Private Reinforcement Learning with PAC and Regret Guarantees

Giuseppe Vietri[*] Borja Balle[†] Akshay Krishnamurthy[‡] Zhiwei Steven Wu[§]

Introduction Privacy-preserving machine learning is critical to the deployment of data-driven solutions in applications involving sensitive data. Differential privacy (DP) [DMNS06] is a de-facto standard for designing algorithms with strong privacy guarantees for individual data. Large-scale industrial deployments – e.g. by Apple [Tea17], Google [EPK14] and the US Census Bureau [Abo18] – and general purpose DP tools for machine learning [ACP19] and data analysis [HBAL19, WZL$^+$19] exemplify that existing methods are well-suited for simple data analysis tasks (e.g. averages, histograms, frequent items) and batch learning problems where the training data is available beforehand. While these techniques cover a large number of applications in the central and (non-interactive) local models, they are often insufficient to tackle machine learning applications involving other threat models. This includes federated learning problems [KMA$^+$19, LSTS19] where devices cooperate to learn a joint model while preserving their individual privacy, and, more generally, interactive learning in the spirit of the reinforcement learning (RL) framework [SB18].

In this paper we contribute to the study of reinforcement learning from the lens of differential privacy. We consider sequential decision-making tasks where users interact with an agent for the duration of a fixed-length episode. At each time-step the current user reveals a state to the agent, which responds with an appropriate action and receives a reward generated by the user. Like in standard RL, the goal of the agent is to learn a policy that maximizes the rewards provided by the users. However, our focus is on situations where the states and rewards that users provide to the agent might contain sensitive information. While users might be ready to reveal such information to an agent in order to receive a service, we assume they want to prevent third parties from making unintended inferences about their personal data. This includes external parties who might have access to the policy learned by the agent, as well as malicious users who can probe the agent's behavior to trigger actions informed by its interactions with previous users. For example, [PWZ$^+$19] recently showed how RL policies can be probed to reveal information about the environment where the agent was trained.

The question we ask in this paper is: how should the learnings an agent can extract from an episode be balanced against the potential information leakages arising from the behaviors of the agent that are informed by such learnings? We answer the question by making two contributions to the analysis of the privacy-utility trade-off in reinforcement learning: (1) we provide the first privacy-preserving RL algorithm with formal accuracy guarantees, and (2) we provide lower bounds on the regret and number of sub-optimal episodes for any differentially private RL algorithm. To measure the privacy provided by episodic RL algorithms we introduce a notion of episodic joint differential privacy (JDP)[5] under

[*]Department of Computer Science and Engineering, University of Minnesota. Supported by the GAANN fellowship from the U.S. Department of Education. Email: vietr002@umn.edu

[†]Now at DeepMind. Email: borja.balle@gmail.com

[‡]Microsoft Research, New York, NY. Email: akshaykr@microsoft.com

[§]School of Computer Science, Carnegie Mellon University. Email: zstevenwu@cmu.edu

[5]see appendix A for formal definition of JDP

continuous observation, a variant of joint differential privacy [KPRU14] that captures the potential information leakages discussed above.

Overview of our results. We study reinforcement learning in a fixed-horizon episodic Markov decision process with S states, A actions, and episodes of length H. We first provide a meaningful privacy formulation for this general learning problem with a strong relaxation of differential privacy: joint differential privacy (JDP) under continual observation, controlled by a privacy parameter $\varepsilon \geq 0$ (larger ε means less privacy). Under this formulation, we give the first known RL sample complexity and regret upper and lower bounds with formal privacy guarantees. First, we present algorithm 2, Public Upper Confidence Bound(PUCB), which satisfies ε-JDP in addition to two utility guarantees: it finds an α-optimal policy with a sample complexity of $\widetilde{O}\left(\frac{SAH^4}{\alpha^2} + \frac{S^2AH^4}{\varepsilon\alpha}\right)$, and achieves a regret rate of $\widetilde{O}\left(H^2\sqrt{SAT} + \frac{SAH^3+S^2AH^3}{\varepsilon}\right)$ over T episodes. In both of these bounds, the first terms $\frac{SAH^4}{\alpha^2}$ and $H^2\sqrt{SAT}$ are the non-private sample complexity and regret rates, respectively. The privacy parameter ε only affects the lower order terms – for sufficiently small approximation α and sufficiently large T, the "cost" of privacy becomes negligible.

We also provide new lower bounds for ε-JDP reinforcement learning. Specifically, by incorporating ideas from existing lower bounds for private learning into constructions of hard MDPs, we prove a sample complexity bound of $\widetilde{\Omega}\left(\frac{SAH^2}{\alpha^2} + \frac{SAH}{\varepsilon\alpha}\right)$ and a regret bound of $\widetilde{\Omega}\left(\sqrt{HSAT} + \frac{SAH}{\varepsilon}\right)$. As expected, these lower bounds match our upper bounds in the dominant term (ignoring H and polylogarithmic factors). We also see that necessarily the utility cost for privacy grows linearly with the state space size, although this does not match our upper bounds. Closing this gap is an important direction for future work.

The PUCB Algorithm In this section, we introduce the Private Upper Confidence Bound algorithm (PUCB), a JDP algorithm with both PAC and regret guarantees. The pseudo-code for PUCB is in algorithm 2. At a high level, the algorithm is a private version of the UBEV algorithm [DLB17]. UBEV keeps track of three types of event statistics, $\widehat{r}_t(s,a,h), \widehat{n}_t(s,a,h), \widehat{m}_t(s,a,s',h)$ (see appendix A.3 for counters description) to compute a policy for each round t. These counters don't satisfy JDP, hence we have to use private versions of these event counters which we denote by $\widetilde{r}_t, \widetilde{n}_t, \widetilde{m}_t$. We implement each private counter using the binary mechanism due to [DNPR10, CSS11]. Since we are adding extra noise to the counters, our algorithm incurs additional error, however we can bound the error of each counter as: $\forall t \in [T] : |\widetilde{n}_t(s,a,h) - \widehat{n}_t(s,a,h)| < E_\varepsilon$ [6], where $\widehat{n}_t, \widetilde{n}_t$ are the count and release at the beginning of the t-th episode. The guarantee is uniform in (s,a,h) and also holds simultaneously for $\widetilde{r}$ and $\widetilde{m}$.

To compute the policy, we define a bonus function $\widetilde{\text{conf}}(s,a,h)$ for each (s,a,h) tuple, which can be decomposed into two parts $\widetilde{\phi}_t(s,a,h)$ and $\widetilde{\psi}_t(s,a,h)$[7]. The term $\widetilde{\phi}_t(\cdot)$ roughly corresponds to the sampling error, while $\widetilde{\psi}_t(\cdot)$ corresponds to errors introduced by the private counters. Using this bonus function, we use dynamic programming to compute an optimistic private Q-function in Algorithm 3. The algorithm here is a standard batch Q-learning update, with $\widetilde{\text{conf}}(\cdot)$ serving as an optimism bonus. The resulting Q-function, called $\widetilde{Q}^+$, encodes a greedy policy, which we use for the t-th episode.

Conclusion In this paper, we provide a JDP algorithm and establish both PAC and regret utility guarantees for episodic tabular MDPs. Our results show that the utility cost for privacy is asymptotically negligible in the large accuracy regime. We also establish the first lower bounds for reinforcement learning with JDP. Beyond the tabular setup considered in this paper, we believe that designing RL algorithms providing state and reward privacy in non-tabular settings is a promising direction for future work with considerable potential for real-world applications.

[6] $E_\varepsilon := \frac{3}{\varepsilon} H \log\left((2SAH + S^2AH)(\beta')^{-1}\right) \log(T)^{5/2}$

[7] $\widetilde{\phi}_t(s,a,h) := \sqrt{\frac{2\ln(T/\beta')}{\max(\widetilde{n}_t(s,a,h) - E_\varepsilon, 1)}}$ and $\widetilde{\psi}_t(s,a,h) := (1+SH)\left(\frac{3E_\varepsilon}{\widetilde{n}_t(s,a,h)} + \frac{2E_\varepsilon^2}{\widetilde{n}_t(s,a,h)^2}\right)$

References

[Abo18] John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

[ACP19] Galen Andrew, Steve Chien, and Nicolas Papernot. Tensorflow privacy. `https://github.com/tensorflow/privacy`, 2019.

[CSS11] T-H Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):26, 2011.

[DLB17] Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 5713–5723, 2017.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724. ACM, 2010.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[HBAL19] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: The IBM differential privacy library. *CoRR*, abs/1907.02444, 2019.

[KMA$^+$19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2019.

[KPRU14] Michael J. Kearns, Mallesh M. Pai, Aaron Roth, and Jonathan Ullman. Mechanism design in large games: incentives and privacy. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 403–410, 2014.

[LSTS19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions, 2019.

[PWZ$^+$19] Xinlei Pan, Weiyao Wang, Xiaoshuai Zhang, Bo Li, Jinfeng Yi, and Dawn Song. How you act tells a lot: Privacy-leaking attack on deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, pages 368–376, 2019.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SS18] Roshan Shariff and Or Sheffet. Differentially private contextual linear bandits. In *Advances in Neural Information Processing Systems*, pages 4296–4306, 2018.

[Tea17] Apple Differential Privacy Team. Learning with privacy at scale. `https://machinelearning.apple.com/2017/12/06/learning-with-privacy-at-scale.html`, 2017.

[WZL$^+$19] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private sql with bounded user contribution, 2019.

A Preliminaries

A.1 Markov Decision Processes

A fixed-horizon *Markov decision process* (MDP) with time-dependent dynamics can be formalized as a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0, H)$. $\mathcal{S}$ is the state space with cardinality S. $\mathcal{A}$ is the action space with cardinality A. $\mathcal{R}(s_h, a_h, h)$ is the reward distribution on the interval $[0,1]$ with mean $r(s_h, a_h, h)$. $\mathcal{P}$ is the transition kernel, given time step h, action a_h and, state s_h the next state is sampled from $s_{t+1} \sim \mathcal{P}(.|s_h, a_h, h)$. Let p_0 be the initial state distribution at the start of each episode, and H be the number of time steps in an episode.

In our setting, an agent interacts with an MDP by following a (deterministic) policy $\pi \in \Pi$, which maps states s and timestamps h to actions, i.e., $\pi(s,h) \in \mathcal{A}$. The *value function* in time step $h \in [H]$ for a policy π is defined as:

$$\begin{aligned} V_h^\pi(s) &= \mathbb{E}\left[\sum_{i=h}^{H} r(s_i, a_i, i) \middle| s_h = s, \pi\right] \\ &= r(s, \pi(s,h), h) + \sum_{s' \in \mathcal{S}} V_{h+1}^\pi(s') \mathcal{P}(s'|s, \pi(s,h), h) \ . \end{aligned}$$

The *expected total reward* for policy π during an entire episode is:

$$\rho^\pi = \mathbb{E}\left[\sum_{i=1}^{H} r(s_i, a_i, i) \middle| \pi\right] = p_0^\top V_1^\pi \ .$$

The *optimal value function* is given by $V_h^*(s) = \max_{\pi \in \Pi} V_h^\pi(s)$. Any policy π such that $V_h^\pi(s) = V_h^*(s)$ for all $s \in \mathcal{S}$ and $h \in [H]$ is called optimal. It achieves the optimal expected total reward $\rho^* = \max_{\pi \in \Pi} \rho^\pi$.

The goal of an RL agent is to learn a near-optimal policy after interacting with an MDP for a finite number of episodes T. During each episode $t \in [T]$ the agent follows a policy π_t informed by previous interactions, and after the last episode it outputs a final policy π.

Definition 1. An agent is (α, β)*-probably approximately correct* (PAC) with sample complexity $f(S, A, H, \frac{1}{\alpha}, \log(\frac{1}{\beta}))$, if with probability at least $1 - \beta$ it follows an α-optimal policy π such that $\rho^* - \rho^\pi \leq \alpha$ except for at most $f(S, A, H, \frac{1}{\alpha}, \log(\frac{1}{\beta}))$ episodes.

Definition 2. The (expected cumulative) *regret* of an agent after T episodes is given by

$$\text{Regret}(T) = \sum_{t=1}^{T} (\rho^* - \rho^{\pi_t}) \ ,$$

where $\pi_1, \ldots \pi_T$ are the policies followed by the agent on each episode.

A.2 Privacy in RL

In some RL application domains such as personalized medical treatments, the sequence of states and rewards received by a reinforcement learning agent may contain sensitive information. For example, individual users may interact with an RL agent for the duration of an episode and reveal sensitive information in order to obtain a service from the agent. This information affects the final policy produced by the agent, as well as the actions taken by the agent in any subsequent interaction. Our goal is to prevent damaging inferences about a user's sensitive information in the context of the interactive protocol in algorithm 1 summarizing the interactions between an RL agent $\mathcal{M}$ and T distinct users.

Throughout the execution of this protocol the agent observes a collection of T state-reward trajectories of length H. Each user u_t gets to observe the actions chosen by the agent during the t-th episode, as well as the final policy π. To preserve the privacy of individual users we enforce a (joint) differential privacy criterion: upon changing one of the users in the protocol, the information observed by the other $T-1$ participants will not change substantially. This criterion must hold even if the $T-1$ participants collude adversarially, by e.g., crafting their states and rewards to induce the agent to reveal information about the remaining user.

Formally, we write $U = (u_1, \ldots, u_T)$ to denote a sequence of T users participating in the RL protocol. Technically speaking a user can be identified with a tree of depth H encoding the state and reward responses they would give to all the A^H possible sequences of actions the agent can choose. During

Algorithm 1: Episodic RL Protocol

```
Input: Agent $\mathcal{M}$ and users $u_1, \ldots, u_T$
for $t \in [T]$ do
    for $h \in [H]$ do
        $u_t$ sends state $s_h^{(t)}$ to $\mathcal{M}$
        $\mathcal{M}$ sends action $a_h^{(t)}$ to $u_t$
        $u_t$ sends reward $r_h^{(t)}$ to $\mathcal{M}$
    end
end
$\mathcal{M}$ releases policy $\pi$
```

the protocol the agent only gets to observe the information along a single root-to-leaf path in each user's tree. For any $t \in [T]$, we write $\mathcal{M}_{-t}(U)$ to denote all the outputs excluding the output for episode t during the interaction between $\mathcal{M}$ and U. This captures all the outputs which might leak information about the t-th user in interactions after the t-th episode, as well as all the outputs from earlier episodes where other users could be submitting information to the agent adversarially to condition its interaction with the t-th users.

We also say that two user sequences U and U' are t-neighbors if they only differ in their t-th user.

Definition 3. A randomized RL agent $\mathcal{M}$ is *ε-jointly differentially private under continual observation* (JDP) if for all $t \in [T]$, all t-neighboring user sequences U, U', and all events $E \subseteq \mathcal{A}^{H \times [T-1]} \times \Pi$ we have

$$\Pr[\mathcal{M}_{-t}(U) \in E] \le e^{\varepsilon} \Pr\left[\mathcal{M}_{-t}(U') \in E\right] .$$

This definition extends to the RL setting the one used in [SS18] for designing privacy-preserving algorithms for linear contextual bandits. The key distinctions is that in our definition each user interacts with the agent for H time-steps (in bandit problems one usually has $H = 1$), and we also allow the agent to release the learned policy at the end of the learning process.

Another distinction is that our definition holds for all past and future outputs. In contrast, the definition of JDP in [SS18] only captures future episodes; hence, it only protects against collusion from future users.

To demonstrate that our definition gives a stronger privacy protection, we use a simple example. Consider an online process that takes as input a stream of binary bits $u = (u_1, \ldots, u_T)$, where $u_t \in \{0, 1\}$ is the data of user t, and on each round t the mechanism outputs the partial sum $m_t(u) = \sum_{i=1}^{t} u_i$. Then the following trivial mechanism satisfies JDP (in terms of future episodes as in the JDP definition of [SS18]): First, sample once from the Laplace mechanism $\xi \sim \mathrm{Lap}(\varepsilon)$ before the rounds begin, and on each round output $\widetilde{m}_t(u) = m_t(u) + \xi$. Note that the view of any future user $t' > t$ is $\widetilde{m}_{t'}(u)$. Now let u be a binary stream with user t bit on and let w be identical to u but with user t bit off. Then, by the differential-privacy guarantee of the Laplace mechanism, a user $t' > t$ cannot distinguish between $\widetilde{m}_{t'}(u)$ and $\widetilde{m}_{t'}(w)$. Furthermore, any coalition of future users cannot provide more information about user t. Therefore this simple mechanism satisfies the JDP definition from [SS18].

However, the simple counting mechanism with one round of Laplace noise does not satisfy JDP for past and future outputs as in our JDP (definition 3). To see why, suppose that user $t-1$ and user $t+1$ collude in the following way: For input u, the view of user $t-1$ is $\widetilde{m}_{t-1}(u)$ and the view of user $t+1$ is $\widetilde{m}_{t+1}(u)$. They also know their own data u_{t-1}, u_{t+1}. Then they can recover the data of the t-th user as follows

$$\widetilde{m}_{t+1}(u) - u_{t+1} - \widetilde{m}_{t-1}(u) = m_{t+1}(u) + \xi - u_{t+1} - m_{t-1}(u) - \xi = \sum_{i=1}^{t+1} u_i - u_{t+1} - \sum_{i=1}^{t-1} u_i = u_t$$

Remark. *1. would the algorithm leak more info for the returning user? yes, but we could bound using group privacy. 2. would other users be affected? no, because JDP prevents arbitrary collusion*

A.3 Counting Mechanism

The algorithm we describe in the next section maintains a set of counters to keep track of events that occur when interacting with the MDP. We denote by $\widehat{n}_t(s, a, h)$ the count of visits to state tuple (s, a, h)

Algorithm 2: Private Upper Confidence Bound (PUCB)

```
Parameters Privacy parameter $\varepsilon$, target failure probability $\beta$
:
Input: Maximum number of episodes $T$, horizon $H$, state space $\mathcal{S}$, action space $\mathcal{A}$
$\varepsilon' := \varepsilon/(3H)$
for $s, a, s', h \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times [H]$ do
    Initialize private counters: $\widetilde{r}(s,a,h), \widetilde{n}(s,a,h), \widetilde{m}(s,a,s',h) := \mathrm{PC}(T, \varepsilon', \beta)$
end
for $t <= 1$ to $T$ do
    Private planning: $\widetilde{Q}_t^+ := \mathtt{PrivQ}(\widetilde{r}, \widetilde{n}, \widetilde{m}, \varepsilon)$
    for $h \leftarrow 1$ to $H$ do
        Let $s$ denote the state during step $h$ and episode $t$
        Execute $a := \arg\max_{a'} \widetilde{Q}_t^+(s, a', h)$
        Observe $r \sim \mathcal{R}(s,a,h)$ and $s' \sim \mathcal{P}(.|s,a,h)$
        Feed $r$ to $\widetilde{r}(s,a,h)$
        Feed 1 to $\widetilde{n}(s,a,h)$ and $\widetilde{m}(s,a,s',h)$ and 0 to all other counters $\widetilde{n}(\cdot,\cdot,h)$ and $\widetilde{m}(\cdot,\cdot,\cdot,h)$
    end
end
```

right before episode t, where $a \in \mathcal{A}$ is the action taken on state $s \in \mathcal{S}$ and time-step $h \in [H]$. Likewise $\widehat{m}_t(s,a,s',h)$ is the count of going from state s to s' after taking actions a before episode t. Finally, we have the counter $\widehat{r}_t(s,a,h)$ for the total reward received by taking action a on state s and time h before episode t. Then, on episode t, the counters are sufficient to create an estimate of the MDP dynamics to construct a policy for episode t. The challenge is that the counters depend on the sequence of states and actions, which is considered sensitive data in this work. Therefore the algorithm must release the counts in a privacy-preserving way, and we do this the private counters proposed by [CSS11] and [DNPR10].

A private counter mechanism takes as input a stream $\sigma = (\sigma_1 \dots, \sigma_T) \in [0,1]^T$ and on any round t releases and approximation of the prefix count $c(\sigma)(t) = \sum_{i=1}^{t} \sigma_i$. In this work we will denote PC as the binary mechanism of [CSS11] and [DNPR10] with parameters ε and T. This mechanism produces a monotonically increasing count and satisfies the following accuracy guarantee: Let $\mathcal{M} := \mathrm{PC}(T, \varepsilon)$ be a private counter and $c(\sigma)(t)$ be the true count on episode t, then given a stream σ, with probability at least $1 - \beta$, simultaneously for all $1 \le t \le T$, we have

$$|\mathcal{M}(\sigma)(t) - c(\sigma)(t)| \le \frac{4}{\varepsilon} \ln(1/\beta) \log(T)^{5/2} \quad .$$

While the stated bound above holds for a single ε-DP counter, our algorithm needs to maintain more than S^2AH many counters. A naive allocation of the privacy budget across all these counters will require noise with scale polynomially with S, A, and H. However, we will leverage the fact that the total change across all counters a user can have scales with the length of the episode H, which allows us to add a much smaller amount of noise that scales linearly in H.

Algorithm 3: $\mathtt{PrivQ}(\widetilde{r}, \widetilde{n}, \widetilde{m}, \varepsilon, \beta)$

```
Input: Private counters $\widetilde{r}, \widetilde{n}, \widetilde{m}$, privacy parameter $\varepsilon$, target failure probability $\beta$
$E_\varepsilon = \frac{3}{\varepsilon} H \log\left((2SAH + S^2AH)(\beta')^{-1}\right) \log(T)^{5/2}$
$\widetilde{V}_{H+1}(s) := 0 \quad \forall s \in \mathcal{S}$
for $h \leftarrow H$ to 1 do
    for $s, a \in \mathcal{S} \times \mathcal{A}$ do
        if $\widetilde{n}_t(s,a,h) \geq 2E_\varepsilon$ then
            $\widetilde{\mathrm{conf}}_t(s,a,h) := (H+1)\widetilde{\phi}_t(s,a,h) + \widetilde{\psi}_t(s,a,h)$
        else
            $\widetilde{\mathrm{conf}}_t(s,a,h) := H$
        end
        $\widetilde{Q}_t(s,a,h) := \frac{1}{\widetilde{n}_t(s,a,h)}\left(\widetilde{r}_t(s,a,h) + \sum_{s' \in \mathcal{S}} \widetilde{V}_{h+1}(s')\widetilde{m}_t(s,a,s',h)\right)$
        $\widetilde{Q}_t^+(s,a,h) := \min\left\{H, \widetilde{Q}_t(s,a,h) + \widetilde{\mathrm{conf}}_t(s,a,h)\right\}$
    end
    $\widetilde{V}_h(s) := \max_a \widetilde{Q}_t^+(s,a,h) \quad \forall s \in \mathcal{S}$
end
Output: $\widetilde{Q}_t^+$
```