP denoted by $M'(\varepsilon)$. The global mixing time of $M'(\varepsilon)$ is then defined. However, when $\varepsilon \to 0$, $M'(\varepsilon) \to M$. This is equivalent to say that when $\varepsilon \to 0$, the mixing time of $M'(\varepsilon)$ tends to infinity.

Proof of (iii) – We use the same technique in the proof of (ii). First, consider the bandit having 4 global states as in Figure 8.3. We denote this bandit by M. We denote the arms' transition functions by p_1 and p_2 and reward functions by r_1 and r_2. It should be clear that for each arm, the span of local bias under any policy is bounded in $[0, 1]$, but the bandit M is multichain. So, the span of global bias function is infinite.

Consider now two 2-state arms with reward functions r_1 and r_2, and transition functions u_1 and u_2 that induce the same state transition: any local state $s \in [2]$ transitions to any other local state $s' \in [2]$ with probability $\frac{1}{2}$. So, u_1 and u_2 are both ergodic, and the span of local bias function is bounded in $[0, 1]$.

With the same technique above, for any $\varepsilon \in (0, 1)$, consider two 2-state arms having reward functions r_1 and r_2, and transition functions $p'_1 = (1-\varepsilon)p_1 + \varepsilon u_1$ and $p'_2 = (1-\varepsilon)p_2 + \varepsilon u_2$. So, for any $\varepsilon \in (0, 1)$, p'_1 and p'_2 are both ergodic and have the span of local bias function bounded in $[0, 1]$. We denote any bandit having exactly two arms (r_1, p'_1) and (r_2, p'_2) by $M'(\varepsilon)$. We will see in Theorem 8.5 that for any $\varepsilon \in (0,1)$, $M'(\varepsilon)$ is a communicating MDP. However, when $\varepsilon \to 0$, $M'(\varepsilon) \to M$. This is equivalent to say that when $\varepsilon \to 0$, the span of global bias function of $M'(\varepsilon)$ tends to infinity. $\square$

With Theorem 8.4, the curse of dimensionality in restless bandits manifests not only in the state size but also in the MDP properties of the bandit. Hence, it is equally important to take into account the dependency between the number of arms and the MDP properties of the restless bandits when deriving the regret bound of RL algorithms.

8.4.2 Positive results

This section shows a few results in which the assumption on local arms implies "desirable" properties on the bandit. The term "desirable" refers to the fact that the properties frequently required by the general-purpose RL algorithms are satisfied. This term does not imply that the learning algorithm has a regret bound sublinear in the total steps nor that its regret bound is explicitly polynomially in the number of arms.

Theorem 8.5 (Communicating restless bandit)
A restless bandit whose arms are ergodic is a communicating MDP.

Proof. We prove the theorem by its contra position. Assume that a given bandit is not communicating (either weakly communicating or not weakly communicating). Then, in this bandit, there are global states that are not reachable from other states. This implies two possibilities: (1) for some arms, some local states are not recurrent, (2) all arms are recurrent but periodic. Each possibility implies that there exists at least one arm that is not ergodic. $\square$

This theorem means that if all arms are ergodic, then the bandit has a finite diameter. The following theorem also provides a positive result on the structure of bandit.

Theorem 8.6 (Unichain restless bandit)
A restless bandit whose arms are unichain with at least one local state reachable in one step from any other local states under both rest and activate actions is

unichain with at least one global state reachable in one step from any other global states under any policies.

Proof. Consider the bandit M described in Section 8.3. For each arm $i \in [n]$, let z_i be the state that is reachable in one time step from any other states under both rest and activate actions. We have that $P^{a_i}(s_i, z_i) > 0$ for all $i \in [n], x_i \in \mathcal{S}_i, a_i \in \{0, 1\}$. For any global action $\boldsymbol{a} \in \mathcal{A}(m)$, any global states $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{X}$, the bandit M's state transition is given by $p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) = \prod_{i=1}^{n} P^{a_i}(s_i, s_i')$. Then, for any action $\boldsymbol{a} \in \mathcal{A}(m)$, any state $\boldsymbol{s} \in \mathcal{X}$, we have

$$p(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{a}) = \prod_{i=1}^{n} P^{a_i}(s_i, z_i) > 0,$$

because n is finite and for any i, $P^{a_i}(s_i, z_i) > 0$. Hence, the global state $\boldsymbol{z}$ is reachable from any other global states in one step under any policies. In consequence, no policies induce a global Markov chain that has multiple closed irreducible recurrent classes. That concludes the proof. $\square$

Theorem 8.6 means that if the ergodicity coefficient of all arms is strictly smaller than 1, then the ergodicity coefficient of the corresponding bandit is also strictly smaller than 1. As we mentioned above, the ergodicity coefficient is an MDP parameter that provides an upper bound on the MDP's mixing time and span of bias function. We will see this idea when we discuss the regret analysis of RB-TSDE [AM22] with a precise definition of the ergodicity coefficient (see Equation (8.5)) in the sections below.

8.5 Learning algorithms for restless bandits

8.5.1 Discussion on using Whittle index policy as the baseline in regret definition

In Chapter 7, we use Gittins index policy as the baseline policy for evaluating the regret of learning algorithms in discounted rested bandit because this index policy is optimal. Yet, Whittle index policy is only asymptotically optimal under some technical conditions [WW90] and suboptimal in general. This brings two options to the discussion.

(a) Shall we compare the algorithm to an oracle that knows an optimal policy for any restless bandit?

(b) Shall we use Whittle index policy as a baseline policy, knowing that it is generally suboptimal?

Assuming an oracle is what is done in [Ort+12; JAT19; WHL20] when learning in restless bandits and in [OV14; RM20; XT20] when learning in factored-MDPs. The authors are then solely interested in the statistical aspect of the learning algorithms.

We now focus on the second option (b). First, if the bandit is indexable, then Whittle index policy is defined. Also, regret definition requires that the gain of baseline policy is state-independent. So, a possible assumption to use Whittle index policy as the baseline policy is that the unknown bandit is indexable and Whittle index policy is unichain, i.e., the Markov chain induced by Whittle index policy has a single recurrent class. This discussion also extends to the learning approaches. For OFU methods, we have shown in Chapter 7 that optimistic algorithms that apply confidence bonus on arm's transition cannot leverage index policy to choose an imagined bandit M^k with index policy π^k that guarantees the optimism at episode k. Worse yet, there is no guarantee that the optimistic restless bandit M^k is indexable, let alone the optimism. For posterior sampling, we need the imagined bandit M^k to be indexable and the corresponding Whittle index policy π^k to induce state-independent gain. This can be done by making an assumption on the support of the prior distribution. That is, any bandit drawn from the support of the prior distribution is indexable and admits unichain Whittle index policy. We will see in the following that this is what is done in the work of [AM22].

8.5.2 Algorithms with regret guarantee

Similarly to what we have seen in Chapter 7, the current best general-purpose RL algorithms have a regret bound $\tilde{O}\Big(\sqrt{HS^n\binom{n}{m}T}\Big)$ in the restless bandit M described in Section 8.3, where H is the upper bound on the span of the global optimal bias function, and S^n and $\binom{n}{m}$ are the state and action sizes of the bandit M. There are two issues in this regret bound.

(a) One apparent problem is the term S^n that is the state size of M. This problem is resolved in the work of [Ort+12; JAT19; AM22].

(b) The other problem is the implicit dependency between the upper bound H and the number of arms n. As we have seen in Theorem 8.4, a bandit may have the span of global bias function as large as we want.

In the following, we provide a sketch of proof of the regret bound of RB-TSDE [AM22], a modified version of TSDE [Ouy+17] to restless Markovian bandit, and a discussion on their result.

Overview of regret analysis of RB-TSDE [AM22]

In this section, we start by presenting how RB-TSDE [AM22] works and the assumptions needed in [AM22]. Next, we provide an overview of regret analysis of RB-TSDE under the assumptions. Finaly, we finish the section with a discussion of their result.

RB-TSDE is a Bayesian learning algorithm for restless Markovian bandits with the average reward criterion. Extended from TSDE [Ouy+17], RB-TSDE updates its policy episodically as the following. Let t^k be the time step when episode k begins with the convention $t^1 := 1$. For each arm $i \in [n]$, let $N_t(s_i, a_i)$ be the number of times up to time step t that the learner executes action a_i when arm i is in state s_i. RB-TSDE terminates episode $k \geq 1$ at time step $t > t^k$ and updates its policy if

$$t - t^{k-1} > 2t^k \text{ or } N_t(s_i, a_i) > 2N_{t^k}(s_i, a_i) \text{ for some } (s_i, a_i). \tag{8.4}$$

RB-TSDE [AM22] exploits the structure of restless bandit by choosing a prior distribution ϕ_i for each unknown arm $\langle \mathcal{S}_i, \{0,1\}, \{\boldsymbol{r}_i^0, \boldsymbol{r}_i^1\}, \{\boldsymbol{P}_i^0, \boldsymbol{P}_i^1\} \rangle$ before the learning. At time step t^k, RB-TSDE uses the collected observations o_{t^k} to derive a posterior $\phi_i(\cdot \mid o_{t^k})$, and draws a sample $(\{\boldsymbol{r}_i^{0k}, \boldsymbol{r}_i^{1k}\}, \{\boldsymbol{P}_i^{0k}, \boldsymbol{P}_i^{1k}\})$ according to $\phi_i(\cdot \mid o_{t^k})$ for each arm i. The samples $\{(\{\boldsymbol{r}_i^{0k}, \boldsymbol{r}_i^{1k}\}, \{\boldsymbol{P}_i^{0k}, \boldsymbol{P}_i^{1k}\})\}_{i \in [n]}$ is then used to compute Whittle index policy π^k.

RB-TSDE [AM22] computes Whittle index policy π^k of the sampled bandit M^k because Whittle index policy of the unknown bandit M, denoted by π^λ, is used as the baseline policy for evaluating the regret of RB-TSDE. So, the work of [AM22] assumes three conditions: Let $\mathrm{supp}(\phi_i)$ be the support of prior distribution ϕ_i for each arm i. Let ϕ be the joint prior distibution of all arms and $\mathrm{supp}(\phi) := \bigotimes_{i=1}^n \mathrm{supp}(\phi_i)$ be the support of the joint prior ϕ.

Assumption 8.7

For any $i \in [n]$ and $(\{\boldsymbol{r}_i^{0'}, \boldsymbol{r}_i^{1'}\}, \{\boldsymbol{P}_i^{0'}, \boldsymbol{P}_i^{1'}\}) \in \operatorname{supp}(\phi_i)$, the arm $\langle \mathcal{S}_i, \{0,1\}, \{\boldsymbol{r}_i^{0'}, \boldsymbol{r}_i^{1'}\}, \{\boldsymbol{P}_i^{0'}, \boldsymbol{P}_i^{1'}\}\rangle$ is indexable.

Assumption 8.8

Any bandit $M' \in \operatorname{supp}(\phi)$ admits Whittle index policy π' that is unichain.

Assumption 8.9

There exists $\beta^* < 1$ such that any bandit $M' \in \operatorname{supp}(\phi)$ admits an ergodicity coefficient $\beta_{M'} \leq \beta^*$ where

$$\beta_{M'} = 1 - \min_{\substack{\boldsymbol{s},\boldsymbol{s}' \in \mathcal{X} \\ \boldsymbol{a},\boldsymbol{a}' \in \mathcal{A}(m)}} \sum_{\boldsymbol{z} \in \mathcal{X}} \min\{p'(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{a}), p'(\boldsymbol{z} \mid \boldsymbol{s}', \boldsymbol{a}')\}. \tag{8.5}$$

The reasons why Assumptions 8.7 and 8.8 are needed are already discussed in Section 8.5.1. Assumption 8.9 allows us to bound the span of global bias function in a function of β^*. In addition, we should mention that by our Theorem 8.6, Assumption 8.9 implies Assumption 8.8.

Assume that the unknown bandit M is drawn from $\operatorname{supp}(\phi)$. Under Assumptions 8.7 and 8.8, Whittle index policy π^λ is well-defined and induces a state-independent gain g_M^λ in the unknown bandit M. Similarly to Definition 3.2, the regret of RB-TSDE after T steps is given by

$$\text{Regret}(\text{RB-TSDE}, M, T) := T g_M^\lambda - \sum_{t=1}^{T} \sum_{i=1}^{n} r_{t,i}. \tag{8.6}$$

Its Bayesian regret is given by (3.5).

Now, we focus on bounding the regret of RB-TSDE. Let K_T be the total number of episodes up to time T. Then,

$$\begin{aligned}\text{Regret}(\text{RB-TSDE}, M, T) = \sum_{k=1}^{K_T} \sum_{t=t^k}^{t^{k+1}-1} \Big(g_M^\lambda - \sum_{i=1}^{n} r^{a_{t,i}}(s_{t,i}) \Big) \\ + \underbrace{\sum_{i=1}^{n} \Big(r^{a_{t,i}}(s_{t,i}) - r_{t,i} \Big)}_{\Delta_{t,0}}\end{aligned} \tag{8.7}$$

Recall that π^k is the Whittle index policy of the imagined bandit M^k. Under Assumption 8.7, π^k is well-defined, and under Assumption 8.8, the gain of π^k is state-independent. Then, the Bellman evaluation equation for π^k in M^k gives: for global state $\boldsymbol{s}_t \in \mathcal{X}$, $\boldsymbol{a}_t = \pi^k(\boldsymbol{s}_t)$,

$$0 = \sum_{i=1}^{n} r^{a_{t,i}k}(s_{t,i}) - g^{\pi^k}_{M^k} + \sum_{\boldsymbol{s}' \in \mathcal{X}} p^k(\boldsymbol{s}' \mid \boldsymbol{s}_t, \boldsymbol{a}_t) h^{\pi^k}_{M^k}(\boldsymbol{s}') - h^{\pi^k}_{M^k}(\boldsymbol{s}_t). \tag{8.8}$$

Then, adding Equation (8.8), $0 = \sum_{\boldsymbol{s}' \in \mathcal{X}} p(\boldsymbol{s}' \mid \boldsymbol{s}_t, \boldsymbol{a}_t)\Big(h^{\pi^k}_{M^k}(\boldsymbol{s}') - h^{\pi^k}_{M^k}(\boldsymbol{s}')\Big)$, and $0 = h^{\pi^k}_{M^k}(\boldsymbol{s}_{t+1}) - h^{\pi^k}_{M^k}(\boldsymbol{s}_{t+1})$ to Equation (8.7), and regrouping terms give

$$\begin{aligned}
\text{Regret}(\text{RB-TSDE}, M, T) = \sum_{k=1}^{K_T} \sum_{t=t^k}^{t^{k+1}-1} & \underbrace{(g^{\lambda}_{M} - g^{\pi^k}_{M^k})}_{\Delta_{t,1}} + \underbrace{\sum_{i=1}^{n}\Big(r^{a_{t,i}k}(s_{t,i}) - r^{a_{t,i}}(s_{t,i})\Big)}_{\Delta_{t,2}} \\
& + \underbrace{\sum_{\boldsymbol{s}' \in \mathcal{X}} \Big(p^k(\boldsymbol{s}' \mid \boldsymbol{s}_t, \boldsymbol{a}_t) - p(\boldsymbol{s}' \mid \boldsymbol{s}_t, \boldsymbol{a}_t)\Big) h^{\pi^k}_{M^k}(\boldsymbol{s}')}_{\Delta_{t,3}} + \underbrace{h^{\pi^k}_{M^k}(\boldsymbol{s}_{t+1}) - h^{\pi^k}_{M^k}(\boldsymbol{s}_t)}_{\Delta_{t,4}} \\
& + \underbrace{\sum_{\boldsymbol{s}' \in \mathcal{X}} p(\boldsymbol{s}' \mid \boldsymbol{s}_t, \boldsymbol{a}_t) h^{\pi^k}_{M^k}(\boldsymbol{s}') - h^{\pi^k}_{M^k}(\boldsymbol{s}_{t+1})}_{\Delta_{t,5}} + \Delta_{t,0}.
\end{aligned}$$

The six quantities are bounded as the following.

- The terms $\Delta_{t,0}$ and $\Delta_{t,5}$ are martingale difference terms whose expected value is zero.
- The expected value of $\sum_{k=1}^{K_T} \sum_{t=t^k}^{t^{k+1}-1} \Delta_{t,1}$ is bounded by $r_{max} K_T$ [AM22, Appendix A.2]
- The term $\Delta_{t,2}$ is bounded by Hoeffding's inequality (or simply zero if the rewards are deterministic).
- The telescopic sum $\sum_{k=1}^{K_T} \sum_{t=t^k}^{t^{k+1}-1} \Delta_{t,4}$ is bounded by $\sum_{k=1}^{K_T} sp(\boldsymbol{h}^{\pi^k}_{M^k})$.

The term $\Delta_{t,3}$ is bounded by $sp(\boldsymbol{h}^{\pi^k}_{M^k}) \left\| \boldsymbol{p}^k(\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t) - \boldsymbol{p}(\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t) \right\|_{\ell_1}$. In general MDPs, the concentration of global transition $\sum_{\boldsymbol{s}' \in \mathcal{X}} \left| p^k(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) - p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \right|$ is bounded proportionally to $\sqrt{|\mathcal{X}|}$ (see [Qia+20]). However, restless bandit is a well-structure MDP. The work of [AM22] exploits this structure as the following: For any global state $\boldsymbol{s} \in \mathcal{X}$ and global action $\boldsymbol{a} \in \mathcal{A}(m)$, it follows from [JAT19, Lemma 13] that

$$\sum_{\boldsymbol{s}' \in \mathcal{X}} \left| p^k(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) - p(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) \right| \le \sum_{i=1}^{n} \sum_{s'_i \in \mathcal{S}_i} \left| P^{a_i k}(s_i, s'_i) - P^{a_i}(s_i, s'_i) \right|,$$

where $\sum_{s'_i \in \mathcal{S}_i} \left| P^{a_i k}(s_i, s'_i) - P^{a_i}(s_i, s'_i) \right|$ is the concentration of the local transition, which is bounded proportionally to $\sqrt{S}$ using Weissman's inequality.

Using the doubling trick (8.4), [AM22, Lemma A.1] gives $K_T \leq 2\sqrt{nST\ln(T)}$. So, what is left to bound is the span of the global bias function $sp(\boldsymbol{h}^{\pi^k}_{M^k})$. By [AM22, Lemma 5.1], for any $k \geq 1$, $sp(\boldsymbol{h}^{\pi^k}_{M^k}) \leq \dfrac{2r_{max}}{1-\beta^*}$.

In summary, RB-TSDE [AM22] enjoys the following Bayesian regret bound.

Proposition 8.10 ([AM22, Theorem 4.1])
Assume that $M \in \operatorname{supp}(\phi)$. Under Assumptions 8.7-8.9,

$$\text{BayesRegret}(\text{RB-TSDE}, \phi, T) < 40\frac{r_{max}}{1-\beta^*} nS\sqrt{T\ln(T)} \tag{8.9}$$

This result presents an exponential improvement compared to the current best general-purpose RL algorithms whose regret bound is $\tilde{O}\left(\sqrt{HS^n\binom{n}{m}T}\right)$.

However, the upper bound on the span of global bias function $H := \dfrac{2r_{max}}{1-\beta^*}$ in (8.9) has no explicit dependency with the number of arms n.

In the following section, we provide our last result that shows the dependency between β^* and n under Assumption 8.9.

Local and global ergodicity coefficients in restless Markovian bandit

Consider the bandit M described in Section 8.3.

Theorem 8.11 (Local and global ergodicity coefficients)
For each arm $i \in [n]$, the ergodicity coefficient γ_i of the arm is defined by

$$\gamma_i = 1 - \min_{\substack{s_i, s'_i \in \mathcal{S}_i \\ a, a' \in \{0,1\}}} \sum_{z_i \in \mathcal{S}_i} \min\{P^a(s_i, z_i), P^{a'}(s'_i, z_i)\}.$$

Similarly, the ergodicity coefficient β_M of the bandit M is defined in the same manner as (8.5).
If for any arm i, $\gamma_i < 1$, then $\beta_M < 1$.
Moreover, if there exists $\varepsilon > 0$ such that for any arm $i \in [n]$,

$$\gamma_i \leq 1 - \varepsilon, \tag{8.10}$$

then $\beta_M \le 1 - \varepsilon^n$.

Proof. For any two global states $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{X}$ and any two actions $\boldsymbol{a}, \boldsymbol{a}' \in \mathcal{A}(m)$, we have

$$\begin{aligned}\sum_{\boldsymbol{z}\in\mathcal{X}} \min\{p(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{a}), p(\boldsymbol{z} \mid \boldsymbol{s}', \boldsymbol{a}')\} &= \sum_{\boldsymbol{z}\in\mathcal{X}} \min\left\{\prod_{i=1}^{n} P^{a_i}(s_i, z_i), \prod_{i=1}^{n} P^{a'_i}(s'_i, z_i)\right\} \\ &\ge \sum_{\boldsymbol{z}\in\mathcal{X}} \prod_{i=1}^{n} \min\left\{P^{a_i}(s_i, z_i), P^{a'_i}(s'_i, z_i)\right\} \\ &= \prod_{i=1}^{n}\left(\sum_{z_i\in\mathcal{S}_i} \min\left\{P^{a_i}(s_i, z_i), P^{a'_i}(s'_i, z_i)\right\}\right).\end{aligned}$$

Since for any $i \in [n]$, $\gamma_i < 1$, we have that

$$\begin{aligned}\sum_{z_i\in\mathcal{S}_i} \min\{P^{a_i}(s_i, z_i), P^{a'_i}(s'_i, z_i)\} &\ge \min_{\substack{x_i,y_i\in\mathcal{S}_i \\ a,a'\in\{0,1\}}} \sum_{z_i\in\mathcal{S}_i} \min\{P^{a}(s_i, z_i), P^{a'}(s'_i, z_i)\} \\ &= 1 - \gamma_i > 0.\end{aligned}$$

In consequences, $\sum_{\boldsymbol{z}\in\mathcal{X}} \min\{p(\boldsymbol{z} \mid \boldsymbol{s}, \boldsymbol{a}), p(\boldsymbol{z} \mid \boldsymbol{s}', \boldsymbol{a}')\}{>}0$. We conclude that $\beta_M{<}1$. Moreover, if there exists $\varepsilon > 0$ such that (8.10) holds for any $i \in [n]$, then it follows that

$$\prod_{i=1}^{n}\left(\sum_{z_i\in\mathcal{S}_i} \min\{P^{a_i}(s_i, z_i), P^{a'_i}(s'_i, z_i)\}\right) \ge \varepsilon^n.$$

We can conclude that $\beta_M \le 1 - \varepsilon^n$ using its definition. □

Assumption 8.9 is equivalent to saying that there exists $\varepsilon > 0$ such that any bandit $M' \in \operatorname{supp}(\phi)$ satisfies the condition (8.10) of Theorem 8.11, and $\beta^* = 1 - \varepsilon^n$. The latter implies that $H := \dfrac{2r_{max}}{1-\beta^*} = 2r_{max}\left(\dfrac{1}{\varepsilon}\right)^n$. Therefore, (8.9) remains exponential in n. However, this certainly does not cancel the exponential improvement in the regret bound of RB-TSDE [AM22] over the general-purpose RL algorithms in learning restless Markovian bandit problems.

8.6 Conclusion

In this chapter, we provide different examples that show that defining a subclass of restless bandits with desirable global properties by only making assumptions about arms is difficult. Our examples also show that no RL algorithms can perform uniformly well over the general class of restless bandits whose arms are unichain. Furthermore, a restless bandit can be multichain even though its arms are ergodic. We also present a few positive results, such as a restless bandit is communicating if its arms are ergodic. Finally, we discuss the requirements when using Whittle index policy as the baseline policy in regret definition for learning restless bandits and provide an overview of regret analysis of RB-TSDE [AM22], a modified version of TSDE [Ouy+17], whose regret bound provides an exponential improvement over the current best general-purpose RL algorithms.

We believe that it is challenging to derive an upper bound on the span of global bias function or the global diameter of restless bandit that is polynomially in the number of arms. In fact, it is impossible to do so for general restless bandits as we have seen in this chapter that the MDP properties of some restless bandits, such as the global diameter, or the span of global bias function can be exponential in the number of arms. This calls for restrictive assumptions on bandits to design an algorithm with a regret bound explicitly polynomially in the number of arms, similar to the work of [WHL20]. This also calls for model-free algorithms that directly learn Whittle index of the unknown restless bandit in the direction of [Fu+19; GJN21; Nak+21; AB22].

9 Conclusions and Future Work

9.1 Conclusions

This thesis addresses two grand questions in Markovian bandits: (1) index computation given the bandit's parameters and (2) minimizing the regret and the runtime of learning algorithms using the problem structure and index policy when the bandit's parameters are unknown. For the former, we have introduced an algorithm for computing Whittle or Gittins indices of a Markovian arm in subcubic time complexity. For the latter, we have proposed three learning algorithms with a regret guarantee sublinear in the number of arms for rested bandits with discount. Moreover, two of the three learning algorithms can leverage Gittins index to have a runtime linear in the number of arms. We have also pointed out the difficulties of minimizing the regret when learning in restless bandits with average reward criterion.

We have seen that the structure of arms plays a crucial role in Markovian bandit problems. First, the Whittle index policy is defined for restless Markovian bandits whose arms satisfy a notion of indexability, but this notion becomes unclear when some arms admit some local optimal policies that are multichain. Lastly, when learning in restless bandits, the MDP properties of the bandit, such as the diameter, the mixing time, and the span of the global bias functions, depend heavily on the arms' structure.

This thesis also provides an argument that supports the power of Bayesian algorithms: They can be easily tailored to the structure of the problem to learn. For instance, MB-PSRL has a regret guarantee and a runtime scalable in the number of arms when learning in rested Markovian bandits with discount. Also, RB-TSDE [AM22] can leverage the Whittle index to have a scalable runtime when learning in restless Markovian bandits with the average reward criterion. Meanwhile, the optimistic algorithms that use confidence bonuses on the arms' state transitions likely have a runtime non-scalable in the number of arms in learning Markovian bandits.

9.2 Future work

There are several directions to extend the work developed in this thesis. Some of them are outlined in the following.

Computing the Whittle index of arms with a sparse transition structure. The possibility of designing a Whittle index computation algorithm in restless Markovian arms with sparse transition matrix warrants further investigation. Many applications in which the Whittle index policy performs exceptionally well admit a sparse arm's transition structure (see, e.g., [WM95; Nin02; AL18], also [WHL20] and references therein). On the basis of our index computation algorithm, one may want to investigate the combination of the sparse matrix inversion (see e.g., [DM62; NR83]) with the Sherman-Morrison-Woodbury formula. In this direction, the work of [Van91] investigates how the Sherman-Morrison-Woodbury formula can be used in the inversion of a sparse matrix with dense columns. It would be exciting to adapt this work to our algorithm.

Upper bounds on the span of the global bias functions of restless bandit and the number of arms. The ergodicity coefficient of the bandit is used in the recent work of [AM22] to upper bound the span of the global bias functions when learning in restless bandits with the average reward criterion. Nevertheless, we still do not know the optimal upper bound on this span in terms of the number of arms. Our Theorem 8.11 shows an exponential dependency between an upper bound on the ergodicity coefficient of the bandit and the number of arms. However, we believe that this upper bound is not tight for the span of the global bias functions. Indeed, we have performed a few numerical experiments that advocate a linear dependency between an upper bound on this span and the number of arms when the condition of Theorem 8.11 is satisfied. Therefore, we conjecture that the span of the global bias functions is upper bounded linearly in the number of arms as the following.

Conjecture 9.1
Consider a restless bandit M having n arms. Each arm $\langle \mathcal{S}_i, \{0,1\}, \{\boldsymbol{r}_i^0, \boldsymbol{r}_i^1\}, \{\boldsymbol{P}_i^0, \boldsymbol{P}_i^1\}\rangle$ is an MDP with a finite state space $\mathcal{S}_i$ and a binary action space $\{0,1\}$. For each arm $i \in [n]$, let γ_i be its ergodicity coefficient defined by

$$\gamma_i = 1 - \min_{\substack{s_i, s_i' \in \mathcal{S}_i \\ a,a' \in \{0,1\}}} \sum_{z_i \in \mathcal{S}_i} \min\{P^a(s_i, z_i), P^{a'}(s_i', z_i)\}.$$

Let $\boldsymbol{h}^*$ be the global optimal bias function of the bandit M in the average reward criterion. We conjecture that

$$sp(\boldsymbol{h}^*) \leq \sum_{i=1}^{n} \frac{sp(r_i)}{1-\gamma_i}.$$

Model-free learning algorithms for Markovian bandits. This thesis focuses on model-based learning algorithms for Markovian bandits. Another direction of research would be investigating model-free algorithms. For rested bandits with discount, the work of [Duf95] uses the Q-learning (QL) algorithm to estimate the Gittins index and softmax for the exploration. Similarly, the work of [AB22] uses a QL-based algorithm to estimate the Whittle index of undiscounted restless bandits. Meanwhile, the works of [Jin+18; Wei+20] derive QL-like algorithms with a regret guarantee when learning in generic MDPs. It would be interesting to investigate how these works can be connected.

Learning algorithms for restless multi-armed multi-action bandits. Restless multi-armed multi-action bandits (R(MA)^{2}B) generalize the restless Markovian bandits. The works of [GHK11; HG15] extend the notion of indexability to R(MA)^{2}B. However, only the subclass of R(MA)^{2}B with a special monotonic structure is analyzed in their work. In [GGY22b], a novel LP-update policy is proposed for the finite-horizon setting. When the number of arms grows to infinity, their policy is proven to achieve optimality. Moreover, their LP-update policy is applicable for general R(MA)^{2}Bs (i.e., the indexability condition is not required). A natural extension of our learning problem is to consider the case where the unknown environment is a R(MA)^{2}B. For model-free algorithms, the work of [Kil+21] proposes two algorithms: (i) QL-based algorithm to learn the index policy proposed by [GHK11] and (ii) Lagrange policy QL algorithm. It is interesting to see how Bayesian approach can be used in this learning problem.

List of publications

[GGK22] Nicolas Gast, Bruno Gaujal, and Kimang Khun. "Learning Algorithms for Markovian Bandits:Is Posterior Sampling more Scalable than Optimism?" In: *Transactions on Machine Learning Research* (2022). URL: `https://openreview.net/forum?id=Sh3RF9JowK`. cit. on p. 107

[GGK23] Nicolas Gast, Bruno Gaujal, and Kimang Khun. "Testing indexability and computing Whittle and Gittins index in subcubic time". In: *Mathematical Methods of Operations Research* (June 2023). DOI: `10.1007/s00186-023-00821-4`. URL: `https://doi.org/10.1007/s00186-023-00821-4`. cit. on pp. 47, 61, 109

Bibliography

[AAR09] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. "On the Gittins index in the M/G/1 queue". In: *Queueing Systems* 63.1-4 (2009), p. 437. cit. on p. 1

[AAR11] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. "Properties of the Gittins index with application to optimal scheduling". In: *Probability in the Engineering and Informational Sciences* 25.3 (2011), pp. 269–288. cit. on pp. 1, 64

[AB22] Konstantin E Avrachenkov and Vivek S Borkar. "Whittle index based Q-learning for restless bandits with average reward". In: *Automatica* 139 (2022), p. 110186. cit. on pp. 1, 39, 64, 111, 162, 183, 187

[ABB01] Jinane Abounadi, Dimitrib Bertsekas, and Vivek S Borkar. "Learning algorithms for Markov decision processes with average cost". In: *SIAM Journal on Control and Optimization* 40.3 (2001), pp. 681–698. cit. on p. 162

[ABG09] Thomas W Archibald, DP Black, and Kevin D Glazebrook. "Indexability and index heuristics for a simple class of inventory routing problems". In: *Operations research* 57.2 (2009), pp. 314–326. cit. on p. 1

[ACF02] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem". In: *Machine learning* 47.2 (2002), pp. 235–256. cit. on p. 111

[ACJ21] Priyank Agrawal, Jinglin Chen, and Nan Jiang. "Improved worst-case regret bounds for randomized least-squares value iteration". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 6566–6573. cit. on p. 117

[AGV21] Urtzi Ayesta, Manu K Gupta, and Ina Maria Verloop. “On the computation of Whittle’s index for Markovian restless bandits”. In: *Mathematical Methods of Operations Research* 93.1 (2021), pp. 179–208. cit. on p. 65

[AJ17] Shipra Agrawal and Randy Jia. “Posterior sampling for reinforcement learning: worst-case regret bounds”. In: *arXiv preprint arXiv:1705.07041* (2017). cit. on pp. 117, 138

[AL09] Sahand Haji Ali Ahmad and Mingyan Liu. “Multi-channel opportunistic access: A case of restless bandits with multiple plays”. In: *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2009, pp. 1361–1368. cit. on p. 36

[AL18] Samuli Aalto and Pasi Lassila. “Whittle index approach to energy-aware dispatching”. In: *2018 30th International Teletraffic Congress (ITC 30)*. Vol. 1. IEEE. 2018, pp. 19–27. cit. on p. 186

[ALT19] Samuli Aalto, Pasi Lassila, and Ianire Taboada. “Whittle index approach to opportunistic scheduling with partial channel information”. In: *Performance Evaluation* 136 (2019), p. 102052. cit. on p. 1

[AM19a] Nima Akbarzadeh and Aditya Mahajan. “Dynamic spectrum access under partial observations: A restless bandit approach”. In: *2019 16th Canadian Workshop on Information Theory (CWIT)*. IEEE. 2019, pp. 1–6. cit. on p. 36

[AM19b] Nima Akbarzadeh and Aditya Mahajan. “Restless bandits with controlled restarts: Indexability and computation of Whittle index”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE. 2019, pp. 7294–7300. cit. on pp. 1, 36, 64

[AM20] Nima Akbarzadeh and Aditya Mahajan. “Conditions for indexability of restless bandits and an O (Kˆ 3) algorithm to compute Whittle index”. In: *arXiv preprint arXiv:2008.06111* (2020). cit. on pp. 39, 42, 49, 62, 64, 85, 94, 97, 98, 99

[AM21] Nima Akbarzadeh and Aditya Mahajan. “Maintenance of a collection of machines under partial observability: Indexability and computation of Whittle index”. In: *arXiv preprint arXiv:2104.05151* (2021). cit. on p. 64

[AM22] Nima Akbarzadeh and Aditya Mahajan. “On learning Whittle index policy for restless bandits with scalable regret”. In: *arXiv preprint arXiv:2202.03463* (2022). cit. on pp. 159, 160, 161, 163, 176, 177, 178, 180, 181, 182, 183, 185, 186

[And+03] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. “An introduction to MCMC for machine learning”. In: *Machine learning* 50 (2003), pp. 5–43. cit. on p. 27

[Ans+03] PS Ansell, Kevin D Glazebrook, José Nino-Mora, and M O’Keeffe. “Whittle’s index policy for a multi-class queueing system with convex holding costs”. In: *Mathematical Methods of Operations Research* 57.1 (2003), pp. 21–39. cit. on pp. 1, 40, 162

[AOM17] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. “Minimax regret bounds for reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 263–272.
cit. on pp. 4, 20, 21, 22, 26, 27, 29, 31, 32, 33, 108, 117, 119, 120, 143, 144

[APZ18] Konstantin Avrachenkov, Alexei Piunovskiy, and Yi Zhang. “Impulsive control for G-AIMD dynamics with relaxed and hard constraints”. In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE. 2018, pp. 880–887. cit. on p. 1

[Avr+13] Konstantin Avrachenkov, Urtzi Ayesta, Josu Doncel, and Peter Jacko. “Congestion control of TCP flows in Internet routers by means of index policy”. In: *Computer Networks* 57.17 (2013), pp. 3463–3478. cit. on p. 1

[AVW87] Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. “Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-Part II: Markovian rewards”. In: *IEEE Transactions on Automatic Control* 32.11 (1987), pp. 977–982. cit. on p. 37

[BC12] Sébastien Bubeck and Nicolo Cesa-Bianchi. “Regret analysis of stochastic and nonstochastic multi-armed bandit problems”. In: *Foundations and Trends® in Machine Learning* 5.1 (2012), pp. 1–122. cit. on pp. 25, 35, 144

[BDM17] Marc G Bellemare, Will Dabney, and Rémi Munos. “A distributional perspective on reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 449–458. cit. on p. 21

[BK97] Apostolos N Burnetas and Michael N Katehakis. “Optimal adaptive policies for Markov decision processes”. In: *Mathematics of Operations Research* 22.1 (1997), pp. 222–255. cit. on p. 25

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. “Variational inference: A review for statisticians”. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877. cit. on p. 27

[BMT20] Hippolyte Bourel, Odalric Maillard, and Mohammad Sadegh Talebi. “Tightening exploration in upper confidence reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1056–1066.
cit. on pp. 26, 108, 122

[BP17] Vivek S Borkar and Sarath Pattathil. “Whittle indexability in egalitarian processor sharing systems”. In: *Annals of Operations Research* (2017), pp. 1–21.
cit. on pp. 1, 64

[BS20] David B Brown and James E Smith. “Index policies and performance bounds for dynamic selection problems”. In: *Management Science* 66.7 (2020), pp. 3029–3050. cit. on p. 41

[BT02] Ronen I Brafman and Moshe Tennenholtz. “R-max-a general polynomial time algorithm for near-optimal reinforcement learning”. In: *Journal of Machine Learning Research* 3.Oct (2002), pp. 213–231. cit. on p. 22

[BT12] Peter L Bartlett and Ambuj Tewari. “REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs”. In: *arXiv preprint arXiv:1205.2661* (2012). cit. on pp. 26, 28, 128, 160

[CK86] Yih Ren Chen and Michael N Katehakis. “Linear programming for finite state multi-armed bandit problems”. In: *Mathematics of Operations Research* 11.1 (1986), pp. 180–183. cit. on p. 63

[CM14] Jhelum Chakravorty and Aditya Mahajan. “Multi-armed bandits, Gittins index, and its calculation”. In: *Methods and applications of statistics in clinical trials: Planning, analysis, and inferential methods* 2.416-435 (2014), p. 455. cit. on pp. 41, 42, 63, 96, 109

[CW87] Don Coppersmith and Shmuel Winograd. “Matrix multiplication via arithmetic progressions”. In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 1–6. cit. on p. 4

[Dab+18] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. “Distributional reinforcement learning with quantile regression”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018. cit. on p. 21

[Dah+22] Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan, and Stephen L Smith. “Scalable operator allocation for multi-robot assistance: A restless bandit approach”. In: *IEEE Transactions on Control of Network Systems* (2022). cit. on p. 36

[DB15] Christoph Dann and Emma Brunskill. “Sample complexity of episodic fixed-horizon reinforcement learning”. In: *Advances in Neural Information Processing Systems* 28 (2015). cit. on p. 22

[DJ+09] Arnaud Doucet, Adam M Johansen, et al. “A tutorial on particle filtering and smoothing: Fifteen years later”. In: *Handbook of nonlinear filtering* 12.656-704 (2009), p. 3. cit. on p. 27

[DM62] Andrew Lloyd Dulmage and Nathan Saul Mendelsohn. “On the inversion of sparse matrices”. In: *Mathematics of Computation* 16.80 (1962), pp. 494–496. cit. on p. 186

[Dom+21] Omar Darwiche Domingues, Pierre Ménard, Emilie Kaufmann, and Michal Valko. “Episodic reinforcement learning in finite mdps: Minimax lower bounds revisited”. In: *Algorithmic Learning Theory*. PMLR. 2021, pp. 578–598. cit. on pp. 25, 26

[Duf95] Michael O Duff. “Q-learning for bandit problems”. In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 209–217. cit. on pp. 36, 111, 123, 155, 187

[FCG10] Sarah Filippi, Olivier Cappé, and Aurélien Garivier. “Optimism in reinforcement learning and Kullback-Leibler divergence”. In: *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2010, pp. 115–122. cit. on pp. 26, 108, 121, 122, 149, 150

[Fin97] Daniel Fink. “A compendium of conjugate priors”. In: *See http://www. people. cornell. edu/pages/df36/CONJINTRnew% 20TEX. pdf* 46 (1997). cit. on p. 152

[FPL18] Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. “Near optimal exploration-exploitation in non-communicating Markov decision processes”. In: *Advances in Neural Information Processing Systems* 31 (2018). cit. on p. 26

[FPL20] Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. “Improved analysis of ucrl2 with empirical bernstein inequality”. In: *arXiv preprint arXiv:2007.05456* (2020). cit. on pp. 28, 160

[Fru+17] Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Emma Brunskill. “Regret minimization in mdps with options without prior knowledge”. In: *Advances in Neural Information Processing Systems* 30 (2017). cit. on p. 26

[Fru+18] Ronan Fruit, Matteo Pirotta, Alessandro Lazaric, and Ronald Ortner. “Efficient bias-span-constrained exploration-exploitation in reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1578–1586. cit. on pp. 26, 28, 108, 122, 160

[Fru19] Ronan Fruit. “Exploration-exploitation dilemma in Reinforcement Learning under various form of prior knowledge”. PhD thesis. Université de Lille 1, Sciences et Technologies; CRIStAL UMR 9189, 2019. cit. on p. 138

[Fu+19] Jing Fu, Yoni Nazarathy, Sarat Moka, and Peter G Taylor. “Towards q-learning the whittle index for restless bandits”. In: *2019 Australian & New Zealand Control Conference (ANZCC)*. IEEE. 2019, pp. 249–254. cit. on pp. 39, 64, 111, 162, 183

[GGW11] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011. cit. on pp. 36, 38, 109

[GGY20] Nicolas Gast, Bruno Gaujal, and Chen Yan. “Exponential Convergence Rate for the Asymptotic Optimality of Whittle Index Policy”. In: *arXiv preprint arXiv:2012.09064* (2020). cit. on pp. 36, 91

[GGY22a] Nicolas Gast, Bruno Gaujal, and Chen Yan. *LP-based policies for restless bandits: necessary and sufficient conditions for (exponentially fast) asymptotic optimality*. 2022. DOI: 10.48550/ARXIV.2106.10067. URL: https://arxiv.org/abs/2106.10067. cit. on p. 41

[GGY22b] Nicolas Gast, Bruno Gaujal, and Chen Yan. “The LP-update policy for weakly coupled Markov decision processes”. In: *arXiv preprint arXiv:2211.01961* (2022). cit. on p. 187

[GHK11] Kevin D Glazebrook, David J Hodge, and Chris Kirkbride. “General notions of indexability for queueing control and asset management”. In: *The Annals of Applied Probability* 21.3 (2011), pp. 876–907. cit. on p. 187

[Git79] John C Gittins. “Bandit processes and dynamic allocation indices”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 41.2 (1979), pp. 148–164. cit. on pp. 1, 2, 4, 36, 38, 109, 120

[GJN21] Lachlan J Gibson, Peter Jacko, and Yoni Nazarathy. “A Novel Implementation of Q-Learning for the Whittle Index”. In: *EAI International Conference on Performance Evaluation Methodologies and Tools*. Springer. 2021, pp. 154–170. cit. on pp. 39, 49, 64, 65, 162, 183

[GKO09] Kevin D Glazebrook, Christopher Kirkbride, and Jamal Ouenniche. “Index policies for the admission control and routing of impatient customers to heterogeneous service stations”. In: *Operations Research* 57.4 (2009), pp. 975–989. cit. on p. 1

[GM02] KD Glazebrook and HM Mitchell. “An index policy for a stochastic scheduling model with improving/deteriorating jobs”. In: *Naval Research Logistics (NRL)* 49.7 (2002), pp. 706–721. cit. on pp. 1, 40, 162

[GM15] Aditya Gopalan and Shie Mannor. “Thompson sampling for learning parameterized markov decision processes”. In: *Conference on Learning Theory*. PMLR. 2015, pp. 861–898. cit. on p. 26

[GMS19] Aurélien Garivier, Pierre Ménard, and Gilles Stoltz. “Explore first, exploit next: The true shape of regret in bandit problems”. In: *Mathematics of Operations Research* 44.2 (2019), pp. 377–399. cit. on p. 25

[GRK06] Kevin D Glazebrook, Diego Ruiz-Hernandez, and Christopher Kirkbride. “Some indexable families of restless bandit problems”. In: *Advances in Applied Probability* 38.3 (2006), pp. 643–672. cit. on pp. 1, 40, 162

[GSZ00] Xianping Guo, Peng Shi, and Weiping Zhu. “A new strong optimality criterion for nonstationary Markov decision processes”. In: *Mathematical methods of operations research* 52 (2000), pp. 287–306. cit. on p. 51

[GU18] François Le Gall and Florent Urrutia. “Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2018, pp. 1029–1046. cit. on p. 88

[Gue+03] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. “Efficient solution algorithms for factored MDPs”. In: *Journal of Artificial Intelligence Research* 19 (2003), pp. 399–468. cit. on p. 109

[HF17] Weici Hu and Peter Frazier. “An asymptotically optimal index policy for finite-horizon restless bandits”. In: *arXiv preprint arXiv:1707.00205* (2017). cit. on p. 41

[HG15] David J Hodge and Kevin D Glazebrook. “On the asymptotic optimality of greedy index heuristics for multi-action restless bandits”. In: *Advances in Applied Probability* 47.3 (2015), pp. 652–667. cit. on pp. 36, 187

[Hua+16] Jianyu Huang, Tyler M Smith, Greg M Henry, and Robert A Van De Geijn. “Strassen’s algorithm reloaded”. In: *SC’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2016, pp. 690–701. cit. on pp. 90, 101

[Hua18] Jianyu Huang. “Practical fast matrix multiplication algorithms”. PhD thesis. 2018. cit. on pp. 90, 101

[HZG21a] Jiafan He, Dongruo Zhou, and Quanquan Gu. “Nearly minimax optimal reinforcement learning for discounted MDPs”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 22288–22300. cit. on p. 22

[HZG21b] Jiafan He, Dongruo Zhou, and Quanquan Gu. “Nearly minimax optimal reinforcement learning for discounted MDPs”. In: *Advances in Neural Information Processing Systems* 34 (2021). cit. on pp. 113, 118

[Ish+21] Haque Ishfaq, Qiwen Cui, Viet Nguyen, et al. “Randomized Exploration in Reinforcement Learning with General Value Function Approximation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4607–4616. cit. on p. 117

[JA18] Nan Jiang and Alekh Agarwal. “Open problem: The dependence of sample complexity lower bounds on planning horizon”. In: *Conference On Learning Theory*. PMLR. 2018, pp. 3395–3398. cit. on p. 22

[JAT19] Young Hun Jung, Marc Abeille, and Ambuj Tewari. “Thompson sampling in non-episodic restless bandits”. In: *arXiv preprint arXiv:1910.05654* (2019). cit. on pp. 160, 162, 171, 177, 180

[Jin+18] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. “Is Q-learning provably efficient?” In: *Advances in neural information processing systems* 31 (2018). cit. on pp. 21, 22, 25, 26, 187

[JOA10] Thomas Jaksch, Ronald Ortner, and Peter Auer. “Near-optimal Regret Bounds for Reinforcement Learning.” In: *Journal of Machine Learning Research* 11.4 (2010). cit. on pp. 4, 20, 21, 22, 24, 25, 26, 27, 28, 29, 30, 31, 108, 120, 121, 122, 128, 140, 160, 164

[JT19] Young Hun Jung and Ambuj Tewari. “Regret bounds for thompson sampling in episodic restless bandit problems”. In: *Advances in Neural Information Processing Systems* 32 (2019). cit. on pp. 36, 110

[Kak03] Sham Machandranath Kakade. “On the sample complexity of reinforcement learning”. PhD thesis. 2003. cit. on p. 22

[Kil+21] Jackson A Killian, Arpita Biswas, Sanket Shah, and Milind Tambe. “Q-learning Lagrange policies for multi-action restless bandits”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 871–881. cit. on p. 187

[KPT21] Jackson A Killian, Andrew Perrault, and Milind Tambe. “Beyond" To Act or Not to Act": Fast Lagrangian Approaches to General Multi-Action Restless Bandits”. In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 2021, pp. 710–718. cit. on pp. 36, 110

[KS02] Michael Kearns and Satinder Singh. “Near-optimal reinforcement learning in polynomial time”. In: *Machine learning* 49.2 (2002), pp. 209–232. cit. on p. 22

[KV87] Michael N Katehakis and Arthur F Veinott Jr. “The multi-armed bandit problem: decomposition and computation”. In: *Mathematics of Operations Research* 12.2 (1987), pp. 262–268. cit. on pp. 36, 63, 111

[LAV15] Maialen Larrañaga, Urtzi Ayesta, and Ina Maria Verloop. “Asymptotically optimal index policies for an abandonment queue with convex holding cost”. In: *Queueing systems* 81.2 (2015), pp. 99–169. cit. on p. 1

[Lev+20] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. “Offline reinforcement learning: Tutorial, review, and perspectives on open problems”. In: *arXiv preprint arXiv:2005.01643* (2020). cit. on p. 20

[LGR12] Sascha Lange, Thomas Gabel, and Martin Riedmiller. “Batch reinforcement learning”. In: *Reinforcement learning*. Springer, 2012, pp. 45–73. cit. on p. 20

[Li+21] Gen Li, Laixi Shi, Yuxin Chen, Yuantao Gu, and Yuejie Chi. “Breaking the sample complexity barrier to regret-optimal model-free reinforcement learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17762–17776. cit. on p. 26

[LPW17] David A Levin, Yuval Peres, and Elizabeth L Wilmer. *Markov chains and mixing times*. Vol. 107. American Mathematical Soc., 2017. cit. on p. 15

[LT00] Christopher Lott and Demosthenis Teneketzis. “On the optimality of an index rule in multichannel allocation for single-hop mobile networks with multiple service classes”. In: *Probability in the Engineering and Informational Sciences* 14.3 (2000), pp. 259–297. cit. on p. 40

[LZ10] Keqin Liu and Qing Zhao. “Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access”. In: *IEEE Transactions on Information Theory* 56.11 (2010), pp. 5547–5567. cit. on p. 1

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533. cit. on p. 21

[Mur07] Kevin P Murphy. “Conjugate Bayesian analysis of the Gaussian distribution”. In: *def* 1.2σ2 (2007), p. 16. cit. on p. 152

[Nak+21] Khaled Nakhleh, Santosh Ganji, Ping-Chun Hsieh, I Hou, and Srinivas Shakkottai. “NeurWIN: Neural Whittle Index Network For Restless Bandits Via Deep RL”. In: *Advances in Neural Information Processing Systems* 34 (2021). cit. on pp. 49, 64, 72, 162, 183

[Nin02] José Nino-Mora. “Dynamic allocation indices for restless projects and queueing admission control: a polyhedral approach”. In: *Mathematical programming* 93.3 (2002), pp. 361–413. cit. on p. 186

[Niñ07a] José Niño-Mora. “A (2/3) n 3 fast-pivoting algorithm for the Gittins index and optimal stopping of a Markov chain”. In: *INFORMS Journal on Computing* 19.4 (2007), pp. 596–606. cit. on pp. 42, 63

[Niñ07b] José Niño-Mora. "Dynamic priority allocation via restless bandit marginal productivity indices". In: *Top* 15.2 (2007), pp. 161–198. cit. on pp. 39, 42, 64, 91, 93, 94

[Niñ10] José Niño-Mora. "Characterization and Computation of Restless Bandit Marginal Productivity Indices". In: *1st International ICST Workshop on Tools for solving Structured Markov Chains*. 2010. cit. on pp. 42, 62, 64

[Niñ14] José Niño-Mora. "A dynamic page-refresh index policy for web crawlers". In: *International Conference on Analytical and Stochastic Modeling Techniques and Applications*. Springer. 2014, pp. 46–60. cit. on p. 1

[Niñ20] José Niño-Mora. "A fast-pivoting algorithm for Whittle's restless bandit index". In: *Mathematics* 8.12 (2020), p. 2226. cit. on pp. 39, 42, 62, 63, 64, 72, 84, 85, 92, 93, 94, 97, 98, 99

[NR83] H Niessner and K Reichert. "On computing the inverse of a sparse matrix". In: *International journal for numerical methods in engineering* 19.10 (1983), pp. 1513–1526. cit. on p. 186

[OPT18] Jungseul Ok, Alexandre Proutiere, and Damianos Tranos. "Exploration in structured reinforcement learning". In: *Advances in Neural Information Processing Systems* 31 (2018). cit. on p. 25

[Ort+12] Ronald Ortner, Daniil Ryabko, Peter Auer, and Rémi Munos. "Regret bounds for restless markov bandits". In: *International conference on algorithmic learning theory*. Springer. 2012, pp. 214–228. cit. on pp. 36, 110, 160, 162, 171, 177

[Ort20] Ronald Ortner. "Regret bounds for reinforcement learning via markov chain concentration". In: *Journal of Artificial Intelligence Research* 67 (2020), pp. 115–128. cit. on pp. 26, 28, 160

[ORV13] Ian Osband, Daniel Russo, and Benjamin Van Roy. "(More) efficient reinforcement learning via posterior sampling". In: *Advances in Neural Information Processing Systems* 26 (2013). cit. on pp. 4, 20, 21, 22, 26, 27, 29, 33, 34, 108, 120, 123, 138

[Ouy+17] Yi Ouyang, Mukul Gagrani, Ashutosh Nayyar, and Rahul Jain. "Learning unknown markov decision processes: A thompson sampling approach". In: *Advances in neural information processing systems* 30 (2017). cit. on pp. 20, 21, 26, 28, 117, 119, 160, 162, 163, 178, 183

[OV14] Ian Osband and Benjamin Van Roy. "Near-optimal reinforcement learning in factored mdps". In: *Advances in Neural Information Processing Systems* 27 (2014). cit. on pp. 109, 177

[OV16] Ian Osband and Benjamin Van Roy. "On lower bounds for regret in reinforcement learning". In: *arXiv preprint arXiv:1608.02732* (2016). cit. on p. 120

[OV17] Ian Osband and Benjamin Van Roy. "Why is posterior sampling better than optimism for reinforcement learning?" In: *International conference on machine learning*. PMLR. 2017, pp. 2701–2710. cit. on p. 138

[PT94] Christos H Papadimitriou and John N Tsitsiklis. “The complexity of optimal queueing network control”. In: *Proceedings of IEEE 9th Annual Conference on Structure in Complexity Theory*. IEEE. 1994, pp. 318–322. cit. on pp. 39, 160, 164

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. cit. on pp. 9, 10, 11, 12, 13, 14, 15, 16, 24, 41, 51, 53, 54, 55, 56, 57, 67, 68, 79, 109

[Qia+20] Jian Qian, Ronan Fruit, Matteo Pirotta, and Alessandro Lazaric. “Concentration inequalities for multinoulli random variables”. In: *arXiv preprint arXiv:2001.11595* (2020). cit. on pp. 138, 180

[Rag+08] Vivek Raghunathan, Vivek Borkar, Min Cao, and P Roshan Kumar. “Index policies for real-time multicast scheduling for wireless broadcast systems”. In: *IEEE INFOCOM 2008-The 27th Conference on Computer Communications*. IEEE. 2008, pp. 1570–1578. cit. on p. 1

[RM20] Aviv Rosenberg and Yishay Mansour. “Oracle-efficient reinforcement learning in factored MDPs with unknown structure”. In: *arXiv preprint arXiv:2009.05986* (2020). cit. on pp. 109, 177

[Rus+18] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. “A Tutorial on Thompson Sampling”. In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96. cit. on p. 114

[Sch93] Anton Schwartz. “A reinforcement learning method for maximizing undiscounted rewards”. In: *Proceedings of the tenth international conference on machine learning*. Vol. 298. 1993, pp. 298–305. cit. on p. 162

[SF78] Paul J Schweitzer and Awi Federgruen. “The functional equations of undiscounted Markov renewal programming”. In: *Mathematics of Operations Research* 3.4 (1978), pp. 308–321. cit. on pp. 17, 54, 57, 68

[Shi+22] Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. “Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity”. In: *arXiv preprint arXiv:2202.13890* (2022). cit. on p. 21

[SHS18] Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf. “SOAP: One clean analysis of all age-based scheduling policies”. In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2.1 (2018), pp. 1–30. cit. on p. 1

[Son08] Isaac M Sonin. “A generalized Gittins index for a Markov chain and its recursive calculation”. In: *Statistics & Probability Letters* 78.12 (2008), pp. 1526–1533. cit. on pp. 42, 63

[Str69] Volker Strassen. “Gaussian elimination is not optimal”. In: *Numerische mathematik* 13.4 (1969), pp. 354–356. cit. on pp. 62, 84, 85

[Tho33] William R Thompson. “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* 25.3-4 (1933), pp. 285–294. cit. on pp. 26, 109

[TL10] Cem Tekin and Mingyan Liu. “Online algorithms for the multi-armed bandit problem with markovian rewards”. In: *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2010, pp. 1675–1682. cit. on pp. 36, 37

[TL12] Cem Tekin and Mingyan Liu. “Online learning of rested and restless bandits”. In: *IEEE Transactions on Information Theory* 58.8 (2012), pp. 5588–5611. cit. on p. 110

[TM18] Mohammad Sadegh Talebi and Odalric-Ambrym Maillard. “Variance-aware regret bounds for undiscounted reinforcement learning in mdps”. In: *Algorithmic Learning Theory*. PMLR. 2018, pp. 770–805. cit. on pp. 108, 122

[TQS20] Yi Tian, Jian Qian, and Suvrit Sra. “Towards minimax optimal reinforcement learning in factored markov decision processes”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19896–19907. cit. on p. 109

[Tsi07] John N Tsitsiklis. “NP-hardness of checking the unichain condition in average cost MDPs”. In: *Operations research letters* 35.3 (2007), pp. 319–323. cit. on p. 15

[Van91] Robert J Vanderbei. “Splitting dense columns in sparse linear systems”. In: *Linear Algebra and its Applications* 152 (1991), pp. 107–117. cit. on p. 186

[VBW15] Sofia S Villar, Jack Bowden, and James Wason. “Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges”. In: *Statistical science: a review journal of the Institute of Mathematical Statistics* 30.2 (2015), p. 199. cit. on p. 1

[Ver16] Ina Maria Verloop. “Asymptotically optimal priority policies for indexable and nonindexable restless bandits”. In: *The Annals of Applied Probability* 26.4 (2016), pp. 1947–1995. cit. on p. 40

[Ver18] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018. cit. on p. 152

[Wan+20] Ruosong Wang, Simon S Du, Lin Yang, and Sham Kakade. “Is long horizon rl more difficult than short horizon RL?” In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9075–9085. cit. on p. 22

[Wat89] Christopher John Cornish Hellaby Watkins. “Learning from delayed rewards”. In: (1989). cit. on pp. 20, 162

[Wei+03] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. “Inequalities for the L1 deviation of the empirical distribution”. In: *Hewlett-Packard Labs, Tech. Rep* (2003). cit. on p. 130

[Wei+20] Chen-Yu Wei, Mehdi Jafarnia Jahromi, Haipeng Luo, Hiteshi Sharma, and Rahul Jain. “Model-free reinforcement learning in infinite-horizon average-reward markov decision processes”. In: *International conference on machine learning*. PMLR. 2020, pp. 10170–10180. cit. on pp. 164, 187

[Whi88] Peter Whittle. “Restless bandits: Activity allocation in a changing world”. In: *Journal of applied probability* 25.A (1988), pp. 287–298. cit. on pp. 2, 36, 39, 40, 96

[Whi96] Peter Whittle. *Optimal control: basics and beyond*. John Wiley & Sons, Inc., 1996. cit. on pp. 2, 36, 38, 39, 40, 41, 153

[WHL20] Siwei Wang, Longbo Huang, and John Lui. “Restless-UCB, an efficient and low-complexity algorithm for online restless bandits”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11878–11889. cit. on pp. 36, 110, 162, 177, 183, 186

[WM95] Hong Shen Wang and Nader Moayeri. “Finite-state Markov channel-a useful model for radio communication channels”. In: *IEEE transactions on vehicular technology* 44.1 (1995), pp. 163–171. cit. on p. 186

[Woo50] Max A Woodbury. *Inverting modified matrices*. 1950. cit. on p. 89

[WSY20] Ruosong Wang, Russ R Salakhutdinov, and Lin Yang. “Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6123–6135. cit. on p. 117

[WW90] Richard R Weber and Gideon Weiss. “On an index policy for restless bandits”. In: *Journal of applied probability* (1990), pp. 637–648. cit. on pp. 40, 41, 162, 176

[XT20] Ziping Xu and Ambuj Tewari. “Reinforcement learning in factored mdps: Oracle-efficient algorithms and tighter regret bounds for the non-episodic setting”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18226–18236. cit. on pp. 109, 177

[ZB19] Andrea Zanette and Emma Brunskill. “Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7304–7312. cit. on pp. 20, 22, 26, 27

[ZF21] Xiangyu Zhang and Peter I Frazier. “Restless bandits with many arms: Beating the central limit theorem”. In: *arXiv preprint arXiv:2107.11911* (2021). cit. on p. 41

[ZJ19] Zihan Zhang and Xiangyang Ji. “Regret minimization for reinforcement learning by evaluating the optimal bias function”. In: *Advances in Neural Information Processing Systems* 32 (2019). cit. on pp. 22, 25, 26, 28, 160

[ZJD21] Zihan Zhang, Xiangyang Ji, and Simon Du. “Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon”. In: *Conference on Learning Theory*. PMLR. 2021, pp. 4528–4531. cit. on pp. 22, 26

Abstract

A Markovian bandit is a sequential decision problem in which the decision maker has to activate a set of bandit's arms at each time, and the active arms evolve in a Markovian manner. There are two types of Markovian bandits: (i) *rested* bandits where the arms that are not activated (i.e., are passive) remain frozen, and (ii) *restless* bandits where the passive arms evolve in a Markovian manner. In general, Markovian bandits suffer from the curse of dimensionality that often makes the exact solution computationally intractable. So, one has to resort to tractable heuristics such as index policies. Two celebrated indices are the Gittins index for rested bandits and the Whittle index for restless bandits.

This thesis focuses on two questions (1) index computation when all model parameters are known and (2) learning algorithms when the parameters are unknown.

For index computation, we point out the ambiguities in the classical indexability definition and propose a definition that assures the uniqueness of the Whittle index when this latter exists. We then develop an algorithm for testing the indexability and computing the Whittle indices of a restless arm. The theoretical complexity of our algorithm is $\mathcal{O}(S^{2.5286})$, where S is the number of arm's states.

For learning in rested bandits, we propose modifications of PSRL and UCBVI algorithms that we call MB-PSRL and MB-UCBVI. We show that they can leverage Gittins index policy to have a regret guarantee and a runtime scalable in the number of arms. Furthermore, we show that MB-UCRL2, a modification of UCRL2, also has a regret guarantee scalable in the number of arms. However, MB-UCRL2 has a runtime exponential in the number of arms. When learning in restless bandits, the regret guarantee depends heavily on the structure of the bandit. We study how the structure of arms translates into the structure of the bandit. We exhibit a subclass of restless bandits that are not learnable. We also show that it is difficult to construct a subclass of restless bandits with a desirable learning structure by only making assumptions about arms.

Résumé

Un bandit markovien est un problème de décision séquentielle dans lequel un sous-ensemble de bras doivent être activés à chaque instant, et les bras évoluent de manière markovienne. Il y a deux catégories de bandits markoviens. Si les bras qui ne sont pas activés restent figés, on entre alors dans la catégorie des bandits markoviens *avec repos*. S'ils évoluent de manière markovienne, on parle alors de bandit markovien *sans repos*. En général, les bandits markoviens souffrent de la malédiction de la dimension qui rend souvent la solution exacte prohibitive en terme de calculs. Il faut donc recourir à des heuristiques telles que les politiques d'indice. Deux indices célèbres sont l'indice de Gittins pour les bandits avec repos et l'indice de Whittle pour les bandits sans repos.

Cette thèse se concentre sur deux questions : (1) le calcul d'indices lorsque tous les paramètres du modèle sont connus et (2) les algorithmes d'apprentissage lorsque les paramètres sont inconnus.

Pour le calcul de l'indice, nous relevons les ambiguïtés de la définition classique de l'indexabilité et proposons une définition qui assure l'unicité de l'indice de Whittle quand ce dernier existe. Nous développons ensuite un algorithme testant l'indexabilité et calculant les indices de Whittle. La complexité théorique de notre algorithme est $\mathcal{O}(S^{2.5286})$, où S est le nombre d'états du bras.

Pour l'apprentissage dans les bandits avec repos, nous montrons que MB-PSRL et MB-UCBVI, des versions modifiées des algorithmes PSRL et UCBVI, peuvent tirer parti de la politique d'indice de Gittins pour avoir une garantie de regret et un temps d'exécution qui passent à l'échelle avec le nombre de bras. De plus, nous montrons que MB-UCRL2, une version modifiée de UCRL2, possède également une garantie de regret qui passe à l'échelle. Cependant, MB-UCRL2 a un temps d'exécution exponentiel dans le nombre de bras. Lors de l'apprentissage dans les bandits sans repos, la garantie de regret dépend fortement de la structure du bandit. Ainsi, nous étudions comment la structure des bras se traduit dans la structure du bandit. Nous exposons une sous-classe de bandits sans repos qui ne sont pas apprenables. Nous montrons également qu'il est difficile de construire des hypothèses sur les bras qui rendent les bandits sans repos apprenables efficacement.