AsyncQVI: Asynchronous-Parallel Q-Value Iteration for Reinforcement Learning with Near-Optimal Sample Complexity

Yibo Zeng, Fei Feng and Wotao Yin
ybzeng15@fudan.edu.cn; fei.feng@math.ucla.edu; wotaoyin@math.ucla.edu

Abstract

Given a discounted Markov decision process $(\mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \gamma)$, we aim to find an $\varepsilon-$optimal policy efficiently. Our algorithm assumes:

- a generative model GM. GM takes any state-action pair (s, a) as input and outputs a sample of next state s' and reward $r^a_{ss'}$ following P.
- N parallel agents running asynchronously with shared memory.

It achieves:

- near-optimal sample complexity.
- $\mathcal{O}(\mathcal{S})$ memory complexity.
- linear parallel speedup.

Related Work

Related Async-Parallel Dynamic Programming or RL Algorithms for DMDPs

Algorithms	Methods	Delay	Rate	Sample	Memory	References
Totally Async QVI	DP	**Unbdd**	–	N/A	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$	[1]
Partially Async QVI	DP	**Bdd**	–	N/A	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$	[1]
Async Q-learning	RL	**Unbdd**	–	–	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$	[2]
AsyncQVI	RL	**Bdd**	√	√	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	This Work

Related RL Algorithms with a Generative Model

Algorithms	Async	Sample Complexity	Memory	References
Variance-Reduced VI	×	$\tilde{O}(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^4\varepsilon^2}\log(\frac{1}{\delta}))$	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$	[3]
Variance-Reduced QVI	×	$\tilde{O}(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^3\varepsilon^2}\log(\frac{1}{\delta}))$ (log-factored optimal)	$\mathcal{O}(\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert)$	[4]
AsyncQVI	√	$\tilde{O}(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^5\varepsilon^2}\log(\frac{1}{\delta}))$	$\mathcal{O}(\lvert\mathcal{S}\rvert)$	This Work

References

[1] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[2] John N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, Sep 1994.

[3] Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. Society for Industrial and Applied Mathematics, 2018.

[4] Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. In *Advances in Neural Information Processing Systems*, pages 5192–5202, 2018.

Algorithm

Input: $\varepsilon \in (0, (1-\gamma)^{-1})$, $\delta \in (0,1)$, L, K;

Shared variables: $\mathbf{v} \leftarrow \mathbf{0}$, $\pi \leftarrow \mathbf{0}$, $t \leftarrow 0$;

Private variables: $\hat{\mathbf{v}}, r, S, q$;

While $t < L$**, every agent asynchronously:**

1. select a state s_t and an action a_t;
2. copy shared variable to local memory $\hat{\mathbf{v}} \leftarrow \mathbf{v}$;
3. call $\mathrm{GM}(s_t, a_t)$ K times and collect samples $\{s'_1, \ldots, s'_K\}$ and $\{r_1, \ldots, r_K\}$.
4. $q \leftarrow \frac{1}{K}\sum_{k=1}^K r_k + \gamma \frac{1}{K}\sum_{k=1}^K \hat{v}_{s'_k} - \frac{(1-\gamma)\varepsilon}{4}$;
5. if $q > v_{s_t}$
   mutex lock
   $v_{s_t} \leftarrow q$, $\pi_{s_t} \leftarrow a_t$
   mutex unlock
6. $t \leftarrow t + 1$

Insight

AsyncQVI is an approximation of the Q-value iteration with both asynchronous delay and stochastic estimation.

1. Q-value iteration with full update:

$$Q_{s,a}(t+1) = \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} Q_{s',a'}, \ \forall\, s, a$$

2. Q-value iteration with coordinate update and asynchronous delay:

$$Q_{s,a}(t+1) = \begin{cases} \sum_{s'} p^a_{ss'} r^a_{ss'} + \gamma \sum_{s'} p^a_{ss'} \max_{a'} \hat{Q}_{s',a'}(t), & \text{if updating } (s,a) \text{ at } t; \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

3. AsyncQVI: Asynchronous Q-value iteration with stochastic estimation:

$$Q_{s,a}(t+1) = \begin{cases} \frac{1}{K}\sum_k r_k + \gamma \frac{1}{K}\sum_k \max_{a'} \hat{Q}_{s'_k,a'}(t) - (1-\gamma)\varepsilon/4 & \text{if updating } (s,a) \text{ at } t; \\ Q_{s,a}(t), & \text{o.w.} \end{cases}$$

Convergence of AsyncQVI is established through building a sequence of type 2 with the same asynchronous delay. Estimation error is controlled through enough sampling and the discounted factor.

Theory

Theorem 1 *Under partial asynchronism, given accuracy parameters ε and δ, with $L = \lceil 2B_1 + \frac{B_1+B_2-1}{1-\gamma}\log\left(\frac{2}{(1-\gamma)\varepsilon}\right)\rceil$ and $K = \lceil \frac{8}{(1-\gamma)^4\varepsilon^2}\log\left(\frac{4L}{\delta}\right)\rceil$, AsyncQVI returns an ε-optimal policy π with probability at least $1-\delta$. Here B_1 is the uniform consecutive update bound and B_2 is the uniform communication delay bound.*

Corollary 1 *Under partial asynchronism, AsyncQVI returns an ε-optimal policy π with probability at least $1-\delta$ at the sample complexity*

$$\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|}{(1-\gamma)^5\varepsilon^2}\log\left(\frac{1}{\delta}\right)\right),$$

provided that $B_1 + B_2 = \mathcal{O}(|\mathcal{S}||\mathcal{A}|)$.

Test Problem

We test the sailing problem with two positioning noises: a wind noise $\mathcal{N}(0, \sigma_1^2)$ and a vortex noise $\mathcal{N}(0, \sigma_2^2)$. The latter occurs with probability p. Given the current position (x, y) and an action (δ_x, δ_y), the next position is

$$\left(x + \delta_x + \mathcal{N}(0,\sigma_1^2),\ y + \delta_y + \mathcal{N}(0,\sigma_1^2)\right) \sim 1-p, \text{ or}$$
$$\left(x + \delta_x + \mathcal{N}(0,\sigma_1^2+\sigma_2^2),\ y + \delta_y + \mathcal{N}(0,\sigma_1^2+\sigma_2^2)\right) \sim p.$$

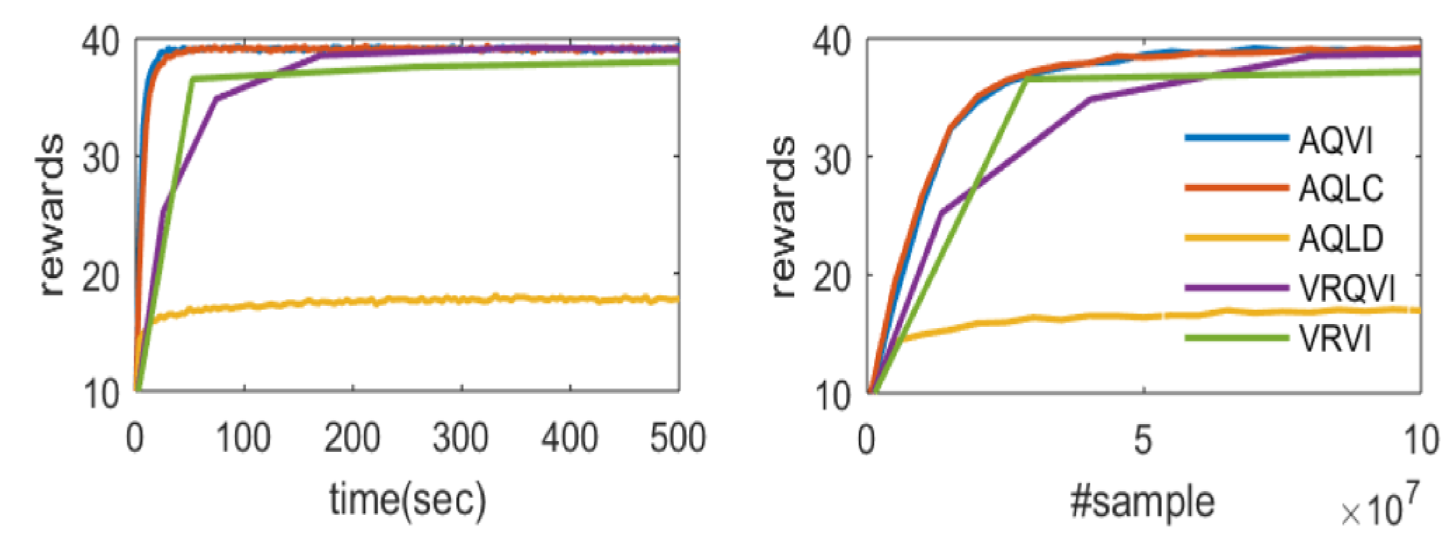
We set the instant reward as

$$d \times \left|\frac{\text{angle between wind and action directions}}{45}\right|,$$

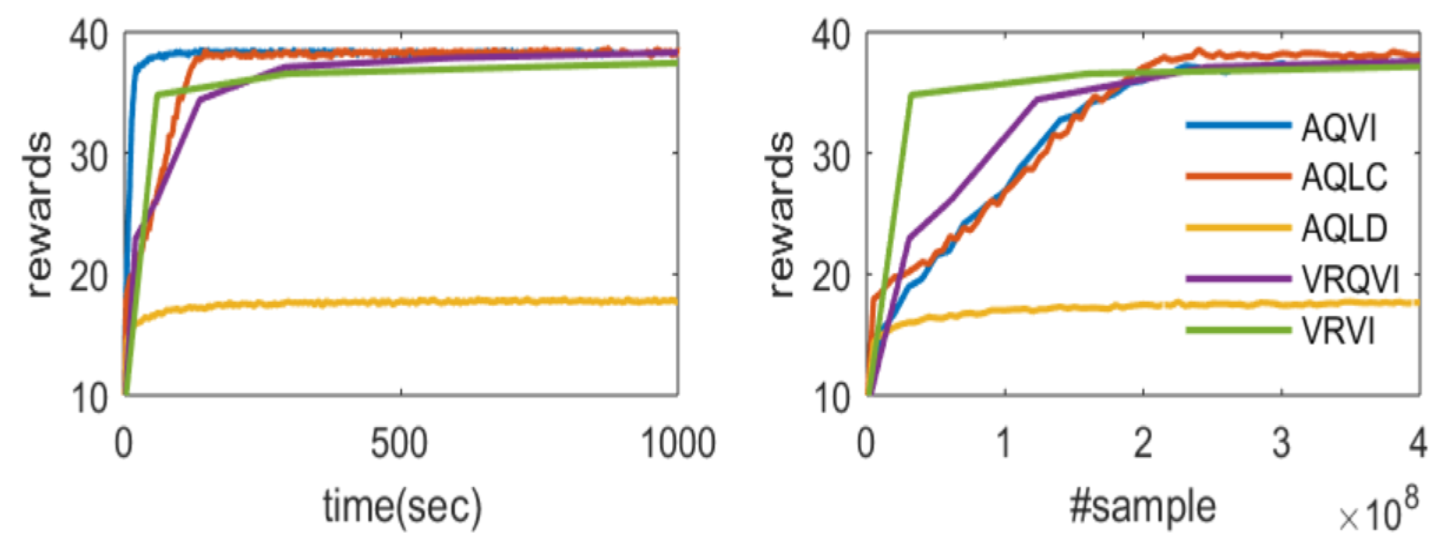
where d is a constant hyperparameter.

Results

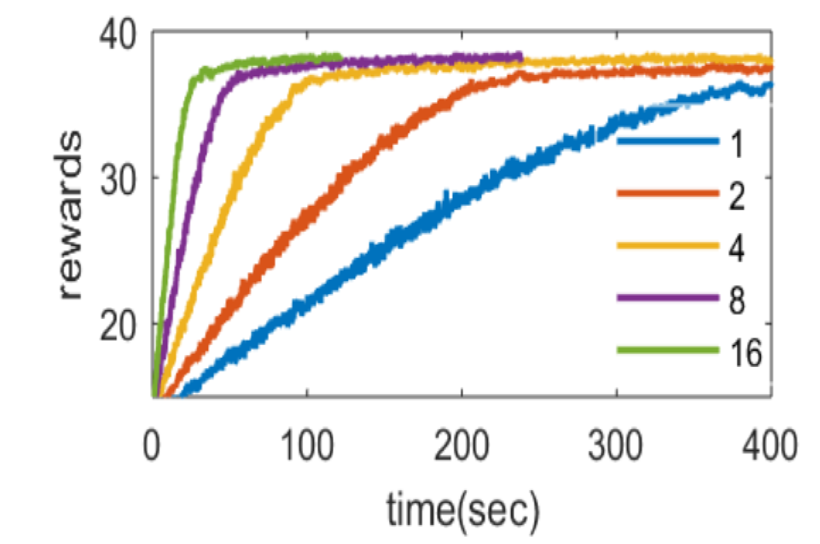
We compared five algorithms: AsyncQVI (AQVI), Async Q-learning with Constant stepsize (AQLC), Async Q-learning with diminishing stepsize (AQLD), Variance-Reduced QVI (VRQVI), and Varian-reduced VI (VRVI). For parallel algorithms (the first three), we use 20 threads. We also test parallel performance. Overall, our algorithm is similar to Q-learning but with less memory and averagely 10× faster than variance-reduced methods with 3× more samples. Linear parallel speedup is achieved.



(a) $\sigma_1 = 0.1, p = 0, d = 0.05$



(b) $\sigma = 0.1, p = 0.05, \sigma_2 = 1, d = 0.05$



(c) Test with doubling threads.