Towards Deployment-Efficient Reinforcement Learning: Lower Bound and Optimality

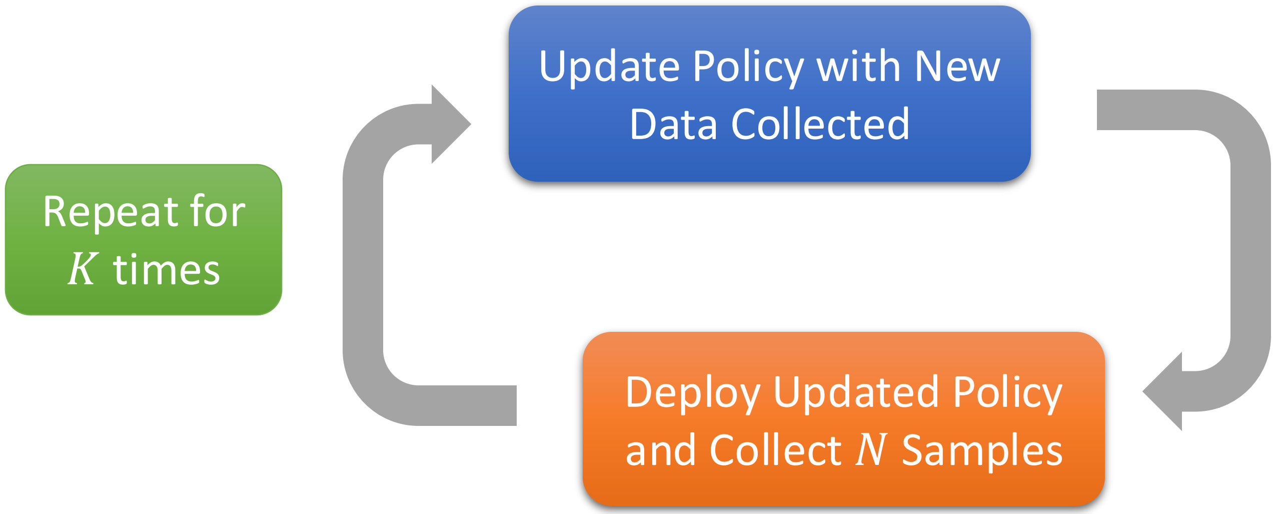
Jiawei Huang[1], Jinglin Chen[1], Li Zhao[2], Tao Qin[2], Nan Jiang[1], Tie-Yan Liu[2]

[1] University of Illinois at Urbana-Champaign

[2] Microsoft Research Asia

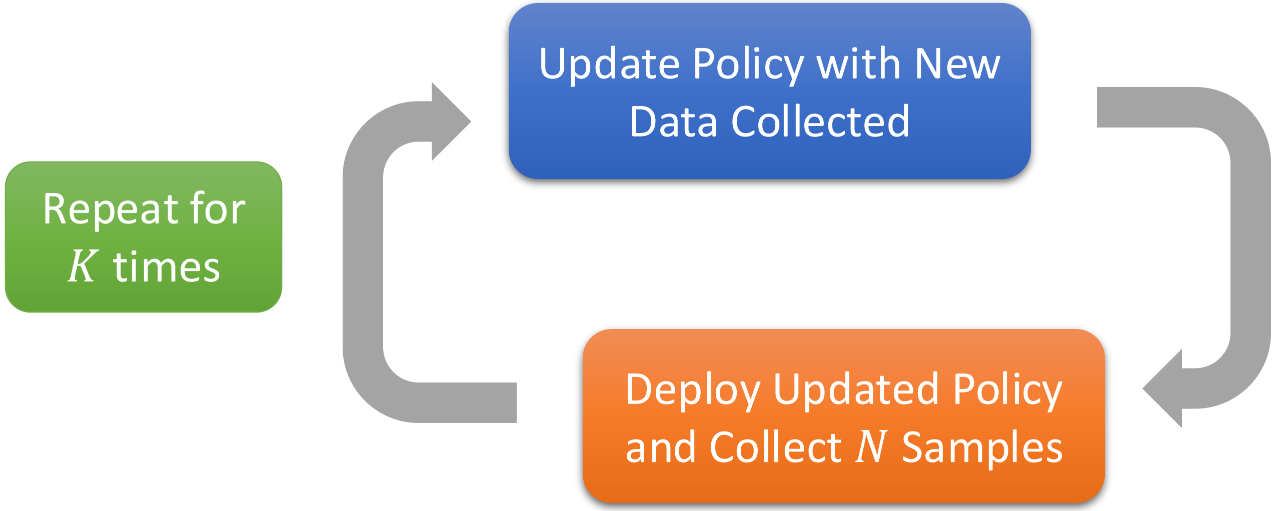
Background

- An Abstraction of Learning Process of Online RL



Background

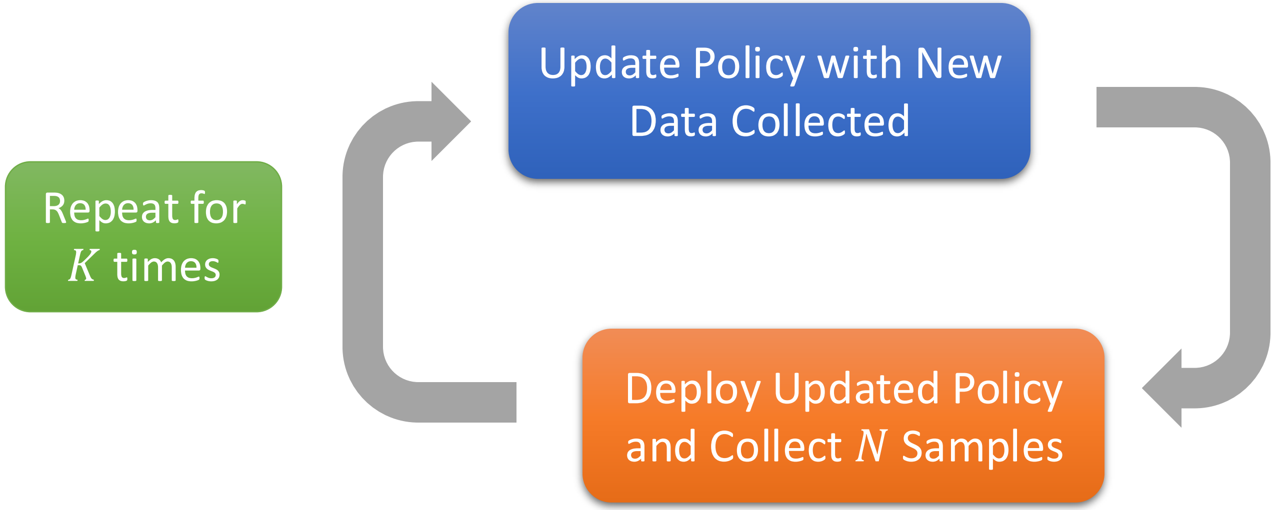
- An Abstraction of Learning Process of Online RL



Deploy a new policy can be costly in many real-world applications.

- Recommendation System
- Education System
- Robotics
- …

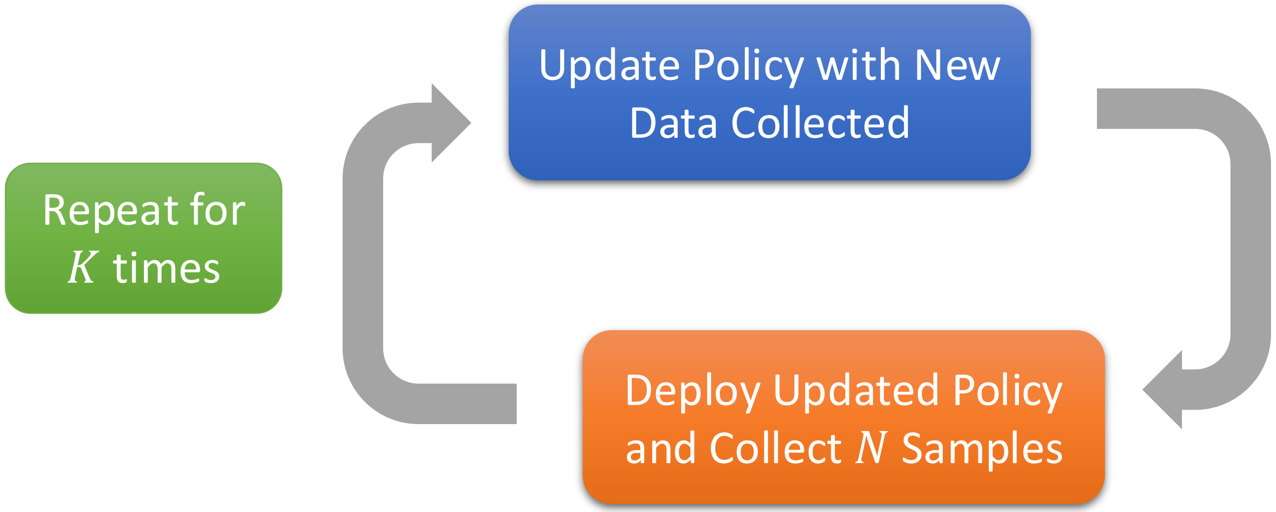
Background



Learn a good policy with K **as small as possible**, while N can be large.

Background

- An Abstraction of Learning Process of Online RL



Deploy a new policy can be costly in many real-world applications.

- Recommendation System
- Education System
- Robotics
- …

We expect to learn a good policy with K **as small as possible**, while N can be large.

A New Theoretical Formulation for DE-RL

- **[Definition] Deployment Complexity**
  - An algorithm has ***deployment complexity K***, if for arbitrary MDP, $\epsilon, \delta > 0$, the algorithm:
    - (1) return ϵ-optimal policy w.p. $1 - \delta$ **after $\boldsymbol{K}$ policy switching**,
    - (2) collect N trajectories in each deployment, with the constraint that N is **polynominal level in standard parameters**.

To avoid trivial case

- **Overall Goal of Deployment-Efficient RL (DE-RL)**
  - *Q1: Lower bound of the deployment complexity?*
  - *Q2: Any algorithms matching lower bound?*

Related Work

- **Empirical Literature**
  - [1] propose the conception of "Deployment-Efficient RL", while a clear theoretical formulation is still an open problem.

- **Theoretical Literature (low-switching cost RL)**
  - Tabular MDP [2] and Linear MDP [3]
  - Relies on "adaptive policy switching strategy", unrealistic when policy deployment is time-consuming.
  - Consider to reduce # of policy switching while achieving near-optimal regret;
  - Only study deploying deterministic policy each time;

Result in sub-optimal deployment complexity

[1] Matsushima et. al., 2020, Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization
[2] Bai et. al., 2020, Provably efficient q-learning with low switching cost
[3] Gao et. al., 2021, A provably efficient algorithm for linear markov decision process with low switching cost

Linear MDP as a Concrete Example

- **Episodic (Time-dependent) Linear MDP**
  - Horizon Length: H
  - State-action feature: $\phi(s_h, a_h) \in R^d$, with $||\phi(s_h, a_h)|| \leq 1$
  - Reward is linear: $r_h(s, a) = \phi^T(s_h, a_h)\theta_h$
  - Transition is linear: $\mathrm{P_h}(s_{h+1}|s_h, a_h) = \phi^T(s_h, a_h)\mu_h(s_{h+1})$

Linear MDP as a Concrete Example

- **Episodic (Time-dependent) Linear MDP**
  - Horizon Length: H
  - State-action feature: $\phi(s_h, a_h) \in R^d$, with $||\phi(s_h, a_h)|| \leq 1$
  - Reward is linear: $r_h(s, a) = \phi^T(s_h, a_h)\theta_h$
  - Transition is linear: $\mathrm{P_h}(s_{h+1}|s_h, a_h) = \phi^T(s_h, a_h)\mu_h(s_{h+1})$

- **[Definition] Deployment Complexity in Linear MDP**
  - An algorithm has ***deployment complexity K***, if for arbitrary MDP, $\epsilon, \delta > 0$, the algorithm:
    - (1) return ϵ-optimal policy w.p. $1 - \delta$ **after $\boldsymbol{K}$ policy switching,**
    - (2) collect N trajectories in each deployment, with the constraint that N is $\mathbf{Poly}(\boldsymbol{d}, \boldsymbol{H}, \frac{\mathbf{1}}{\boldsymbol{\epsilon}}, \boldsymbol{log}\frac{\mathbf{1}}{\boldsymbol{\delta}})$.

Linear MDP as a Concrete Example

- **Episodic (Time-dependent) Linear MDP**
  - Horizon Length: H
  - State-action feature: $\phi(s_h, a_h) \in R^d$, with $||\phi(s_h, a_h)|| \leq 1$
  - Reward is linear: $r_h(s, a) = \phi^T(s_h, a_h)\theta_h$
  - Transition is linear: $\mathrm{P_h}(s_{h+1}|s_h, a_h) = \phi^T(s_h, a_h)\mu_h(s_{h+1})$

- **Separately consider two settings**
  - **Case 1**: Can only deploy deterministic policy;
    - Setting in previous low-switching cost setting
  - **Case 2**: Allow to deploy arbitrary policy (deterministic/stochastic/non-Markovian);
    - Largely omitted in previous literatures, but strictly lower deployment complexity.

Lower Bound of Deployment Complexity

- **Case 1**: Only allow to deploy deterministic policy;
  - **Lower bound**: $\Omega(dH)$
  - **Intuition**: a (d, H)-linear MDP has dH different "dimensions to explore", while each deterministic policy can only explore one of them.

- **Case 2**: Allow to deploy arbitrary policy;
  - **Lower bound**: $\Omega(H/\log N)$
  - **Intuition**: the agent can only push exploration forward for $O(\log N)$ time steps (layers).

Lower Bound of Deployment Complexity

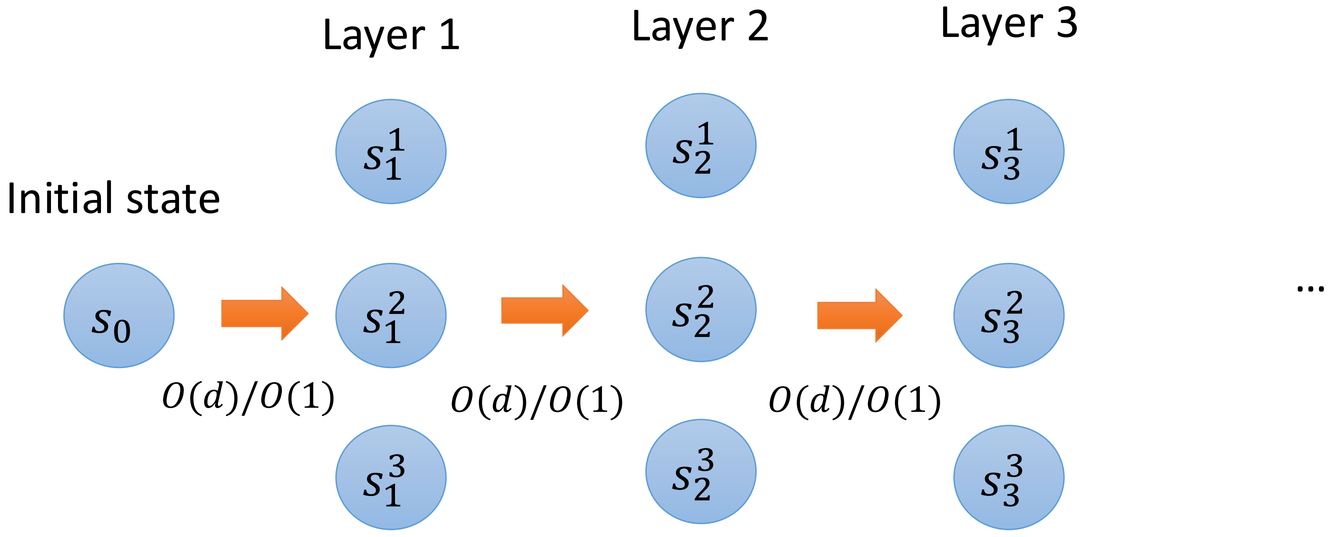
- **Case 1**: Only allow to deploy deterministic policy;
  - **Lower bound**: $\Omega(dH)$
  - **Intuition**: a (d, H)-linear MDP has dH different "dimensions to explore", while each deterministic policy can only explore one of them.

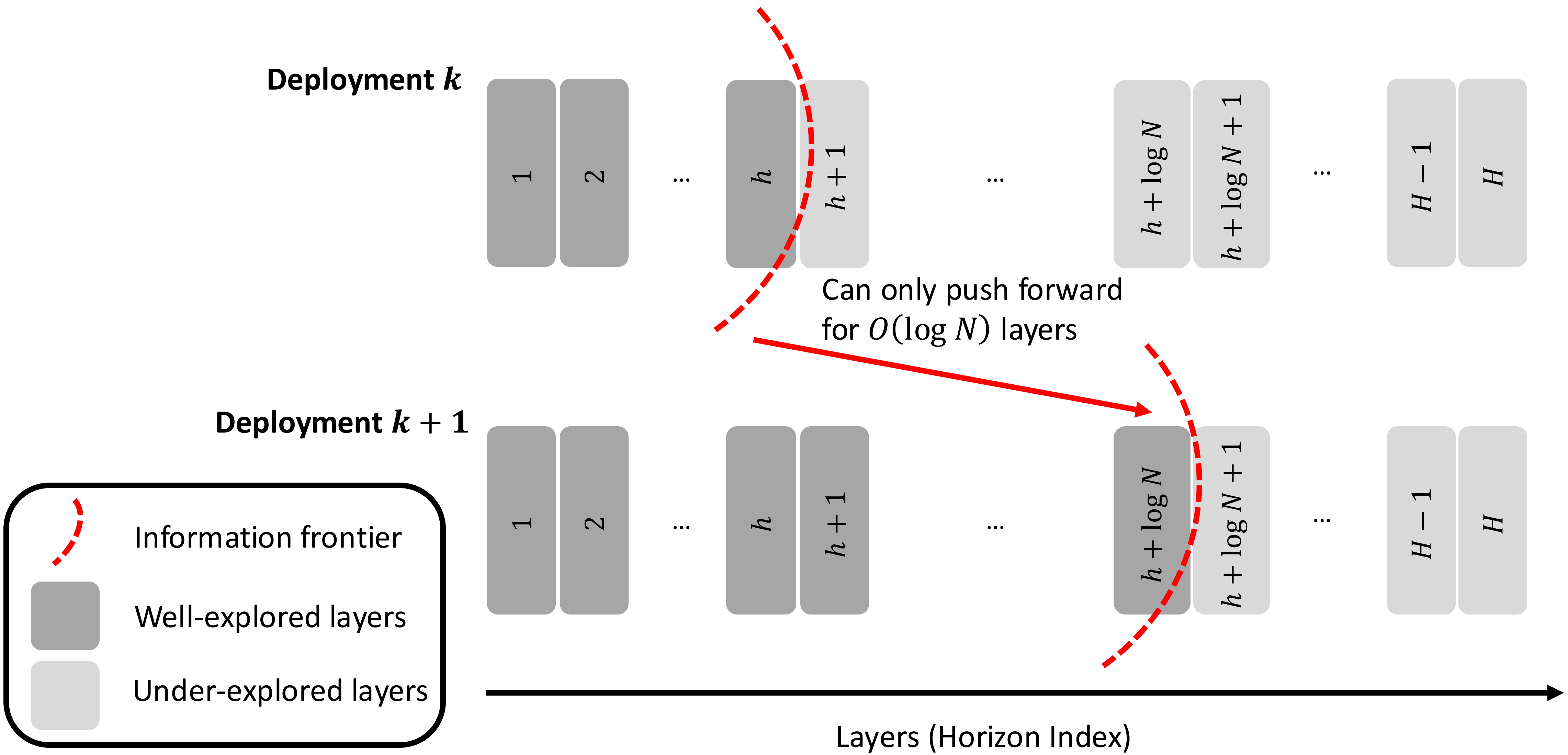
- **Case 2**: Allow to deploy arbitrary policy;
  - **Lower bound**: $\Omega(H/\log N)$
  - **Intuition**: the agent can only push exploration forward for $O(\log N)$ time steps (layer).

- **Corollary for time-homogeneous linear MDP:**
  - $\Omega(d)$ for Case 1 (deterministic policy setting)
  - $\Omega(\min\{d, H\}/\log N)$ for Case 2 (arbitrary policy setting)

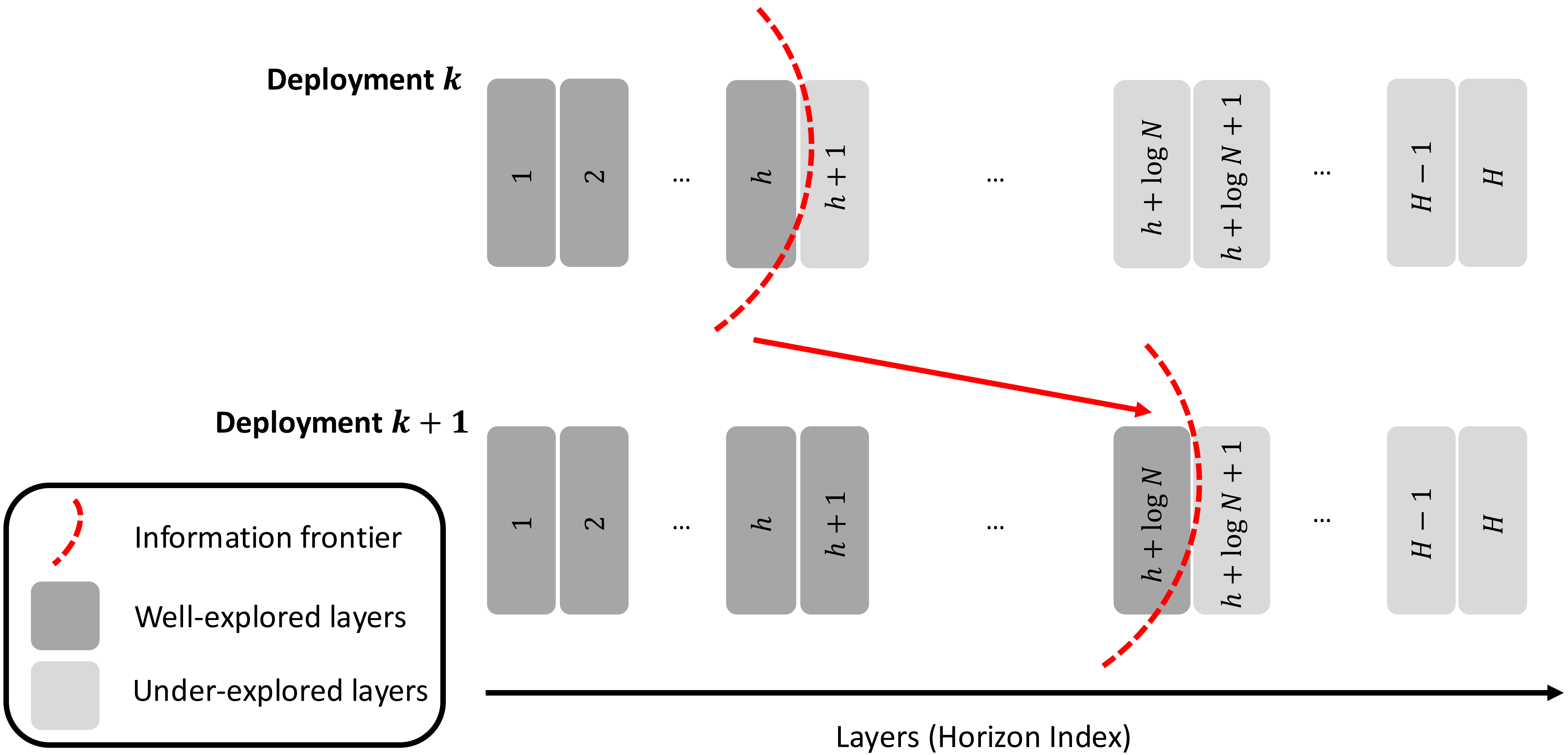
Algorithm with Near-Optimal Deployment Complexity

- High-level idea
  - Explore each layer (time step) per $O(d)$ or $O(1)$ deployments





Deployment k
1
2
...
h
$h+1$
...
$h+\log N$
$h+\log N+1$
...
$H-1$
H
Can only push forward
for $O(\log N)$ layers
Deployment $k+1$
1
2
...
h
$h+1$
...
$h+\log N$
$h+\log N+1$
...
$H-1$
H
Information frontier
Well-explored layers
Under-explored layers
Layers (Horizon Index)



Deployment k
1
2
...
h
$h+1$
...
$h+\log N$
$h+\log N+1$
...
$H-1$
H
Deployment $k+1$
1
2
...
h
$h+1$
...
$h+\log N$
$h+\log N+1$
...
$H-1$
H
Information frontier
Well-explored layers
Under-explored layers
Layers (Horizon Index)

Algorithm with Near-Optimal Deployment Complexity

- **Case 1**: Only allow to deploy deterministic policy;
  - **Algorithm**: LSVI-UCB [Jin et. al., 2020] + Layer-by-layer
  - A novel elliptical potential lemma (Lem. 4.2)
  - **Guarantees**:
    - Deployment complexity K: $O(dH)$;
    - (Asmptotically) # of Trajs N: $O(\frac{H^4 d^3}{\epsilon^2} \log^2 \frac{Hd}{\delta\epsilon})$
  - **Remarks**:
    - Layer-by-layer strategy is not necessary for deployment efficiency in case 1, but some additional benefits (see Appx. C.4).
    - Similar analysis can be extended to reward-free setting

Algorithm with Near-Optimal Deployment Complexity

- **Case 2**: Allow to deploy arbitrary policy;
  - **Algorithm**: a new batch exploration algorithm
    - Explore in a layer-by-layer manner
    - For each layer
      - Use **a novel covariance matrix estimation** method to evaluate the exploration ability of given policies;
      - Plus a **bonus-term driven** method to find a set of deterministic policies **cover all the dimensions.**
  - Require an additional reachability assumption $\nu_{\min}$.
  - **Guarantees**:
    - Deployment complexity K: $\Theta(H)$;
    - (Asmptotically) # of Trajs N: $\mathrm{Poly}(d, H, \frac{1}{\epsilon}, \log\frac{1}{\delta}, \frac{1}{\nu_{\min}})$
  - **Remarks**:
    - Naturally a reward-free exploration
    - Open problem: is it possible to remove dependence on $\nu_{\min}$

Thanks!