2021 級 信工計算機組 學生干皓丞 2101212850 人工智慧作業報告

Email : zxdfgcv@gmail.com
About me : https://kancheng.github.io/

0. 作業說明

此報告為人工智慧課程五篇閱讀報告中的第三篇 Markov decision process，全報告包含 agent、search、Markov decision process、Bayesian Network、Reinforcement Learning。第一篇所佔的領域涵蓋 search 和 Reinforcement Learning。因為考量自身到對該領域知識的掌握程度不足，全報告採心得與翻譯。

* 研究團隊在前言就先說因應 Reinforcement Learning 下，agent 在未知**"環境"**中的重複行動。而形式上環境是 Markov Decision Process (MDP)，團隊基於此提新的方法。

GitHub Project : https://github.com/kancheng/kan-readpaper-cv-and-ai-in-2021

1. 原文獻資訊與作者

Planning in Markov Decision Processes with Gap-Dependent Sample Complexity
用間隙相關樣本複雜性來做出利用馬爾可夫決策過程所做出的規劃

Anders Jonsson; Universitat Pompeu Fabra; anders.jonsson@upf.edu
Emilie Kaufmann; CNRS & ULille (CRIStAL), Inria SequeL; emilie.kaufmann@univ-lille.fr
Pierre Ménard; Inria Lille, SequeL team; pierre.menard@inria.fr
Omar Darwiche Domingues; Inria Lille, SequeL team; omar.darwiche-domingues@inria.fr
Edouard Leurent; Renault & Inria Lille, SequeL team; edouard.leurent@inria.fr
Michal Valko; DeepMind Paris; valkom@deepmind.com

https://arxiv.org/abs/2006.05879
https://proceedings.neurips.cc/paper/2020
https://proceedings.neurips.cc/paper/2020/hash/0d85eb24e2add96ff1a7021f83c1abc9-Abstract.html
Part of Advances in Neural Information Processing Systems 33 (NeurIPS 2020)

2. 報告內容心得與講述

(1) Motivation 动机

該研究的研究者提出名為一個名為 MDP-GapE 的算法，這個演算法是一種新的基於“軌跡的蒙特卡洛樹搜索演算法“(trajectory-based Monte-Carlo Tree Search algorithm)，此算法用於在馬爾可夫決策過程中進行規劃，該算法會在過程中進行規畫，且而這個過程的轉換是具有‘有限支持‘(a finite support)。

在強化學習下工分為三種，所謂的監督、無監督、強化，而馬柯夫決策過程(MDP;Markov Decision Process)是強化學習的基礎。而基於馬柯夫性質有了馬柯夫決策過程，所謂的馬可夫性質是我們下一步的狀態只和我們當前的狀態有關與我們當前狀態之前的所有狀態是無關的。

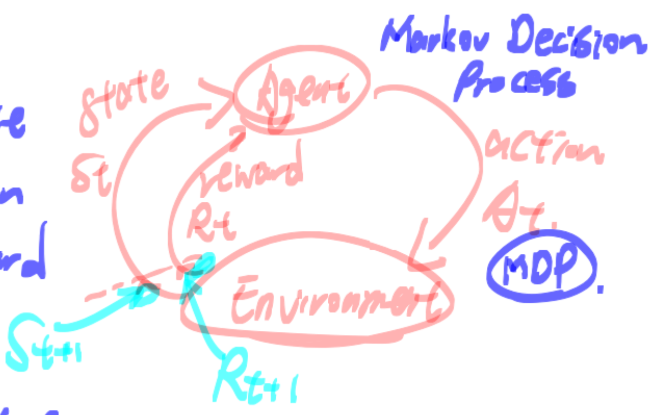
*備註 :讀該文獻建議先找先備知識。
https://ithelp.ithome.com.tw/articles/10233543
https://www.bilibili.com/s/video/BV1Zb411u7MK
https://www.zhihu.com/question/39916945

依據文獻前言的研究者指出在強化學習 (RL) 中，代理(agent)　在由狀態所描述出來的未知環境中重複行動並觀察獎勵。而這當中的環境是一個馬爾可夫決策過程（MDP） M = (S,A, p, r)，其中 S 是狀態空間，A 是動作空間，p = {Ph}h>=1 一組轉換內核 和 r = {rh}h>=1 一組獎勵函數。

* agent -> 可以被翻譯成顧問、代理，或某技術文章中的被識別的物件，又或者是機器人，總之還是以原文為準，中文為參考用。

而當中的蒙特卡羅樹搜索 (MCTS;Monte-Carlo Tree Search) 則是蒙特卡羅規劃的一種形式，它使用前向模型對當前狀態的轉換進行採樣，而不是可以在任何地方採樣的完整生成模型。大多數 MCTS 算法會從當下的狀態(the current state)之中去做採樣軌跡(sample trajectories)，並廣泛泛用於確定性遊戲，例如圍棋。比如過往的 AlphaZero 算法，使用價值和策略估計來指導規劃，以生成改進這些估計的軌跡。

強化學習:
3.要素 (1) State
(2) Action
(3) Reward
2.特性,
(1)馬爾過程隨機性、
(2)下一個狀態轉換只跟現在的狀態有關



狀態 State, $S = \{s_0, s_1, \ldots, s_n\}$
動作 Action, $A(s) = \{a_0, a_1, \ldots, a_m\}$ $\rightarrow$ $\underbrace{M = (S, A, P, r)}$.
獎勵 Reward, $R = \{r_0, r_1, \ldots, r_n\}$.
轉移機率 Dynamic, $P = p(s', r \mid s, a)$, for all $s \in S$, $a \in A$

(2) Intuition 直覺

Table 1: Different settings of planning algorithms in the literature

Setting	Input	Output	Optimality criterion
(1) Fixed confidence (action-based)	ε, δ	$\widehat{a}_n$	$\mathbb{P}\left(\overline{r}_n(\widehat{a}_n) \leq \varepsilon\right) \geq 1-\delta$
(2) Fixed confidence (value-based)	ε, δ	$\widehat{V}(s_1)$	$\mathbb{P}\left(\lvert\widehat{V}(s_1) - V^\star(s_1)\rvert \leq \varepsilon\right) \geq 1-\delta$
(3) Fixed budget	n (budget)	$\widehat{a}_n$	$\mathbb{E}\left[\overline{r}_n(\widehat{a}_n)\right]$ decreasing in n
(4) Anytime	-	$\widehat{a}_n$	$\mathbb{E}\left[\overline{r}_n(\widehat{a}_n)\right]$ decreasing in n

We propose an algorithm in the *fixed confidence* setting (ε, δ): after n calls to the generative model, the algorithm should return an action $\hat{a}_n$ such that $\overline{r}_n(\hat{a}_n) \leq \varepsilon$ with probability at least $1-\delta$. We prove that its *sample complexity* n is bounded in high probability by a quantity that depends on the sub-optimality gaps of the actions that are applicable in state s_1. We also provide experiments showing its effectiveness. The only assumption that we make on the MDP is that the support of the transition probabilities $p_h(\cdot|s,a)$ should have cardinality bounded by $B < \infty$, for all s, a and h.

研究者提出了一種在固定置信度設置 (fixed confidence)(ε , δ) 中的算法：在對生成模型進行 n 次調用後，該算法應該返回一個動作 (a^)n 使得 (r_)n((a^)n)<= ε " 的概率至少為 1 - δ 。證明了它的樣本複雜度 n 以高概率受到一個數量的限制，該數量取決於適用於狀態 s1 的動作的次優差距。

該研究貢獻則是我們提出了 MDP-GapE，這是一種新的 MCTS 算法，用於在 B < ∞ 的情況下進行規劃。MDP-GapE 可以執行高效的蒙特卡洛規劃，而且是一種簡單的基於軌跡的算法，在實際測試中表現良好，僅依賴於前向模型。另外大多數實際的 MCTS 算法在理論上還沒有得到很好的實踐，但我們證明了 MDP-GapE 樣本複雜度的上限。該方法的界限取決於與探索過程中遇到的狀態-動作對相關的次優差距。

(3) Justification 理由

Table 2: Algorithms with sample complexity guarantees

Algorithm	Setting	Sample complexity	Remarks
Sparse Sampling [19]	(1)-(2)	$H^5(BK)^H/\varepsilon^2$ or $\varepsilon^{-\left(2+\frac{\log(K)}{\log(1/\gamma)}\right)}$	proved in Lemma 1
OLOP [2]	(3)	$\varepsilon^{-\max\left(2, \frac{\log \kappa}{\log(1/\gamma)}\right)}$	open loop, $\kappa \in [1, K]$
OP [3]	(4)	$\varepsilon^{-\frac{\log \kappa}{\log(1/\gamma)}}$	known MDP, $\kappa \in [0, BK]$
BRUE [8]	(4)	$H^4(BK)^H/\Delta^2$	minimal gap Δ
StOP [27]	(1)	$\varepsilon^{-\left(2+\frac{\log \kappa}{\log(1/\gamma)}+o(1)\right)}$	$\kappa \in [0, BK]$
TrailBlazer [13]	(2)	$\varepsilon^{-\max\left(2, \frac{\log(B\kappa)}{\log(1/\gamma)}+o(1)\right)}$	$\kappa \in [1, K]$
SmoothCruiser [14]	(2)	ε^{-4}	only regularized MDPs
MDP-GapE (ours)	(1)	$\sum_{a_1 \in \mathcal{A}} \frac{H^2(BK)^{H-1}B}{(\Delta_1(s_1,a_1) \vee \Delta \vee \varepsilon)^2}$	see Corollary 1

Corollary 1. *The number of episodes used by MDP-GapE satisfies*

$$\mathbb{P}\left(\tau = \mathcal{O}\left(\sum_{a_1} \frac{(BK)^{H-1}}{(\Delta_1(s_1,a_1) \vee \Delta \vee \varepsilon)^2}\left[\log\left(\frac{1}{\delta}\right) + BH\log(BK)\right]\right)\right) \geq 1-\delta\,.$$

(4) Framework 框架

Algorithm 1 MDP-GapE

```
1: Input: confidence level $\delta$, tolerance $\varepsilon$
2: initialize data lists $\mathcal{D}_h \leftarrow [\,]$ for all $h \in [H]$
3: for $t = 1 \dots$ do
4:     //Update confidence bounds
5:     $U_h^{t-1}, L_h^{t-1} \leftarrow \texttt{UpdateBounds}(t, \delta, \mathcal{D}_h)$
6:     if $U_1^{t-1}(s_1, c^t) - L_1^{t-1}(s_1, b^t) \le \varepsilon$ then
7:         return $b_{t-1}$, break
8:     end if
9:     // Best
10:    $b^{t-1} \leftarrow \underset{b}{\text{argmin}} \left[\max_{a \neq b} U_1^{t-1}(s_1, a) - L_1^{t-1}(s_1, b)\right]$
11:    //Challenger
12:    $c^{t-1} \leftarrow \underset{c \neq b^t}{\text{argmax}}\, U_1^{t-1}(s_1, c)$
13:    //Exploration
14:    $a_1^t \leftarrow \underset{a \in \{b^{t-1}, c^{t-1}\}}{\text{argmax}} \left[U_1^{t-1}(s_1, a) - L_1^{t-1}(s_1, a)\right]$
15:    observe reward $r_1^t$, next state $s_2^t$, save $\mathcal{D}_1$.append$(s_1^t, a_1^t, s_2^t, r_1^t)$
16:    for step $h = 2, \dots, H$ do
17:        $a_h^t \leftarrow \underset{a}{\text{argmax}}\, U_h^{t-1}(s_h^t, a)$
18:        observe reward $r_{h-1}^t$, next state $s_h^t$,save $\mathcal{D}_h$.append$(s_h^t, a_h^t, s_{h+1}^t, r_h^t)$
19:    end for
20: end for
```

(5) Result 结果

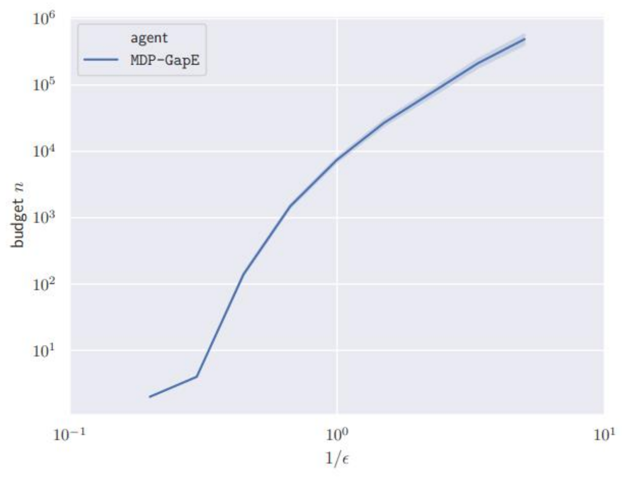


Figure 1: Polynomial dependency of the number n of oracle calls with respect to $1/\varepsilon$.

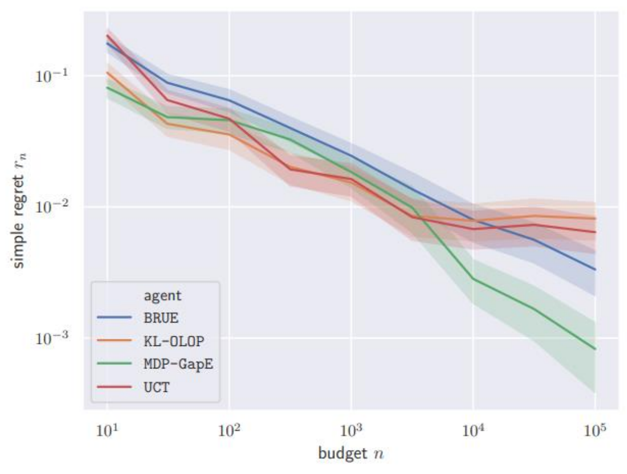


Figure 2: Comparison to KL-OLOP in a fixed-budget setting.

Figure 1: Polynomial dependency of the number n of oracle calls with respect to 1/ε.

圖 1：oracle 調用次數 n 與 1/ε 的多項式相關性。

Figure 2: Comparison to KL-OLOP in a fixedbudget setting.

圖 2：在固定預算設置中與 KL-OLOP 的比較。

該研究將自己所提出的 MDP-GapE 與當下現有的三個方案進行比較。所以一共有四個方案進行實測，能夠從圖二看到藍色為 BRUE 算法，該算法使用統一探索並處理閉環策略(explores uniformly and handles closed-loop policies)，第二個方案是黃色，也就是 KL-OLOP 算法，這個算法擁有研究者所提出的 MDP-GapE 方發，擁有相同的獎勵 uth 和狀態值 Uth 的置信上限，但僅限於開環策略 ，亦即只有動作序列。而第三個綠色則是研究者所提出的 MDP-GapE，而第四個紅色則是 UCT 算法，這個

算法的也屬於閉環，同時會在所有的深度進行樂觀探索(closed-loop and performs optimistic exploration at all depths)，此算法相當流行，雖然該算法的變化版本缺乏理論保證，但在很多實際應用上很成功。從結果來看證明研究者所提出的 MDP-GapE 在高預算制度(the high-budget regime)中與這些基線相比具有優勢。

回到研究者的摘要上所述 ：
Our experiments reveal that MDP-GapE is also effective in practice, in contrast with other algorithms with sample complexity guarantees in the fixed-confidence setting, that are mostly theoretical.
我們的實驗表明，在與固定置信度設置(the fixed-confidence setting)中具有樣本複雜性保證的(sample complexity guarantees)其他主要是理論性的算法，相比之下 MDP-GapE 在實踐中也是有效的。
(研究者說他這東西在實踐中有效，其他都是理論性 !!!!)

3. 原研究文獻

Abstract 摘要

We propose MDP-GapE, a new trajectory-based Monte-Carlo Tree Search algorithm for planning in a Markov Decision Process in which transitions have a finite support.
我們提出了 MDP-GapE，這是一種新的基於軌跡的蒙特卡洛樹搜索算法(trajectory-based Monte-Carlo Tree Search algorithm)，此算法用於在馬爾可夫決策過程中進行規劃，在過程中進行規畫，而這個過程的轉換具有有限支持(a finite support)。
* trajectory 彈道, 軌道
We prove an upper bound on the number of calls to the generative models needed for MDP-GapE to identify a near-optimal action with high probability.
我們證明了對 MDP-GapE 所需的生成模型調用次數的上限，以用最高產生的可能性(高概率識別)，從而找出接近最好的動作。

This problem dependent sample complexity result is ***expressed in terms of the sub-optimality gaps of the state-action pairs*** *that are visited during exploration*.
這種問題相關的樣本複雜性結果 是以 在我們尋找最高可能性的期間考慮過(探索期間訪問)的狀態-動作對的次優差距(the sub-optimality gaps of the state-action pairs)來被產生出來。

Our experiments reveal that MDP-GapE is also effective in practice, in contrast with other algorithms with sample complexity guarantees in the fixed-confidence setting, that are mostly theoretical.
我們的實驗表明，在與固定置信度設置(the fixed-confidence setting)中具有樣本複雜性保證的(sample complexity guarantees)其他主要是理論性的算法，相比之下 MDP-GapE 在實踐中也是有效的。
(研究者說他這東西在實踐中有效，其他都是理論性 !!!!)

1 Introduction 前言

In reinforcement learning (RL), an agent repeatedly takes actions and observes rewards in an unknown

environment described by a state.
在強化學習 (RL) 中，代理(agent)　在由狀態所描述出來的未知環境中　重複行動並觀察獎勵。

** agent -> 可以被翻譯成顧問、代理，或某技術文章中的被識別的物件，又或者是機器人，總之還是以原文為準，中文為參考用。*

Formally, the environment is a Markov Decision Process (MDP) M = (S,A, p, r), where S is the state space, A the action space, p = {Ph}h>=1 a set of transition kernels and r = {rh}h>=1 a set of reward functions.
形式上，環境是一個馬爾可夫決策過程（MDP） M = (S,A, p, r)，其中 S 是狀態空間，A 是動作空間，p = {Ph}h>=1 一組轉換內核 和 r = {rh}h>=1 一組獎勵函數。

By taking action a in state s at step h, the agent reaches a state s’ with probability ph(s’|s, a) and receives a random reward with mean rh(s, a).
通過在步驟 h 在狀態 s 中採取行動 a，智能體以 ph(s'|s, a) 的概率到達狀態 s'，並收到平均為 rh(s, a) 的隨機獎勵。

A common goal is to learn a policy π = (πh)h>=1 that maximizes cumulative reward by taking action π h(s) in state s at step h.
一個共同的目標是學習策略 π = (πh)h>=1，通過在步驟 h 在狀態 s 中採取行動 πh(s) 來最大化累積獎勵。

If the agent has access to a generative model, it may plan before acting by generating additional samples in order to improve its estimate of the best action to take next.
如果代理可以訪問生成模型，它可以通過生成額外的樣本在行動之前進行計劃，以改進其對下一步採取的最佳行動的估計。

In this work, we consider Monte-Carlo planning as the task of recommending a good action to be taken by the agent in a given state s1, by using samples gathered from a generative model.
在這項工作中，我們將蒙特卡洛規劃視為通過使用從生成模型收集的樣本來推薦代理在給定狀態 s1 中採取的良好行動的任務。

*備註： $\widehat{a}$ (圖)-> \widehat a (LaTex)-> (a^)
我用此方式表達，由於這只是給自己閱讀方便的手稿，不打算處理數學式用得太費力。

Let Q*(s1; a) be the maximum cumulative reward, in expectation, that can be obtained from state s1 by first taking action a, and let (^a)n be the recommended action after n calls to the generative model.
假設 Q*(s1; a) 是最大累積獎勵，在期望中，可以通過首先採取行動 a 從狀態 s1 獲得，並讓 (^a)n 是對生成模型調用 n 次後的推薦行動。

The quality of the action recommendation is measured by its simple regret, defined as (r_)n((a^)n) := V *(s1) – Q*(s, (a^)n), where V*(s1) := maxa Q*(s1, a).

動作推薦的質量由其簡單的遺憾來衡量，定義為 (r_)n((a^)n) := V *(s1) － Q*(s, (a^)n)，其中 V*(s1) := maxa Q*(s1, a)。

We propose an algorithm in the fixed confidence setting (ε,δ): after n calls to the generative model, the algorithm should return an action (a^)n such that (r_)n((a^)n)<= ε " with probability at least 1 - δ.
我們提出了一種在固定置信度設置 (ε,δ) 中的算法：在對生成模型進行 n 次調用後，該算法應該返回一個動作 (a^)n 使得 (r_)n((a^)n)<= ε " 的概率至少為 1 - δ。

We prove that its sample complexity n is bounded in high probability by a quantity that depends on the sub-optimality gaps of the actions that are applicable in state s1.
我們證明了它的樣本複雜度 n 以高概率受到一個數量的限制，該數量取決於適用於狀態 s1 的動作的次優差距。

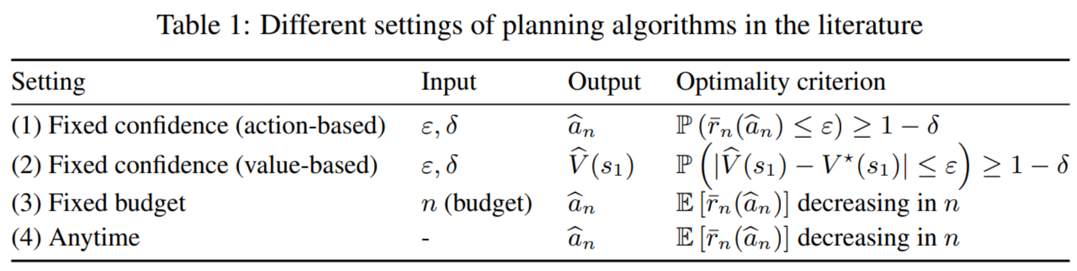


Table 1: Different settings of planning algorithms in the literature

Setting	Input	Output	Optimality criterion
(1) Fixed confidence (action-based)	ε, δ	$\widehat{a}_n$	$\mathbb{P}\left(\bar{r}_n(\widehat{a}_n) \leq \varepsilon\right) \geq 1-\delta$
(2) Fixed confidence (value-based)	ε, δ	$\widehat{V}(s_1)$	$\mathbb{P}\left(\lvert\widehat{V}(s_1) - V^\star(s_1)\rvert \leq \varepsilon\right) \geq 1-\delta$
(3) Fixed budget	n (budget)	$\widehat{a}_n$	$\mathbb{E}\left[\bar{r}_n(\widehat{a}_n)\right]$ decreasing in n
(4) Anytime	-	$\widehat{a}_n$	$\mathbb{E}\left[\bar{r}_n(\widehat{a}_n)\right]$ decreasing in n

Table 1: Different settings of planning algorithms in the literature
表 1：文獻中規划算法的不同設置

We also provide experiments showing its effectiveness.
我們還提供了顯示其有效性的實驗。

The only assumption that we make on the MDP is that the support of the transition probabilities ph(·|s, a) should have cardinality bounded by B < ∞, for all s, a and h.
我們對 MDP 所做的唯一假設是，對於所有 s、a 和 h，轉移概率 ph(·|s, a) 的支持應該具有以 B < ∞ 為界的基數。

Monte-Carlo Tree Search (MCTS) is a form of Monte-Carlo planning that uses a forward model to sample transitions from the current state, as opposed to a full generative model that can sample anywhere.
蒙特卡羅樹搜索 (MCTS) 是蒙特卡羅規劃的一種形式，它使用前向模型對當前狀態的轉換進行採樣，而不是可以在任何地方採樣的完整生成模型。

Most MCTS algorithms sample trajectories from the current state [1], and are widely used in deterministic games such as Go.
大多數 MCTS 算法從當前狀態 [1] 中採樣軌跡，並廣泛用於確定性遊戲，例如圍棋。

The AlphaZero algorithm [25] guides planning using value and policy estimates to generate trajectories that improve these estimates.
AlphaZero 算法 [25] 使用價值和策略估計來指導規劃，以生成改進這些估計的軌跡。

The MuZero algorithm [24] combines MCTS with a model-based method which has proven useful for stochastic environments.
MuZero 算法 [24] 將 MCTS 與基於模型的方法相結合，該方法已被證明適用於隨機環境。

Hence efficient Monte-Carlo planning may be instrumental for learning better policies.
因此，有效的蒙特卡洛規劃可能有助於學習更好的政策。

Despite their empirical success, little is known about the sample complexity of state-of-the-art MCTS algorithms.
儘管他們在經驗上取得了成功，但對最先進的 MCTS 算法的樣本複雜性知之甚少。

Related work 相關工作

The earliest MCTS algorithm with theoretical guarantees is Sparse Sampling [19], whose sample complexity is polynomial in $1/\varepsilon$ in the case $B < \infty$ (see Lemma 1).
最早的具有理論保證的 MCTS 算法是稀疏採樣 [19]，其樣本複雜度在 $B < \infty$ 的情況下為 $1/\varepsilon$ 的多項式（參見引理 1）。

However, it is not trajectory-based and does not select actions adaptively, making it very inefficient in practice.
然而，它不是基於軌跡的(trajectory-based)，也不會自適應地選擇動作，因此在實踐中效率很低。

Since then, adaptive planning algorithms with small sample complexities have been proposed in different settings with different optimality criteria.
從那時起，在具有不同最優性標準的不同設置中提出了具有小樣本複雜性的自適應規划算法。

In Table 1, we summarize the most common settings, and in Table 2, we show the sample complexity of related algorithms (omitting logarithmic terms and constants) when $B < \infty$.
在表 1 中，我們總結了最常見的設置，在表 2 中，我們顯示了當 $B < \infty$ 時相關算法的樣本複雜度（省略對數項和常數）。

Algorithms are either designed for a discounted setting with $\gamma < 1$ or an episodic setting with horizon H.
算法要么是為 $\gamma < 1$ 的折扣環境設計的，要么是為 H 水平的情節環境設計的。

Sample complexities are stated in terms of the accuracy ε (for algorithms with fixed-budget guarantees we solve $E[(\bar{r})n] = \varepsilon$ for n), the number of actions K, the horizon H or the discount factor γ and a problem-dependent quantity κ which is a notion of branching factor of near-optimal nodes whose exact definition varies.
求解 $E[(\bar{r})n] = \varepsilon$ for n)、動作數量 K、範圍 H 或折扣因子 γ 和一個問題相關量 κ，它是近似最優節點的分支因子的概念，其精確 定義不同。

A first category of algorithms rely on optimistic planning [22], and require additional assumptions:
第一類算法依賴於樂觀規劃 [22]，並需要額外的假設：

a deterministic MDP [15], the open loop setting [2, 21] in which policies are sequences of actions instead of state-action mappings (the two are equivalent in MDPs with deterministic transitions), or an MDP with known parameters [3].
確定性 MDP [15]，開環設置 [2, 21]，其中策略是動作序列而不是狀態-動作映射（兩者在具有確定性轉換的 MDP 中是等效的），或具有已知參數的 MDP [3] .

For MDPs with stochastic and unknown transitions, polynomial sample complexities have been obtained for StOP [27], TrailBlazer [13] and SmoothCruiser [14] but the three algorithms suffer from numerical inefficiency, even for $B < \infty$.
對於具有隨機和未知轉換的 MDP，已經為 StOP [27]、TrailBlazer [13] 和 SmoothCruiser [14] 獲得多項式樣本複雜度，但是這三種算法都存在數值效率低下的問題，即使對於 $B < \infty$。

Indeed, StOP explicitly reasons about policies and storing them is very costly, while TrailBlazer and SmoothCruiser require a very large amount of recursive calls even for small MDPs.
事實上，StOP 明確地對策略進行推理並存儲它們的成本非常高，而 TrailBlazer 和 SmoothCruiser 需要非常大量的遞歸調用，即使對於小的 MDP 也是如此。

We remark that popular MCTS algorithms such as UCT [20] are not (ε, δ)-correct and do not have provably small sample complexities.
我們注意到流行的 MCTS 算法如 UCT [20] 不是 (ε, δ) 正確的，並且沒有可證明的小樣本複雜性。

In the setting $B < \infty$, BRUE [8] is a trajectory-based algorithm that is anytime and whose sample complexity depends on the smallest sub-optimality gap $\Delta := \min(\ldots)$.

$$\text{gap } \Delta := \min_{a \neq a^\star} \left(V^\star(s_1) - Q^\star(s_1, a)\right).$$

在 $B < \infty$ 的情況下，BRUE [8] 是一種基於軌跡的算法，該算法隨時可用，其樣本複雜度取決於最小的次優差距 $\Delta := \min(\ldots)$ 。

For planning in deterministic games, gap-dependent sample complexity bounds were previously provided in a fixed-confidence setting [16, 18].
對於確定性遊戲中的規劃，先前在固定置信度設置中提供了依賴於間隙的樣本複雜性界限 [16, 18]。

Our proposal, MDP-GapE, can be viewed as a non-trivial adaptation of the UGapE-MCTS algorithm [18] to planning in MDPs.
我們的提議，MDP-GapE，可以被視為 UGapE-MCTS 算法 [18] 對 MDP 中規劃的非平凡適應。

The defining property of MDP-GapE is that it uses a best arm identification algorithm, UGapE [10], to select

the first action in a trajectory, and performs optimistic planning thereafter, which helps refining confidence intervals on the intermediate Q-values.
MDP-GapE 的定義屬性是它使用最佳臂識別算法 UGapE [10] 來選擇軌跡中的第一個動作，然後執行樂觀規劃，這有助於細化中間 Q 值的置信區間。

Best arm identification tools have been previously used for planning in MDPs [23, 28] and UGapE also served as a building block for StOP [27].
最好的手臂識別工具以前已用於 MDP [23, 28] 中的規劃，而 UGapE 也用作 StOP [27] 的構建塊。

Finally, going beyond worse-case guarantees for RL is an active research direction, and in a different context gap-dependent bounds on the regret have recently been established for tabular MDPs [26, 29].
最後，超越 RL 的最壞情況保證是一個活躍的研究方向，並且在不同的上下文中，最近為表格 MDP 建立了基於間隙的遺憾邊界 [26, 29]。

Contributions 貢獻

We present MDP-GapE, a new MCTS algorithm for planning in the setting $B < \infty$.
我們提出了 MDP-GapE，這是一種新的 MCTS 算法，用於在 $B < \infty$ 的情況下進行規劃。

MDP-GapE performs efficient Monte-Carlo planning in the following sense:
MDP-GapE 在以下意義上執行高效的蒙特卡洛規劃：

First, it is a simple trajectory-based algorithm which performs well in practice and only relies on a forward model.
首先，它是一種簡單的基於軌跡的算法，在實踐中表現良好，僅依賴於前向模型。

Second, while most practical MCTS algorithms are not well understood theoretically, we prove upper bounds on the sample complexity of MDP-GapE.
其次，雖然大多數實際的 MCTS 算法在理論上還沒有得到很好的理解，但我們證明了 MDP-GapE 樣本複雜度的上限。

Our bounds depend on the sub-optimality gaps associated to the state-action pairs encountered during exploration.
我們的界限取決於與探索過程中遇到的狀態-動作對相關的次優差距。

This is in contrast to StOP and TrailBlazer, two algorithms for the same setting, whose guarantees depend on a notion of near-optimal nodes which can be harder to interpret, and that can be inefficient in practice.
這與 StOP 和 TrailBlazer 形成對比，兩種算法用於相同的設置，它們的保證取決於接近最優節點的概念，這可能更難解釋，並且在實踐中可能效率低下。

In the anytime setting, BRUE also features a gap-dependent sample complexity, but only through the worst-case gap Δ defined above.

在任何時間設置中，BRUE 還具有依賴於間隙的樣本複雜度，但只能通過上面定義的最壞情況間隙 Δ 。

As can be seen in Table 1, the upper bound for MDP-GapE given in Corollary 1 improves over that of BRUE as it features the gap of each possible first action a1, Δ1(s1, a1) = V*(s1) – Q*1(s1, a1), and scales better with the planning horizon H.

$$\Delta_1(s_1, a_1) = V^\star(s_1) - Q_1^\star(s_1, a_1)$$

從表 1 中可以看出，推論 1 中給出的 MDP-GapE 的上限比 BRUE 的上限有所提高，因為它具有每個可能的第一個動作 a1 的間隙，Δ1(s1, a1) = V*(s1) – Q*1(s1, a1)，並且隨著規劃範圍 H 擴展得更好。

Furthermore, our proof technique relates the pseudo-counts of any trajectory prefix to the gaps of state-action pairs on this trajectory, which evidences the fact that MDP-GapE does not explore trajectories uniformly.
此外，我們的證明技術將任何軌跡前綴的偽計數與該軌跡上的狀態-動作對的間隙相關聯，這證明了 MDP-GapE 沒有統一探索軌跡的事實。

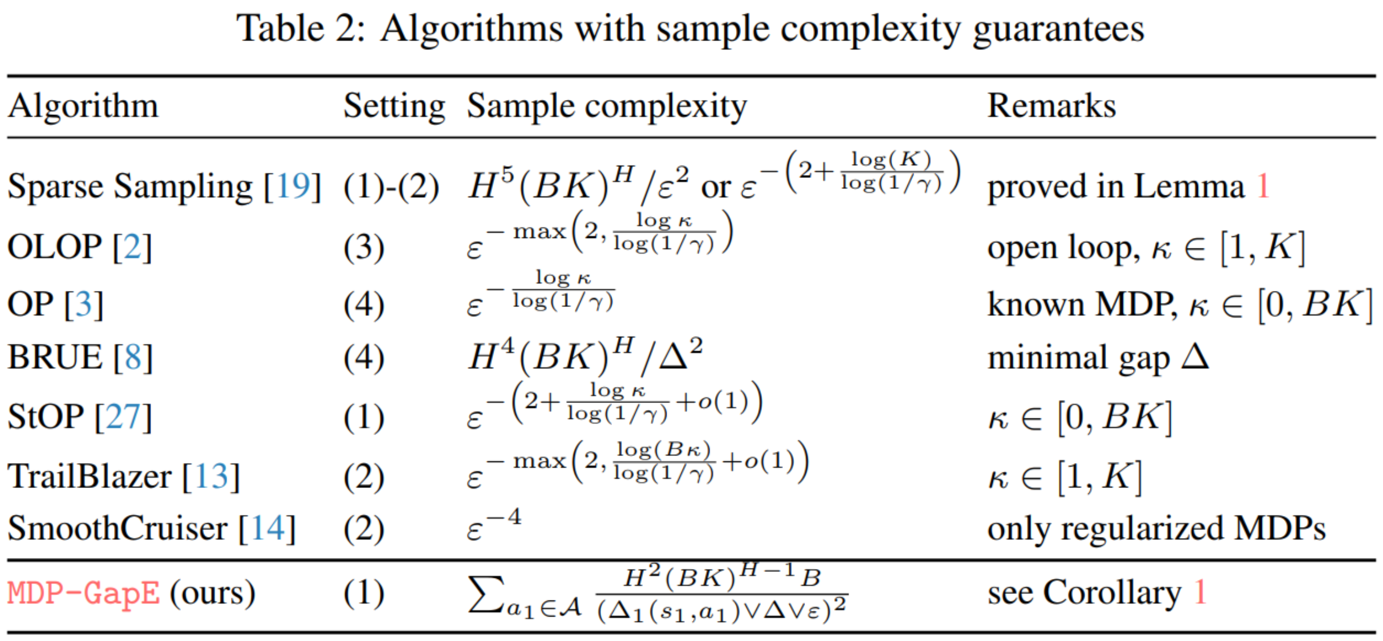


Table 2: Algorithms with sample complexity guarantees

Algorithm	Setting	Sample complexity	Remarks
Sparse Sampling [19]	(1)-(2)	$H^5(BK)^H/\varepsilon^2$ or $\varepsilon^{-\left(2+\frac{\log(K)}{\log(1/\gamma)}\right)}$	proved in Lemma 1
OLOP [2]	(3)	$\varepsilon^{-\max\left(2,\frac{\log\kappa}{\log(1/\gamma)}\right)}$	open loop, $\kappa \in [1, K]$
OP [3]	(4)	$\varepsilon^{-\frac{\log\kappa}{\log(1/\gamma)}}$	known MDP, $\kappa \in [0, BK]$
BRUE [8]	(4)	$H^4(BK)^H/\Delta^2$	minimal gap Δ
StOP [27]	(1)	$\varepsilon^{-\left(2+\frac{\log\kappa}{\log(1/\gamma)}+o(1)\right)}$	$\kappa \in [0, BK]$
TrailBlazer [13]	(2)	$\varepsilon^{-\max\left(2,\frac{\log(B\kappa)}{\log(1/\gamma)}+o(1)\right)}$	$\kappa \in [1, K]$
SmoothCruiser [14]	(2)	ε^{-4}	only regularized MDPs
MDP-GapE (ours)	(1)	$\sum_{a_1\in\mathcal{A}} \frac{H^2(BK)^{H-1}B}{(\Delta_1(s_1,a_1)\vee\Delta\vee\varepsilon)^2}$	see Corollary 1

Table 2: Algorithms with sample complexity guarantee
表 2：具有樣本複雜度保證的算法

2 Learning Framework and Notation 學習框架和符號

We consider a discounted episodic setting where H ∈ N* is a horizon and γ ∈ (0, 1] a discount parameter.
我們考慮一個折扣情景設置，其中 H ∈ N* 是一個範圍，而 γ ∈ (0, 1] 是一個折扣參數。

The transition kernels p = (p1, . . . , pH) and reward functions r = (r1, . . . , rH) can have distinct definitions in each step of the episode.
轉換核 p = (p1, . . , pH) 和獎勵函數 r = (r1, . . , rH) 在情節的每個步驟中都有不同的定義。

The optimal value of selecting action a in state s1 is Q*(s1, a) = max […], where the supremum is taken over (deterministic) policies π = (π1, . . . , πH), and the expectation is on a trajectory s1, a1, . . . , sh, ah where sh ∼ ph−1(·|sh−1, ah−1) and ah = πh(sh) for h ∈ [2, H].

$$Q^\star(s_1, a) = \max_\pi \mathbb{E}^\pi \left[\sum_{h=1}^{H} \gamma^{h-1} r_h(s_h, a_h) \middle| a_1 = a \right]$$

在狀態 s1 中選擇動作 a 的最優值是 Q*(s1, a) = max […]，其中 supremum 被接管（確定性）策略 π = (π1, . . , πH)，期望是 在軌跡 s1, a1, ... , sh, ah 其中 sh ∼ ph−1(·|sh−1, ah−1) 和 ah = πh(sh) 對於 h ∈ [2, H]。

With this definition, an optimal action in state s1 is a* ∈ argmaxa∈A(s1)Q* (s1, a).
根據這個定義，狀態 s1 中的最優動作是 a* ∈ argmaxa∈A(s1)Q* (s1, a)。

We assume that there is a maximal number K of actions available in each state, and that, for each (s, a), the support of ph(·|s, a) is bounded by B:
我們假設每個狀態都有最大數量的可用動作，並且對於每個 (s, a)，ph(·|s, a) 的支持度以 B 為界：

that is, B is the maximum number of possible next states when applying any action.
也就是說，B 是應用任何操作時可能的下一個狀態的最大數量。

We further assume that the rewards are bounded in [0, 1].
我們進一步假設獎勵在 [0, 1] 範圍內。

For each pair of integers i, h such that i ≤ h, we introduce the notation [i, h] = {i, . . . , h} and [h] = [1, h].
對於每對整數 i, h 使得 i ≤ h，我們引入符號 [i, h] = {i, . . . , h} 和 [h] = [1, h]。

(ε, δ)-correct planning
(ε, δ)-正確規劃

A sequential planning algorithm proceeds as follows.
順序規划算法如下進行。

In each episode t, the agent uses a deterministic policy on the form π t ={π t1, …, π tH} to generate a trajectory (s1, at1, …, stH, atH, rtH), where ath = π th(sth),rth is a reward with expectation rh(sth,ath) and sth+1~Ph(.|sth,ath).
在每個情節 t 中，代理使用形式為 π t ={π t1, ⋯, π tH} 的確定性策略來生成軌跡 (s1, at1, ⋯, stH, atH, rtH)，其中 ath = π th(sth),rth 是期望 rh(sth,ath) 和 sth+1~Ph(.|sth,ath) 的獎勵。

After each episode the agent decides whether it should perform a new episode to refine its guess for a near-optimal action, or whether it can stop and make a guess.
在每個情節之後，代理決定是否應該執行新情節以改進其對接近最佳動作的猜測，或者是否可以停

止並進行猜測。

We denote by τ the stopping rule of the agent, that is the number of episodes performed, and (ˆa)τ the guess.
我們用 τ 表示代理的停止規則，即執行的情節數，(ˆa) τ 表示猜測。

We aim to build an (ε, δ)-correct algorithm, that is an algorithm that outputs a guess (ˆa)τ satisfying while using as few calls to the generative model n = Hτ (i.e. as few episodes τ) as possible.
我們的目標是構建一個 (ε, δ)-correct 算法，該算法輸出滿足的猜測 (ˆa) τ ，同時使用盡可能少的對生成模型 n = H τ 的調用（即盡可能少的情節 τ ）。

$$\mathbb{P}\left(Q^\star(s_1,\hat{a}_\tau) > Q^\star(s_1,a^\star) - \varepsilon\right) \geq 1-\delta \quad \Leftrightarrow \quad \mathbb{P}\left(\bar{r}_{(H\tau)}(\hat{a}_\tau) \leq \varepsilon\right) \geq 1-\delta \tag{1}$$

Our setup permits to propose algorithms for planning in the undiscounted episodic case (in which our bounds will not blow up when γ = 1) and in discounted MDPs with infinite horizon.
我們的設置允許在未貼現的情節情況下（其中當 γ = 1 時我們的邊界不會爆炸）和具有無限視野的貼現 MDP 中提出規划算法。

Indeed, choosing H such that 2γ H/(1 − γ) ≤ ε, an (ε, δ)-correct algorithm for the discounted episodic setting recommends an action that is 2ε-optimal for the discounted infinite horizon setting.
事實上，選擇 H 使得 2γ H/(1 − γ) ≤ ε，一種用於折扣情景設置的 (ε, δ)-正確算法推薦了一個對於折扣無限範圍設置是 2ε-最優的動作。

A (recursive) baseline（遞歸）基線

Sparse Sampling [19] can be tuned to output a guess (ˆa) that satisfies (1), as specified in the following lemma, which provides a baseline for our undiscounted episodic setting (see Appendix F).
可以調整微細[19]以下輸出（1）呼吸的滿足引，如理中所指定，它為我們的未折現場景設置提供了靈感（参見附錄 F）。

Note that Sparse Sampling is not strictly sequential as it does not repeatedly select trajectories.
請注意，稀疏採樣不是嚴格順序的，因為它不會重複選擇軌跡。

Lemma 1. If B < ∞, Sparse Sampling using horizon H and performing O((H5/ε2)log(BK/δ)) transitions in each node is (ε, δ)-correct with sample complexity O(nSS) for nSS : = H5(BK)H/ε2.
引理 1. 如果 B < ∞，使用水平 H 並在每個節點中執行 O((H5/ ε 2)log(BK/ δ)) 轉換的稀疏採樣是 (ε, δ)-正確的，對於 nSS 的樣本複雜度為 O(nSS) : = H5(BK)H/ ε 2。

Structure of the optimal Q-value function

In our algorithm, we will build estimates of the intermediate Q-values, that are useful to compute the

optimal Q-value function Q*(s1, a).
在我們的算法中，我們將構建中間 Q 值的估計，這對於計算最佳 Q 值函數 Q*(s1, a) 很有用。

$$Q_h(s_h, a_h) = \max_{\pi} \mathbb{E}^{\pi}\left[\sum_{i=h}^{H} \gamma^{i-h} r(s_i, a_i) \middle| s_h, a_h\right]$$

Defining Q* (s1, a) = Q1(s1, a) and the optimal action-values Q = (Q1, …, QH) can be computed recursively using the Bellman equations, where we use the convention QH+1(·, ·) = 0:
定義 Q* (s1, a) = Q1(s1, a) 並且最佳動作值 Q = (Q1, ⋯, QH) 可以使用 Bellman 方程遞歸計算，其中我們使用約定 QH+1(·, ·) = 0:

$$Q_h(s_h, a_h) = r_h(s_h, a_h) + \gamma \sum_{s'} p_h(s'|s_h, a_h) \max_{a'} Q_{h+1}(s', a'), \quad h \in [H].$$

Let π* = (π*1, …, π*H)) denote a deterministic optimal policy where, for h ∈ [H], π*h(sh) = argmaxaQh(sh,a), with ties arbitrarily broken.
讓 π* = (π*1, …, π*H)) 表示確定性最優策略，其中，對於 h ∈ [H]，π *h(sh) = argmaxaQh(sh,a)，任意斷開關係。

Hence the optimal value in sh is Qh(sh, π*h(sh)).
因此 sh 中的最優值是 Qh(sh, π *h(sh))。

3. The MDP-GapE Algorithm MDP-GapE 算法

In this section we present MDP-GapE, a generalization of UGapE [10] to Monte-Carlo planning.
在本節中，我們介紹 MDP-GapE，這是 UGapE [10] 對蒙特卡洛規劃的概括。

Like BAI-MCTS for games [18] a core component is the construction of confidence intervals on Q1(s1, a).
就像遊戲的 BAI-MCTS [18] 一樣，一個核心組件是在 Q1(s1, a) 上構建置信區間。

The construction below generalizes that of OP-MDP [3] for known transition probabilities.
下面的構造將 OP-MDP [3] 的構造概括為已知轉移概率。

Confidence bounds on the Q-values - Q 值的置信界限

Our algorithm maintains empirical estimates, superscripted with the episode t, of the transition kernels p and expected rewards r, which are assumed unknown.

我們的算法維護了過渡內核 p 和預期獎勵 r 的經驗估計，上面標有情節 t，假設它們是未知的。

Let nth(sh, ah, sh+1) := be the number of observations of transition (sh, ah, sh+1), and Rth(sh, ah) := the sum of rewards obtained when selecting ah in sh.
設 nth(sh, ah, sh+1) :=… 是轉移 (sh, ah, sh+1) 的觀察次數，Rth(sh, ah) :=… 在 sh 中選擇 ah 時獲得的獎勵總和。

$$n_h^t(s_h, a_h, s_{h+1}) := \sum_{s=1}^t \mathbb{1}\left((s_h^s, a_h^s, s_{h+1}^s) = (s_h, a_h, s_{h+1})\right)$$

$$R_h^t(s_h, a_h) := \sum_{s=1}^t r_h^s(s_h, a_h)\mathbb{1}\left((s_h^s, a_h^s) = (s_h, a_h)\right)$$

We define the empirical transition probabilities (ˆp)t and expected rewards (ˆr)t as follows, for state-action pairs such that nth(sh, ah) := …:
我們將經驗轉移概率 (ˆp)t 和預期獎勵 (ˆr)t 定義如下，對於狀態-動作對，使得 nth(sh, ah) := …：

$$n_h^t(s_h, a_h) := \sum_s n_h^t(s_h, a_h, s) > 0$$

$$\hat{p}_h^t(s_{h+1}|s_h, a_h) := \frac{n_h^t(s_h, a_h, s_{h+1})}{n_h^t(s_h, a_h)}, \quad \text{and} \quad \hat{r}_h^t(s_h, a_h) := \frac{R_h^t(s_h, a_h)}{n_h^t(s_h, a_h)}.$$

As rewards are bounded in [0, 1], we define the following Kullback-Leibler upper and lower confidence bounds on the mean rewards rh(sh, ah) [4]:
由於獎勵在 [0, 1] 範圍內，我們定義了以下 Kullback-Leibler 對平均獎勵 rh(sh, ah) [4] 的置信上下限：

$$u_h^t(s_h, a_h) := \max\left\{v : \mathrm{kl}\left(\hat{r}_h^t(s_h, a_h), v\right) \leq \frac{\beta^r(n_h^t(s_h, a_h), \delta)}{n_h^t(s_h, a_h)}\right\},$$
$$\ell_h^t(s_h, a_h) := \min\left\{v : \mathrm{kl}\left(\hat{r}_h^t(s_h, a_h), v\right) \leq \frac{\beta^r(n_h^t(s_h, a_h), \delta)}{n_h^t(s_h, a_h)}\right\},$$

where βr is an exploration function and kl(u, v) is the binary Kullback-Leibler divergence between two Bernoulli distributions Ber(u) and Ber(v): kl(u, v) = u log (u/v)+(1-u)log((1-u)/(1-v)).
其中 βr 是探索函數，kl(u, v) 是兩個 Bernoulli 分佈 Ber(u) 和 Ber(v) 之間的二元 Kullback-Leibler 散度： kl(u, v) = u log (u/v)+(1-u)log((1-u)/(1-v))。

$$\mathcal{B}\mathrm{er}(v)\text{: } \mathrm{kl}(u, v) = u\log\frac{u}{v} + (1-u)\log\frac{1-u}{1-v}$$

We adopt the convention that uth(sh, ah) = 1, lth(sh, ah) = 0 when nth(sh, ah) = 0.
當 nth(sh, ah) = 0 時，我們採用 uth(sh, ah) = 1, lth(sh, ah) = 0 的約定。

$$u_h^t(s_h, a_h) = 1, \ell_h^t(s_h, a_h) = 0 \text{ when } n_h^t(s_h, a_h) = 0.$$

In order to define confidence bounds on the values Qh, we introduce a confidence set on the probability vector ph(·|sh, ah).
為了定義值 Qh 的置信界限，我們在概率向量 ph(·|sh, ah) 上引入了一個置信集。

We define Cth(sh, ah) = ΣB if nth(sh, ah) = 0 and otherwise Cth(sh, ah):={p....} where ΣB is the set of probability distribution over B elements, β p is an exploration function and KL(p, q) = Σs∈Supp(p)p(s) log(p(s)/q(s)) is the Kullback-Leibler divergence between two categorical distributions p and q with supports satisfying Supp(p) ⊆ Supp(q).

$$\mathcal{C}_h^t(s_h, a_h) := \left\{ p \in \Sigma_B : \mathrm{KL}\big(\widehat{p}_h^t(\cdot|s_h, a_h), p\big) \le \frac{\beta^p(n_h^t(s_h, a_h), \delta)}{n_h^t(s_h, a_h)} \right\}$$

$$\mathrm{KL}(p, q) = \sum_{s \in \mathrm{Supp}(p)} p(s) \log \frac{p(s)}{q(s)}$$

我們定義 Cth(sh, ah) = ΣB 如果 nth(sh, ah) = 0 否則 Cth(sh, ah):={p....} 其中 ΣB 是 B 元素上的概率分佈集，β p 是 探索函數和 KL(p, q) = Σs∈Supp(p)p(s) log(p(s)/q(s)) 是兩個分類分佈 p 和 q 之間的 Kullback-Leibler 散度，支持滿足 Supp(p) ⊆ 補充(q)。

We now define our confidence bounds on the action values inductively.
我們現在歸納地定義對動作值的置信界限。

We use the convention UtH+1(·, ·) = LtH+1(·, ·) = 0, and for all h ∈ [H], Uth, Lth.
我們使用約定 UtH+1(·, ·) = LtH+1(·, ·) = 0，並且對於所有 h ∈ [H], Uth, Lth。

$$U_h^t(s_h, a_h) = u_h^t(s_h, a_h) + \gamma \max_{p \in \mathcal{C}_h^t(s_h, a_h)} \sum_{s'} p(s'|s_h, a_h) \max_{a'} U_{h+1}^t(s', a'),$$

$$L_h^t(s_h, a_h) = \ell_h^t(s_h, a_h) + \gamma \min_{p \in \mathcal{C}_h^t(s_h, a_h)} \sum_{s'} p(s'|s_h, a_h) \max_{a'} L_{h+1}^t(s', a').$$

As explained in Appendix A of [9], optimizing over these KL confidence sets can be reduced to a linear program with convex constraints, that can be solved efficiently with Newton Iteration, which has complexity O(B log(d)) where d is the desired digit precision.
如 [9] 的附錄 A 所述，對這些 KL 置信集的優化可以簡化為具有凸約束的線性程序，可以使用牛頓迭代(Newton Iteration)有效地解決該問題，其複雜度為 O(B log(d))，其中 d 是 所需的數字精度。
We provide in Section 4.1 an explicit choice for the exploration functions βr(n, δ) and βp(n, δ) that govern the size of the confidence intervals.
我們在 4.1 節中為控制置信區間大小的探索函數 βr(n, δ) 和 βp(n, δ) 提供了一個明確的選擇。

Note that if the rewards or transitions are deterministic, or if we know p, we can adapt our confidence bounds by setting β p = 0 or β r = 0.

請注意，如果獎勵或轉換是確定性的，或者如果我們知道 p，我們可以通過設置 β p = 0 或 β r = 0 來調整我們的置信界限。

MDP-GapE

As any fixed-confidence algorithm, MDP-GapE depends on the tolerance parameter ε and the risk parameter δ.
與任何固定置信度算法一樣，MDP-GapE 取決於容差參數 ε 和風險參數 δ。

The dependency in ε is explicit in the stopping rule (4), while the dependency in δ is in the tuning of the confidence bounds, that depend on δ.
ε 中的依賴性在停止規則 (4) 中是明確的，而 δ 中的依賴性在於置信界限的調整，這取決於 δ。

After t trajectories observed, MDP-GapE selects the (t + 1)-st trajectory using the policy πt+1 =(πt+11, . . . , πt+1H) where the first action choice is made according to UGapE:
在觀察到 t 條軌跡後，MDP-GapE 使用策略 π t+1 =(π t+11, . . , π t+1H) 選擇 (t + 1)-st 軌跡，其中根據 UGapE 做出第一個動作選擇：

After t trajectories observed, MDP-GapE selects the $(t+1)$-st trajectory using the policy $\pi^{t+1} = (\pi_1^{t+1}, \dots, \pi_H^{t+1})$ where the first action choice is made according to UGapE:

$$\pi_1^{t+1}(s_1) = \operatorname*{argmax}_{b \in \{b^t, c^t\}} \left[U_1^t(s_1, b) - L_1^t(s_1, b) \right] ,$$

where bt is the current guess for the best action, which is the action b with the smallest upper confidence bound on its gap Q*1(s1, a*) − Q1(s1, b), and ct is some challenger:
其中 bt 是當前對最佳動作的猜測，即動作 b 的差距 Q*1(s1, a*) − Q1(s1, b) 的置信上限最小，而 ct 是一些挑戰者：

$$b^t = \operatorname*{argmin}_{b} \left[\max_{a \neq b} U_1^t(s_1, a) - L_1^t(s_1, b) \right], \qquad (2)$$
$$c^t = \operatorname*{argmax}_{c \neq b^t} U_1^t(s_1, c) . \qquad (3)$$

Then for all remaining steps we follow an optimistic policy, for all h ∈ [2, H], ...
然後對於所有剩餘的步驟，我們遵循樂觀策略，對於所有 h ∈ [2, H], ...

$$\pi_h^{t+1}(s_h) = \operatorname*{argmax}_{a} U_h^t(s_h, a).$$

The stopping rule of MDP-GapE is
MDP-GapE 的停止規則是

$$\tau = \inf\{t \in \mathbb{N} : U_1^t(s_1, c^t) - L_1^t(s_1, b^t) \leq \varepsilon\}, \qquad (4)$$

and the guess output when stopping is (ˆa)τ = bτ.
停止時的猜測輸出是(ˆa)τ = bτ.

A generic implementation of MDP-GapE is given in Algorithm 1 in Appendix A, where we also discuss some implementation details.
附錄 A 的算法 1 中給出了 MDP-GapE 的通用實現，我們還討論了一些實現細節。

Note that, in sharp contrast with the deterministic stopping rule proposed for Sparse Sampling in Lemma 1, MDP-GapE uses an adaptive stopping rule.
請注意，與引理 1 中為稀疏採樣提出的確定性停止規則形成鮮明對比的是，MDP-GapE 使用自適應停止規則。

4 Analysis of MDP-GapE - MDP-GapE 分析

Recall that MDP-GapE uses policy πt+1=… to select the (t + 1)-st trajectory, s1, at+1, …., satisfying at+1h = πt+1h … and st+1h+1 ∼ ph(.|..,..).
回想一下，MDP-GapE 使用策略 πt+1=… 來選擇 (t + 1)-st 軌跡，s1，at+1，⋯，滿足 at+1h = πt+1h … 和 st+1h+1 ∼ ph (.|..,..)。

Recall that MDP-GapE uses policy $\pi^{t+1} = (\pi_1^{t+1}, \ldots, \pi_H^{t+1})$ to select the $(t+1)$-st trajectory, $s_1, a_1^{t+1}, s_2^{t+1}, a_2^{t+1}, \ldots, s_H^{t+1}, a_H^{t+1}$, satisfying $a_h^{t+1} = \pi_h^{t+1}(s_h^{t+1})$ and $s_{h+1}^{t+1} \sim p_h\left(\cdot \middle| s_h^{t+1}, a_h^{t+1}\right)$.

High probability event - 高概率事件

To define an event E that holds with high probability, let Er(resp. Ep) be the event that the confidence regions for the mean rewards (resp. transition kernels) are correct:
為了定義一個以高概率成立的事件 E，讓 Er(resp. Ep) 是平均獎勵（或轉換核）的置信區域是正確的事件：

High probability event To define an event $\mathcal{E}$ that holds with high probability, let $\mathcal{E}^r$ (resp. $\mathcal{E}^p$) be the event that the confidence regions for the mean rewards (resp. transition kernels) are correct:

$$\mathcal{E}^r := \left\{\forall t \in \mathbb{N}^*, \forall h \in [H], \forall (s_h, a_h) \in \mathcal{S} \times \mathcal{A} :\ r_h(s_h, a_h) \in \left[\ell_h^t(s_h, a_h), u_h^t(s_h, a_h)\right]\right\},$$
$$\mathcal{E}^p := \left\{\forall t \in \mathbb{N}^*, \forall h \in [H], \forall (s_h, a_h) \in \mathcal{S} \times \mathcal{A} :\ p_h(\cdot|s_h, a_h) \in \mathcal{C}_h^t(s_h, a_h)\right\}.$$

For a state-action pair (sh, ah), let pπh(sh, ah) be the probability of reaching it at step h under policyπ, and let pth(sh, ah) = pπth(sh, ah).
對於狀態-動作對 (sh, ah)，令 p π h(sh, ah) 是在策略 π 下在步驟 h 到達它的概率，並令 pth(sh, ah) = p π th(sh, ah)。

We define the *pseudo-counts* of the number of visits of (sh, ah)as (¯n)th(sh, ah) := Σts=1 psh(sh, ah).
我們將 (sh, ah) 的訪問次數的偽計數定義(pseudo-counts)為 (¯n)th(sh, ah) := ∑ts=1 psh(sh, ah)。

As nth(sh, ah) – (¯n)th(sh, ah) is a martingale, the counts should not be too far from the pseudo-counts.

由於 nth(sh, ah) – (¯n)th(sh, ah) 是一個鞅，因此計數不應與偽計數相差太遠。

Given a rate function β cnt, we define the event ... .
給定速率函數 β cnt，我們定義事件 ... 。

$$\mathcal{E}^{\mathrm{cnt}} := \left\{ \forall t \in \mathbb{N}^{\star}, \forall h \in [H], \forall (s_h, a_h) \in \mathcal{S} \times \mathcal{A} :\ n_h^t(s_h, a_h) \geq \frac{1}{2}\bar{n}_h^t(s_h, a_h) - \beta^{\mathrm{cnt}}(\delta) \right\}$$

Finally, we define E to be the intersection of these three events: E = Er ∩ Ep ∩ Ecnt.
最後，我們定義 E 為這三個事件的交集：E = Er ∩ Ep ∩ Ecnt。

For a state-action pair (s_h, a_h), let $p_h^{\pi}(s_h, a_h)$ be the probability of reaching it at step h under policy π, and let $p_h^t(s_h, a_h) = p_h^{\pi^t}(s_h, a_h)$. We define the *pseudo-counts* of the number of visits of (s_h, a_h) as $\bar{n}_h^t(s_h, a_h) := \sum_{s=1}^{t} p_h^s(s_h, a_h)$. As $n_h^t(s_h, a_h) - \bar{n}_h^t(s_h, a_h)$ is a martingale, the counts should not be too far from the pseudo-counts. Given a rate function β^{cnt}, we define the event

$$\mathcal{E}^{\mathrm{cnt}} := \left\{ \forall t \in \mathbb{N}^{\star}, \forall h \in [H], \forall (s_h, a_h) \in \mathcal{S} \times \mathcal{A} :\ n_h^t(s_h, a_h) \geq \frac{1}{2}\bar{n}_h^t(s_h, a_h) - \beta^{\mathrm{cnt}}(\delta) \right\}.$$

Finally, we define $\mathcal{E}$ to be the intersection of these three events: $\mathcal{E} = \mathcal{E}^r \cap \mathcal{E}^p \cap \mathcal{E}^{\mathrm{cnt}}$.

4.1 Correctness 正確性

One can easily prove by induction (see Appendix B) that
可以很容易地通過歸納證明（見附錄 B）

$$\mathcal{E}^r \cap \mathcal{E}^p \subseteq \bigcap_{t \in \mathbb{N}^{\star}} \bigcap_{h=1}^{H} \left[\bigcap_{s_h, a_h} \left(Q_h(s_h, a_h) \in \left[L_h^t(s_h, a_h), U_h^t(s_h, a_h) \right] \right) \right].$$

As the arm (ˆa) output by MDP-GapE satisfies L1(s1, (ˆa)) > maxc≠(ˆa) U1(s1, c) − ε, on the event E ⊆ Er ∩ Ep it holds that Q1(s1, (ˆa)) > maxc≠(ˆa) Q1(s1, c) − ε.
由於 MDP-GapE 輸出的 arm (ˆa) 滿足 L1(s1, (ˆa)) > maxc≠(ˆa) U1(s1, c) − ε，在事件 E ⊆ Er ∩ Ep 認為 Q1(s1, (ˆa)) > maxc≠(ˆa) Q1(s1, c) − ε。

Thus MDP-GapE can only output an ε-optimal action.
因此 MDP-GapE 只能輸出一個 ε 最優動作。

Hence a sufficient condition for MDP-GapE to be (ε, δ)-correct is P(E) ≥ 1 − δ.
因此，MDP-GapE 為 (ε, δ) 正確的充分條件是 P(E) ≥ 1 − δ。

In Lemma 2 below, we provide a calibration of the thresholds functions βr, βp and β cnt such that this sufficient condition holds.
在下面的引理 2 中，我們提供了閾值函數 βr、βp 和 βcnt 的校準，使得這個充分條件成立。

This result, proved in Appendix C, relies on new time-uniform concentration inequalities that follow from the method of mixtures [7].
這一結果在附錄 C 中得到證明，它依賴於從混合方法 [7] 得出的新的時間均勻濃度不等式。

Lemma 2. 引理 2

Lemma 2. *For all* $\delta \in [0,1]$, *it holds that* $\mathbb{P}(\mathcal{E}) \geq 1-\delta$ *for the choices*

$$\beta^{r}(n,\delta) = \log(3(BK)^{H}/\delta) + \log\left(e(1+n)\right), \quad \beta^{\text{cnt}}(\delta) = \log\left(3(BK)^{H}/\delta\right),$$

$$\textit{and} \quad \beta^{p}(n,\delta) = \log\left(3(BK)^{H}/\delta\right) + (B-1)\log\left(e(1+n/(B-1))\right).$$

Moreover, the maximum of these three thresholds defined (by continuity when $B=1$*) as*

$$\beta(n,\delta) \;:=\; \max_{c\in\{r,p,\text{cnt}\}} \beta^{c}(n,\delta) = \log\left(3(BK)^{H}/\delta\right) + (B-1)\log\left(e(1+n/(B-1))\right),$$

is such that $n \mapsto \beta(n,\delta)$ *is non-decreasing and* $n \mapsto \beta(n,\delta)/n$ *is non-increasing.*

For all δ ..., it holds that P(ε) >= 1 – δ for the choices ... and ... .
Moreover, the maximum of these three thresholds defined (by continuity when B = 1) as ... is such that n 7 → β(n, δ) is non-decreasing and n → β(n, δ)/n is non-increasing.

對於所有 δ …，它認為 P(ε) >= 1 – δ 對於選擇 … 和 … 。
此外，這三個閾值中的最大值（由 B = 1 時的連續性）定義為 ... 使得 n 7→ β(n, δ) 不減少，n → β(n, δ)/n 不增加 .

4.2 Sample Complexity 樣本複雜度

In order to state our results, we define the following sub-optimality gaps.
為了說明我們的結果，我們定義了以下次優差距。

Δh(sh, ah) measures the gap in future discounted reward between the optimal action π*h(sh) and the action ah, whereas Δ*1(s1, a1) also takes into account the gap of the second best action and the tolerance level ε.
Δh(sh, ah) 衡量最佳動作 π*h(sh) 和動作 ah 之間未來折現獎勵的差距，而 Δ*1(s1, a1) 還考慮了第二個最佳動作的差距和 公差等級 ε。

Definition 1.(定義 1) Recall that Δ

Definition 1. *Recall that* $\Delta = \min_{a\neq a^\star}[Q_1(s_1,a^\star) - Q_1(s_1,a)]$. *For all* $h \in [H]$, *we let*

$$\begin{aligned} \Delta_h(s_h,a_h) &= Q_h(s_h,\pi_h^\star(s_h)) - Q_h(s_h,a_h), \\ \Delta_1^\star(s_1,a_1) &= \max\left(\Delta_1(s_1,a_1);\Delta;\varepsilon\right), \end{aligned}$$

and we denote $\tilde{\Delta}_h(s_h,a_h) = \begin{cases} \Delta_1^\star(s_h,a_h), & \textit{if } h=1, \\ \Delta_h(s_h,a_h), & \textit{if } h\geq 2. \end{cases}$

Our sample complexity bounds follow from the following crucial theorem, which we prove in Appendix D, that relates the pseudo-counts of state-action pairs at time τ to the corresponding gap.
我們的樣本複雜度界限遵循以下關鍵定理，我們在附錄 D 中證明了該定理，該定理將時間 τ 的狀態-動作對的偽計數與相應的間隙相關聯。

Introducing the constant $C_0 = (64\sqrt{2}(1+\sqrt{2}))^2$ and letting $c_\delta = \log\left(\frac{3(BK)^H}{\delta}\right)$, Lemma 12 stated in Appendix G permits to prove that, on the event $\mathcal{E}$, any (s_h, a_h) for which $\tilde{\Delta}_h(s_h, a_h) > 0$ satisfies

$$\bar{n}_h^\tau(s_h,a_h) \le \frac{C_0(BK)^{H-h}}{\tilde{\Delta}_h^2(s_h,a_h)}\left[c_\delta + 2(B-1)\log\left(\frac{C_0(BK)^{H-h}}{\tilde{\Delta}_h^2(s_h,a_h)}\left[\frac{c_\delta}{\sqrt{B-1}} + 2\sqrt{e(B-1)}\right]\right) + (B-1)\right] \quad (5)$$

As $\tilde{\Delta}_1(s_1,a_1) = \max(\Delta_1(s_1,a_1); \Delta; \varepsilon)$ is positive, the following corollary follows from summing the inequality over a_1, as $\bar{n}_1^\tau(s_1,a_1) = n_1^\tau(s_1,a_1)$ and $\tau = \sum_{a_1} n_1^\tau(s_1,a_1)$.

Theorem 1. If E holds, every (sh, ah) is such that (¯n)τh(sh, ah) Introducing the constant C0 = (64√ 2(1 + √ 2))2 and letting c δ = log (3(BK)H/ δ), Lemma 12 stated in Appendix G permits to prove that, on the event E, any (sh, ah) for which (˜ Δ)h(sh, ah) > 0 satisfies … (5)
定理 1. 如果 E 成立，則每個 (sh, ah) 使得 (¯n)τh(sh, ah) 引入常數 C0 = (64√ 2(1 + √ 2))2 並令 cδ = log (3 (BK)H/δ)，附錄 G 中的引理 12 允許證明，在事件 E 上，任何 (sh, ah) 滿足 (~ Δ)h(sh, ah) > 0 … (5)

As (˜ Δ)1(s1, a1) = max (Δ1(s1, a1); Δ; ε) is positive, the following corollary follows from summing the inequality over a1, as (¯n) τ 1(s1, a1) = n τ 1(s1, a1) and τ =Σa1 … .
由於 (~ Δ)1(s1, a1) = max (Δ1(s1, a1); Δ; ε) 為正，因此對 a1 上的不等式求和可得出以下推論，即 (¯n)τ1(s1, a1) = nτ1(s1, a1) 和 τ=Σa1 … .

Corollary 1. The number of episodes used by MDP-GapE satisfies P(…) >= 1-δ.
推論 1. MDP-GapE 使用的情節數量滿足 P(⋯) >= 1- δ 。

Corollary 1. *The number of episodes used by MDP-GapE satisfies*

$$\mathbb{P}\left(\tau = \mathcal{O}\left(\sum_{a_1} \frac{(BK)^{H-1}}{(\Delta_1(s_1,a_1)\vee\Delta\vee\varepsilon)^2}\left[\log\left(\frac{1}{\delta}\right) + BH\log(BK)\right]\right)\right) \ge 1-\delta\,.$$

The upper bound on the sample complexity n = Hτ of MDP-GapE that follows from Corollary 1 improves over the O(H5(BK)H/ε2) sample complexity of Sparse Sampling.
根據推論 1 得出的 MDP-GapE 的樣本複雜度 n = H τ 的上限改進了稀疏採樣的 O(H5(BK)H/ ε 2) 樣本複雜度。

It is also smaller than the O(H4(BK)H/Δ2) samples needed for BRUE to have a reasonable upper bound on its simple regret.
它也小於 BRUE 在其簡單的缺陷上具有檢測方法的 O(H4(BK)H/Δ2)樣本。

The improvement is twofold:
改進是雙重的：

first, this new bound features the problem dependent gap Δ(s1, a1) ∨ Δ ∨ ε for each action a1 in state s1, whereas previous bounds were only expressed with ε or Δ.
首先，對於狀態 s1 中的每個動作 a1，這個新邊界具有問題相關的間隙 Δ(s1, a1) ∨ Δ ∨ ε，而之前的

邊界僅用 ε 或 Δ 表示。

Second, it features an improved scaling in H2.
其次，它在 H2 中具有改進的縮放功能。

It is also possible to provide bounds that features the gaps (˜Δ)h(sh, ah) in the whole tree, beyond depth one.
還可以提供邊界，該邊界以整個樹中的間隙 (~Δ)h(sh, ah) 為特徵，超出深度 1。

To do so, we shall consider trajectories t1:H = (s1, a1, . . . , sH, aH) or trajectory prefixes t1:h = (s1, a1, . . . , sh, ah) for h ∈ [H].
還可以提供邊界，該邊界以整個樹中的間隙 (~Δ)h(sh, ah) 為特徵，超出深度 1。

Introducing the probability pπh(t1:h) that the prefix t1:h is visited under policy π, we can further define the pseudo-counts (¯n)th(t1:h) = Σts=1 p π sh(t1:h).
引入在策略 π 下訪問前綴 t1:h 的概率 p π h(t1:h)，我們可以進一步定義偽計數 (¯n)th(t1:h) = Σ ts=1 p π sh(t1:h)。

One can easily show that for all h ∈ [H], (¯n) τ h(t1:h) ≤ (¯n) τ h(t1:h) ≤ (¯n) τ h (sh, ah), if (sh, ah) is the state-action pair visited in step h in the trajectory t1:H, and (5) leads to the following upper bound.
可以很容易地證明，對於所有 h ∈ [H]，(¯n)τh(t1:h) ≤ (¯n)τh(t1:h) ≤ (¯n)τh (sh, ah)，如果 (sh, ah) 是在軌跡 t1:H 中的步驟 h 中訪問的狀態-動作對，並且 (5) 導致以下上限。

It is also possible to provide bounds that features the gaps $\tilde{\Delta}_h(s_h, a_h)$ in the *whole* tree, beyond depth one. To do so, we shall consider *trajectories* $t_{1:H} = (s_1, a_1, \ldots, s_H, a_H)$ or trajectory *prefixes* $t_{1:h} = (s_1, a_1, \ldots, s_h, a_h)$ for $h \in [H]$. Introducing the probability $p_h^\pi(t_{1:h})$ that the prefix $t_{1:h}$ is visited under policy π, we can further define the pseudo-counts $\bar{n}_h^t(t_{1:h}) = \sum_{s=1}^t p_h^{\pi^s}(t_{1:h})$. One can easily show that for all $h \in [H]$, $\bar{n}_H^\tau(t_{1:H}) \le \bar{n}_h^\tau(t_{1:h}) \le \bar{n}_h^\tau(s_h, a_h)$, if (s_h, a_h) is the state-action pair visited in step h in the trajectory $t_{1:H}$, and (5) leads to the following upper bound.

Corollary 2. On the event E,(推論 2. 在事件 E 上，)

Corollary 2. *On the event* $\mathcal{E}$, $\bar{n}_h^\tau(t_{1:h}) = \mathcal{O}\left(\left[\min_{\ell=1}^h \frac{(BK)^{H-\ell}}{(\tilde{\Delta}_\ell(s_\ell,a_\ell))^2}\right] \log\left(\frac{3(BK)^H}{\delta}\right)\right)$.

In particular, using that τ =Σt1:H∈T (¯n) τ h(t1:H) where T is the set of (BK)H complete trajectories leads to a sample complexity bound featuring all gaps.
特別是，使用 τ =Σt1:H∈T (¯n)τh(t1:H) 其中 T 是 (BK)H 完整軌跡的集合會導致樣本複雜度界限以所有間隙為特徵。

However, its improvement over the bound of Corollary 1 is not obvious in the general case.
然而，在一般情況下，它對 Corollary 1 邊界的改進並不明顯。

For B = 1, that is for planning in a deterministic MDP with possibly random rewards, a slightly different

proof technique leads to the following improved gap-dependent sample complexity bound (see the proof in Appendix E).
對於 B = 1，即為了在可能隨機獎勵的確定性 MDP 中進行規劃，略有不同的證明技術導致以下改進的依賴於間隙的樣本複雜度界限（參見附錄 E 中的證明）。

Theorem 2 (deterministic case). When B = 1, MDP-GapE satisfies
定理 2（確定性情況）。 當 B = 1 時，MDP-GapE 滿足

Theorem 2 (deterministic case). *When $B = 1$, MDP-GapE satisfies*

$$\mathbb{P}\left(\tau = \mathcal{O}\left(\sum_{t_{1:H}\in\mathcal{T}}\left[\min_{h=1}^{H}\frac{\left(\sum_{\ell=h}^{H}\gamma^{\ell}\right)^2}{\left(\tilde{\Delta}_h^2(s_h,a_h)\right)^2}\right]\left(\log\left(\frac{1}{\delta}\right)+H\log(K)\right)\right)\right)\geq 1-\delta.$$

Scaling in ε - ε 縮放

A majority of prior work on planning in MDPs has obtained sample complexity bounds that scale with ε only, in the discounted setting.
在 MDP 中規劃的大部分先前工作都獲得了僅隨 ε 縮放的樣本複雜度界限，在折扣設置中。

Neglecting the gaps, Corollary 1 gives a O(H2(BK)H/ε2) upper bound that yields a crude (˜O)(…) sample complexity in the discounted setting in which H ~ log(1/ε)/ log(1/γ).
忽略這些差距，推論 1 給出了一個 O(H2(BK)H/ε2) 上限，該上限在折扣設置中產生粗略的 (~O)(…) 樣本複雜度，其中 H ~ log(1/ε)/ log(1/γ)。

This exponent is larger than that in previous work, which features some notion of near-optimality dimension κ (see Table 1).
這個指數比之前的工作中的指數大，它具有一些接近最優維度 κ 的概念（見表 1）。

However, our analysis was not tailored to optimizing this exponent, and we show in Section 5 that the empirical scaling of MDP-GapE in ε can be much smaller than the one prescribed by the above crude bound.
然而，我們的分析並不是為了優化這個指數而定制的，我們在第 5 節中表明，ε 中 MDP-GapE 的經驗標度可能比上述粗邊界規定的小得多。

5. Numerical Experiments - 數值實驗

We consider random discounted MDPs with infinite horizon in which the maximal number B of successor states and the sparsity of rewards are controlled.
我們考慮具有無限範圍的隨機折扣 MDP，其中最大的後繼狀態 B 和獎勵的稀疏性受到控制。

The transition kernel is generated as follows: for each transition in S ×A, we uniformly pick B next states in

S.
轉換核生成如下：對於 S ×A 中的每個轉換，我們統一選擇 S 中的 B 個下一個狀態。

The cumulative transition probabilities to these states are computed by sorting B − 1 numbers uniformly sampled in (0, 1).
這些狀態的累積轉移概率是通過對 (0, 1) 中均勻採樣的 B-1 個數字進行排序來計算的。

The reward kernel is computed by selecting a proportion of the transitions to have non-zero rewards with means sampled uniformly in (0, 1).
獎勵內核是通過選擇一定比例的轉換來計算的，以獲得非零獎勵，均值在 (0, 1) 中均勻採樣。

The values for these parameters are shown in Table 3a.
這些參數的值顯示在表 3a 中。

Fixed-confidence: Correction and sample complexity - 固定置信度：校正和樣本複雜度

We verify empirically that MDP-GapE is (ε, δ)-correct while stopping with a reasonable number of oracle calls.
我們憑經驗驗證 MDP-GapE 是 (ε, δ) 正確的，同時以合理數量的 oracle 調用停止。

Table 3b shows the choice of parameters for the algorithm.
表 3b 顯示了算法參數的選擇。

For various values of the desired accuracy ε and of the corresponding planning horizon H = [logγ (ε(1−γ)/2)] (see Section 2), we run simulations on 200 random MDPs.
對於所需精度 ε 的各種值和相應的規劃範圍 H = [logγ (ε(1−γ)/2)]（參見第 2 節），我們在 200 個隨機 MDP 上運行模擬。

We report in Table 4 the distribution of the number n = τH of oracle calls and the simple regret r¯n(ˆan) of MDP-GapE over these 200 runs.
我們在表 4 中報告了在這 200 次運行中，oracle 調用的數量 n = τH 和 MDP-GapE 的簡單遺憾 r¯n(ˆan) 的分佈。

We first observe that MDP-GapE verifies (¯r)n((ˆa)n) < ε in all simulations, despite the use of smaller exploration functions compared to those prescribed in Lemma 2.
我們首先觀察到 MDP-GapE 在所有模擬中驗證 (¯r)n((ˆa)n) < ε，儘管與引理 2 中規定的探索函數相比使用了更小的探索函數。

Table 3: Experimental setting. 實驗設置

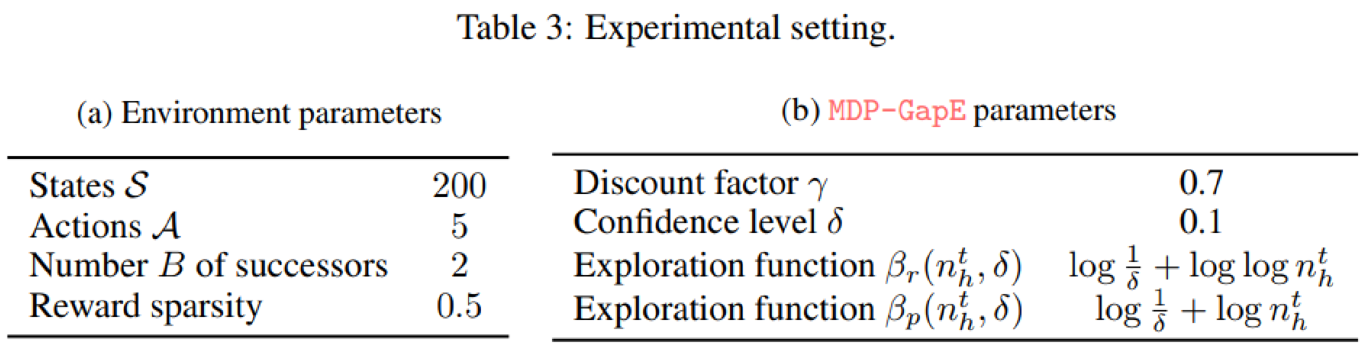


Table 3: Experimental setting.

(a) Environment parameters

States $\mathcal{S}$	200
Actions $\mathcal{A}$	5
Number B of successors	2
Reward sparsity	0.5

(b) MDP-GapE parameters

Discount factor γ	0.7
Confidence level δ	0.1
Exploration function $\beta_r(n_h^t, \delta)$	$\log\frac{1}{\delta} + \log\log n_h^t$
Exploration function $\beta_p(n_h^t, \delta)$	$\log\frac{1}{\delta} + \log n_h^t$

We then compare its sample complexity to that of *Sparse Sampling*, which is deterministic and for which nSS given in Lemma 1 is a tight upper bond.
然後我們將它的樣本複雜度與稀疏採樣的複雜度進行比較，稀疏採樣是確定性的，並且引理 1 中給出的 nSS 是一個緊密的上鍵。

We see that the sample complexity of MDP-GapE is an order of magnitude smaller than that of Sparse Sampling.
我們看到 MDP-GapE 的樣本複雜度比 Sparse Sampling 小一個數量級。

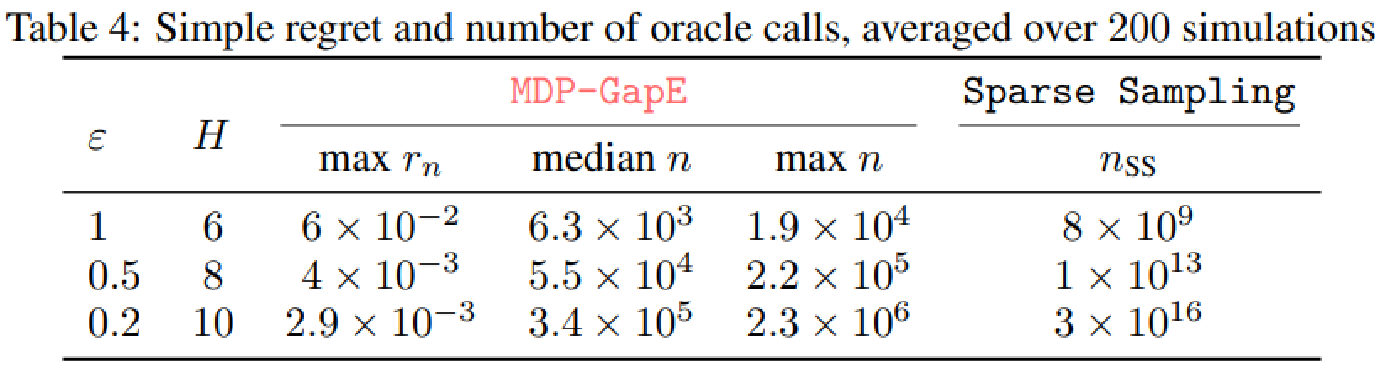


Table 4: Simple regret and number of oracle calls, averaged over 200 simulations

ε	H	MDP-GapE			Sparse Sampling
		max r_n	median n	max n	n_{SS}
1	6	6×10^{-2}	6.3×10^{3}	1.9×10^{4}	8×10^{9}
0.5	8	4×10^{-3}	5.5×10^{4}	2.2×10^{5}	1×10^{13}
0.2	10	2.9×10^{-3}	3.4×10^{5}	2.3×10^{6}	3×10^{16}

Table 4: Simple regret and number of oracle calls, averaged over 200 simulations
表 4：簡單的遺憾和預言機調用次數，平均超過 200 次模擬

Scaling in ε - ε 中的縮放

As discussed above, Corollary 1 with the aforementioned choice of the planning horizon, yields a crude sample complexity bound on the (˜O)(…)=… in our experimental setting.
如上所述，推論 1 與上述規劃範圍的選擇，在我們的實驗設置中產生了一個粗略的樣本複雜度限制在 (~O) 上。

$$\tilde{\mathcal{O}}\left(\varepsilon^{-[2+\log(BK)/\log(1/\gamma)]}\right) = \tilde{\mathcal{O}}\left((1/\varepsilon)^{8.4}\right)$$

However, we observe that the empirical exponent can be much smaller in practice: plotting the average sample complexity n (estimated over the 200 MDPs) as a function of log(1/ε) in Figure 1 and measuring the slope of the curve yields n … O.
然而，我們觀察到經驗指數在實踐中可能要小得多：將平均樣本複雜度 n（在 200 個 MDP 上估

計）繪製為圖 1 中 log(1/ ε) 的函數，並測量曲線的斜率，得出 n .. 。

$$n \simeq \mathcal{O}\big((1/\varepsilon)^{3.9} \big)$$

Comparison to the state of the art - **與最先進技術的比較**

In the fixed-confidence setting, most existing algorithms are considered theoretical and cannot be applied to practical cases.
在固定置信度設置中，大多數現有算法都被認為是理論上的，不能應用於實際案例。

For instance, for our problem with K = 5 and ε = 1, Sparse Sampling [19] and SmoothCruiser [14] both require a fixed budget1 of at least nSS = 8 × 10^9 .
例如，對於 K = 5 和 ε = 1 的問題，稀疏採樣 [19] 和 SmoothCruiser [14] 都需要至少 nSS = 8 × 10^9 的固定預算 1。

Likewise, Trailblazer [13] is a recursive algorithm which did not terminate in our setting.
同樣，Trailblazer [13] 是一種遞歸算法，它不會在我們的設置中終止。

We did not implement StOP [27] as it requires to store a tree of policies, which is very costly even for moderate horizons.
我們沒有實施 StOP [27]，因為它需要存儲一棵策略樹，即使對於中等水平，這也是非常昂貴的。

In comparison, Table 4 shows that MDP-GapE stopped after n = 1.9 × 10^4 oracle calls in the worst case.
相比之下，表 4 顯示 MDP-GapE 在最壞情況下在 n = 1.9 × 10^4 次預言機調用後停止。

To the best of our knowledge, MDP-GapE is the first (ε, δ)-correct algorithm for general MDPs with an easy implementation and a reasonable running time in practice.
據我們所知，MDP-GapE 是第一個適用於一般 MDP 的（ε， δ）校正算法，在實踐中具有簡單的實現和合理的運行時間。

The only planning algorithms that can be run in practice are in the fixed-budget setting, which we now consider.
唯一可以在實踐中運行的規划算法是在我們現在考慮的固定預算設置中。

Fixed-budget evaluation - **固定預算評估**

We compare MDP-GapE to three existing baselines: first, the KL-OLOP algorithm [21], which uses the same upper-confidence bounds on the rewards uth and states values Uth as MDP-GapE, but is restricted to open-loop policies, i.e. sequences of actions only.
我們將 MDP-GapE 與三個現有的基線進行比較：首先，KL-OLOP 算法 [21]，它使用與 MDP-GapE 相同的獎勵 uth 和狀態值 Uth 的置信上限，但僅限於開環策略 ，即僅動作序列。

Second, the BRUE algorithm [8] which explores uniformly and handles closed-loop policies.
其次，BRUE 算法 [8] 統一探索並處理閉環策略。

Third, the popular UCT algorithm [20], which is also closed-loop and performs optimistic exploration at all depths.
第三，流行的 UCT 算法[20]，它也是閉環的，在所有深度進行樂觀探索。

UCT and its variants lack theoretical guarantees, but they have been shown successful empirically in many applications.
UCT 及其變體缺乏理論上的保證，但它們已在許多應用中憑經驗證明是成功的。

For each algorithm, we tune the planning horizon H similarly to KL-OLOP, by dividing the available budget n into τ episodes, where τ is the largest integer such that $\tau \log \tau /(2 \log 1/\gamma) \le n$, and choose $H = \log \tau /(2 \log 1/\gamma)$.
對於每個算法，我們類似於 KL-OLOP，通過將可用預算 n 劃分為 τ 個片段來調整規劃範圍 H，其中 τ 是最大整數，使得 $\tau \log \tau /(2 \log 1/\gamma) \le n$，並選擇 $H = \log \tau /(2 \log 1/\gamma)$。

The exploration functions are those of KL-OLOP and depend on τ : $\beta r(nth, \delta) = \beta p(nth, \delta) = \log(\tau)$.
探索函數是 KL-OLOP 的函數，取決於 τ ： $\beta r(nth, \delta) = \beta p(nth, \delta) = \log(\tau)$。

Again, we perform 200 simulations and report in Figure 2 the mean simple regret, along with its 95% confidence interval.
同樣，我們進行了 200 次模擬，並在圖 2 中報告了平均簡單後悔及其 95% 的置信區間。

We observe that MDP-GapE compares favourably with these baselines in the high-budget regime.
我們觀察到 MDP-GapE 在高預算制度中與這些基線相比具有優勢。

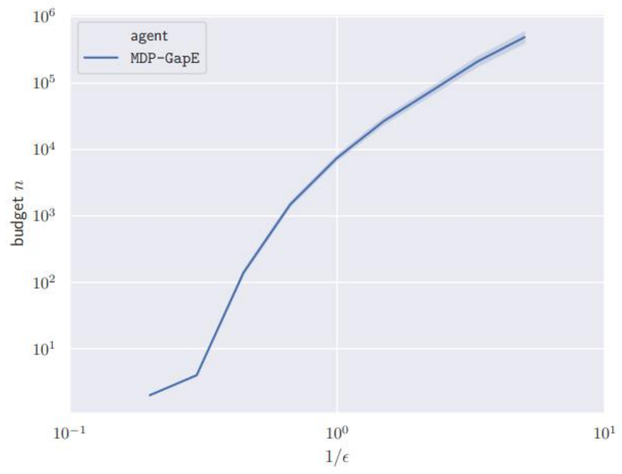


Figure 1: Polynomial dependency of the number n of oracle calls with respect to $1/\varepsilon$.

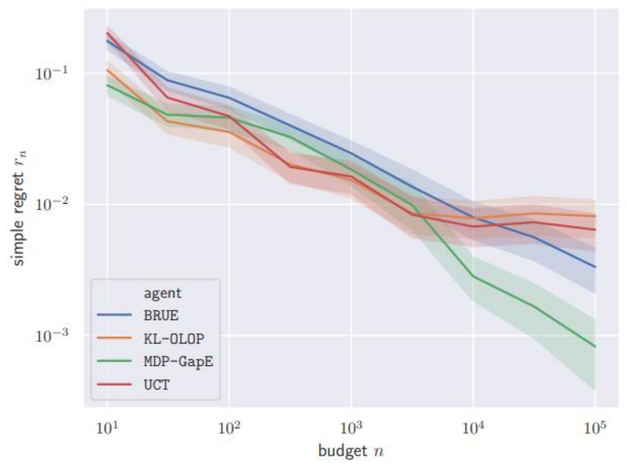


Figure 2: Comparison to KL-OLOP in a fixed-budget setting.

Figure 1: Polynomial dependency of the number n of oracle calls with respect to 1/ε.

圖 1：oracle 調用次數 n 與 1/ε 的多項式相關性。

Figure 2: Comparison to KL-OLOP in a fixedbudget setting.
圖 2：在固定預算設置中與 KL-OLOP 的比較。

6 Conclusion 結論

We proposed a new, efficient algorithm for Monte-Carlo planning in Markov Decision Processes, that combines tools from best arm identification and optimistic planning and exploits tight confidence regions on mean rewards and transitions probabilities.
我們為馬爾可夫決策過程中的蒙特卡洛規劃提出了一種新的、有效的算法，該算法結合了最佳臂識別和樂觀規劃的工具，並利用了平均獎勵和轉移概率的緊密置信區域。

We proved that MDP-GapE attains the smallest existing gap-dependent sample complexity bound for general MDPs with stochastic rewards and transitions, when the branching factor B is finite.
我們證明了當分支因子 B 是有限的時，MDP-GapE 獲得了具有隨機獎勵和轉換的一般 MDP 的最小現有間隙相關樣本複雜度界限。

In future work, we will investigate the worse-case complexity of MDP-GapE, that is try to derive an upper bound on its sample complexity that only features ε and some appropriate notion of near-optimality dimension.
在未來的工作中，我們將研究 MDP-GapE 的最壞情況復雜度，即嘗試推導出其樣本複雜度的上限，該上限僅包含 ε 和一些近似最優維度的適當概念。

Acknowledgments 致謝

Anders Jonsson is partially supported by the Spanish grants TIN2015-67959 and PCIN-2017-082.
Anders Jonsson 得到了西班牙贈款 TIN2015-67959 和 PCIN-2017-082 的部分支持。

A Detailed Algorithm

In this section we provide a detailed algorithm for MDP-GapE, namely Algorithm 1.

Algorithm 1 MDP-GapE

```
1: Input: confidence level $\delta$, tolerance $\varepsilon$
2: initialize data lists $\mathcal{D}_h \leftarrow [\,]$ for all $h \in [H]$
3: for $t = 1 \ldots$ do
4:     //Update confidence bounds
5:     $U_h^{t-1}, L_h^{t-1} \leftarrow \texttt{UpdateBounds}(t, \delta, \mathcal{D}_h)$
6:     if $U_1^{t-1}(s_1, c^t) - L_1^{t-1}(s_1, b^t) \le \varepsilon$ then
7:         return $b_{t-1}$, break
8:     end if
9:     // Best
10:    $b^{t-1} \leftarrow \operatorname*{argmin}_b \left[\max_{a \neq b} U_1^{t-1}(s_1, a) - L_1^{t-1}(s_1, b)\right]$
11:    //Challenger
12:    $c^{t-1} \leftarrow \operatorname*{argmax}_{c \neq b^t} U_1^{t-1}(s_1, c)$
13:    //Exploration
14:    $a_1^t \leftarrow \operatorname*{argmax}_{a \in \{b^{t-1}, c^{t-1}\}} \left[U_1^{t-1}(s_1, a) - L_1^{t-1}(s_1, a)\right]$
15:    observe reward $r_1^t$, next state $s_2^t$, save $\mathcal{D}_1$.append$(s_1^t, a_1^t, s_2^t, r_1^t)$
16:    for step $h = 2, \ldots, H$ do
17:        $a_h^t \leftarrow \operatorname*{argmax}_a U_h^{t-1}(s_h^t, a)$
18:        observe reward $r_{h-1}^t$, next state $s_h^t$,save $\mathcal{D}_h$.append$(s_h^t, a_h^t, s_{h+1}^t, r_h^t)$
19:    end for
20: end for
```

Implementation details There are different ways to store and update the confidence bounds on the Q-value (that is, to specify the `UpdateBounds` subroutine) according to how we merge information across states.

The most obvious one, suggested by previous work [2, 21, 3] (and also implemented for our experiments) does not merge information at all and builds a search *tree* in which a node (s_h, a_h) at depth h is identified with the sequence of h states and actions that leads to it. It leads to a very simple update: after each trajectory, one only needs to update the confidence bounds, $U_h(s_h, a_h)$ and $L_h(s_h, a_h)$, of the visited action-state pairs. Another option is to merge information for the same states and a fixed depth. But in this case the search tree becomes a *graph* and after each trajectory we need to re-compute the values $U_h(s_h, a_h)$ for all stored state action pairs (s_h, a_h) at each depth.