Algorithm: Maximum Node

1. **Input:** $m_0, \varepsilon$
2. **Initialization:** [Only executed on first call]
   3. SampledNodes ← $\emptyset$
4. $r ← 0$
5. **Run:**
   6. if $\frac{1}{\gamma} > \frac{1}{1-\delta}$ then
5. **Output:** $0$
8. **end if**
9. if $\text{SampledNodes} > m$ then
10. ActiveNodes ← SampledNodes(1 : $m$)
11. else
12. while $|\text{SampledNodes}| < m$ do
13. $r ← (\text{new sample of next state})$
14. SampledNodes.append($r$)
15. $r ← r + s$ [new sample of reward]
16. end while
17. ActiveNodes ← SampledNodes
18. end if [At this point, $\text{ActiveNodes} = m$]
19. for all unique nodes $s$ ∈ ActiveNodes do
20. $k ← \text{Recurrence of } s$ in ActiveNodes
21. $\mu_s ← \text{call } s$ with parameters $(k, \varepsilon, r)$
22. $\mu_s ← \mu + \kappa/k/m$
23. end for
24. **Output:** $\mu + r/|\text{SampledNodes}|$

Algorithm: Average Node

1. **Input:** $m, \varepsilon$
2. **Initialization:** [Only executed on first call]
3. **SampledNodes** ← $\emptyset$
4. $r ← 0$
5. **Run:**
6. if $\frac{1}{\gamma} > \frac{1}{1-\delta}$ then
7. **Output:** $0$
8. **end if**
9. if $\text{SampledNodes} > m$ then
10. ActiveNodes ← SampledNodes(1 : $m$)
11. else
12. while $|\text{SampledNodes}| < m$ do
13. $r ← (\text{new sample of next state})$
14. SampledNodes.append($r$)
15. $r ← r + s$ [new sample of reward]
16. end while
17. ActiveNodes ← SampledNodes
18. end if [At this point, $\text{ActiveNodes} = m$]
19. for all unique nodes $s$ ∈ ActiveNodes do
20. $k ← \text{Recurrence of } s$ in ActiveNodes
21. $\mu_s ← \text{call } s$ with parameters $(k, \varepsilon, r)$
22. $\mu_s ← \mu + \kappa/k/m$
23. end for
24. **Output:** $\mu + r/|\text{SampledNodes}|$

Example for $D=0$

- The gap of a node is the difference in value between the best and second best action.
- Low gap refer to hard problems.
- $\Delta(s) = s \rightarrow \gamma s$ with probability $p(s'/s)$.
- The following assumption measure the number of low gap nodes.

Assumption 1. $\exists s, h > 0$ s.t. for all average node $s$ and $t > 0$

$$P[\Delta(s) < t] < \alpha$$

Theorem 3. Under Assumption 1, $d = 0$

When $d = 0$, the Sample complexity is of order $(1/\varepsilon)^2$ which is the same order as Monte Carlo sampling.

Key Ideas

- Tree-based algorithm
- Delicate treatment of uncertainty
- Refining few paths

Analysis

- $\Delta_{new}(s')$: The difference of the sum of discounted rewards staying at $s'$ between an agent playing optimally and one playing first the action toward $s$ and then optimally.

Definition 1 (Near-optimality). We say that a node $s$ depth $h$ is near-optimal, if for all $h' < h$

$$\Delta_{new}(s') \leq \frac{2h-h'}{h'}$$

or the action from $s'$ to $s$ is optimal

with $s'$ the ancestor of $s$ of depth $h'$. Let $N_h$ be the set of all near-optimal nodes of depth $h$.

Definition 2. We define $\kappa \in [1, K]$ as the smallest number such that

$$\exists \ C_0, \forall h \in [N_h], |N_h| \leq C_0(|N_h|)$$

- There are at most $(AS)^d$ nodes of depth $h$ thus $\kappa \leq A$.
- With probability $1 - \delta$, Trailblazer only explore near-optimal nodes.

Definition 3. We define $d \geq 0$ as the smallest $d$ such that there exists $\alpha > 0$ for which for all $h > 0$

$$\sup_{h > 0} \frac{1}{\alpha} \prod_{h'}^{h} \left( \frac{1}{\alpha} \right) \leq \frac{2h-h'}{h'}$$

\(\sup_{h > 0} \frac{1}{\alpha} \prod_{h'}^{h} \left( \frac{1}{\alpha} \right) \leq \frac{2h-h'}{h'}\)

With $S^d$: Random node of depth $h$ chosen according to transition probabilities. $S_{h'}$: $s/h$-node parent of $S^d$ of depth $h'$.

$OPT_{h'}$: 1 if the action at $S_{h'}$ to $S^d$ is optimal else 0.

- It also takes into account the difficulty to identify the near-optimal paths.
- If $d$ is higher when low gap nodes are concentrated.

References

Michael Kearns, Yevgeniy Maruyama, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. ICAL, 1999.
