Capítulo 4

Monte-Carlo Tree Search

A complexidade dos algoritmos VI e PI são proporcionais a cardinalidade do conjunto de estados; para conjunto de estados muito grande, esses algoritmos não podem ser utilizados na prática. Os algoritmos estudados no capítulo anterior permitem tratar problemas com um espaço de estado maiores, desde que a quantidade de estados alcançáveis a partir de um estado inicial de interesse seja razoável. Ainda, se uma boa heurística está disponível *a priori*, na melhor das hipóteses, a complexidade dos algoritmos LRTDP e LAO* dependem da cardinalidade de estados alcançáveis seguindo a política ótima.

Nesse capítulo será considerado alternativas para problemas cuja a cardinalidade do espaços de estados alcançáveis por uma política ótima seja impraticável. Note que a cardinalidade de estados alcançáveis depende de dois fatores: (i) a cardinalidade dos nós sucessores, e (ii) o horizonte para atingir a meta. Dependendo da cardinalidade da ramificação de cada estado, o próprio operador de Bellman pode ser impraticável.

Em tais MDPs, quando uma política não pode ser calculada de forma completa, pode ser mais interessante pensar planejamento

com uma visão on-line. Nesta visão, considera-se a complexidade do planejamento por estado, sendo que a saída do planejador é a ação apenas para aquele estado; então, o planejador é utilizado novamente para cada próximo estado. Nesta visão on-line o próprio algoritmo é a política, mas uma política cujo o processamento por estado é não trivial.

4.1 Prelúdio Probabilístico

4.1.1 Inequalidades

Teorema 12 (Markov's Inequality)**.** *Suponha que X é uma variável aleatória tal que* $\Pr(X \geq 0) = 1$*, então para qualquer* $t > 0$*,*

$$\Pr(X \geq t) \leq \frac{E[X]}{t}.$$

Demonstração. Por conveniência, assuma que X é uma variável aleatória discreta, então:

$$\mathrm{E}[X] = \sum_x x f(x) \geq \sum_{x \geq t} x f(x) \geq \sum_{x \geq t} t f(x) = t \Pr(X \geq t).$$

Logo $\Pr(X \geq t) \leq \dfrac{\mathrm{E}[X]}{t}$. □

Teorema 13 (Chebyshev's Inequality)**.** *Suponha que X é uma variável aleatória tal que exista $Var(X)$, então para qualquer* $t > 0$*,*

$$\Pr(|X - E[X]| \geq t) \leq \frac{Var[X]}{t^2}.$$

Demonstração. Seja $Y = (X - \mathrm{E}[X])^2$. Então, $\Pr(Y \geq 0) = 1$ e $\mathrm{E}[Y] = \mathrm{Var}[X]$. Aplicando a inequalidade de Markov, temos:

$$\Pr(|X - \mathrm{E}[X]| \geq t) = \Pr(Y \geq t^2) \leq \frac{\mathrm{Var}[X]}{t^2}.$$

□

Teorema 14 (Chernoff Bounds)**.** *Suponha que X é uma variável aleatória então para todo $s > 0$:*

$$\Pr(X \geq t) \leq \frac{E[e^{sX}]}{e^{st}}$$

e

$$\Pr(X \leq t) \leq \frac{E[e^{-sX}]}{e^{-st}}$$

Demonstração. Considere o primeiro caso:

$$\Pr(X \geq t) = \Pr(e^{sX} \geq e^{st}).$$

Note que $e^{sX} > 0$, então, pode-se aplicar a inequalidade de Markov para obter $\Pr(X \geq t) \leq \frac{\mathrm{E}[e^{sX}]}{e^{st}}$.

Agora, considere o segundo caso:

$$\Pr(X \leq t) = \Pr(e^{-sX} \geq e^{-st}).$$

Novamente, aplicando inequalidade de Markov, obtém-se $\Pr(X \leq t) \leq \frac{\mathrm{E}[e^{-sX}]}{e^{-st}}$.

□

Lema 1 (Hoeffding's Lemma)**.** *Suponha que $X \in [a, b]$ é uma variável aleatória então:*

$$E[e^{sX}] \leq e^{s\mu} e^{\frac{s^2(a-b)^2}{8}},$$

onde $\mu = E[X]$.

Demonstração. Considere a variável $Y = X - \mu$, tem-se que $X \in [a', b'] = [a - \mu, b - \mu]$, $\mathrm{E}[Y] = 0$ e:

$$\mathrm{E}[e^{sY}] = \mathrm{E}[e^{s(X-\mu)}] = e^{-s\mu}\mathrm{E}[e^{sX}].$$

Uma vez que e^{sY} é convexa, é verdade:

$$e^{sY} \leq \frac{Y - a'}{b' - a'} e^{sb'} + \frac{b' - Y}{b' - a'} e^{sa'}.$$

Considerando a esperança de ambos os lados e definindo $\theta = -\frac{a'}{b' - a'} = -\frac{a'}{b - a}$:

$$\begin{aligned} \mathrm{E}[e^{sY}] &\leq \frac{-a'}{b - a} e^{sb'} + \frac{b'}{b - a} e^{sa'} = (1 - \theta) e^{sa'} + \theta e^{sb'} \\ &= e^{sa'} \left(1 - \theta + \theta e^{s(b-a)}\right) = \left(1 - \theta + \theta e^{s(b-a)}\right) e^{-s\theta(b-a)} \end{aligned}$$

Defina $u = s(b - a)$ e $g(u) = -\theta u + \log(1 - \theta + \theta e^u)$ tal que:

$$\left(1 - \theta + \theta e^{s(b-a)}\right) e^{-s\theta(b-a)} = e^{g(u)}.$$

Finalmente, por conta do Teorema de Taylor, para qualquer u existe $\eta \in [0, u]$ tal que para todo u:

$$g(u) = g(0) + u g'(0) + \frac{u^2}{2} g''(\eta).$$

Note que $g(0) = 0$ e $g'(0) = 0$ e tem-se:

$$g''(u) = \frac{\theta e^u}{1 - \theta + \theta e^u} \left(1 - \frac{\theta e^u}{1 - \theta + \theta e^u}\right) \leq \frac{1}{4}.$$

Logo:

$$\mathrm{E}[e^{sX}] = e^{s\mu} \mathrm{E}[e^{sY}] \leq e^{s\mu} e^{\frac{u^2}{2} g''(\eta)} \leq e^{s\mu} e^{\frac{u^2}{8}} = e^{s\mu} e^{\frac{s^2 (a-b)^2}{8}}.$$

□

Teorema 15 (Hoeffding's Inequality)**.** *Seja $X_1, \ldots, X_N$ variáveis aleatórias independentes e identicamente distribuídas tal que $E[X_i] = \mu$ e $X_i \in [a, b]$. Então, para qualquer $t > 0$,*

$$\Pr(|\overline{X}_N - \mu| \geq t) \leq 2e^{-\frac{2Nt^2}{(b-a)^2}},$$

onde $\overline{X}_N = \dfrac{1}{N}\sum_{i=1}^{N} X_i$.

Demonstração. Considere as variáveis aletórias $\overline{Y_i} = X_i - \mu$, tem-se que $Y_i \in [a', b'] = [a - \mu, b - \mu]$, $\mathrm{E}[Y_i] = 0$ e:

$$\Pr(|\overline{X}_N - \mu| \geq t) = \Pr(|\overline{Y}_N| \geq t).$$

Tem-se que:

$$\begin{aligned} \Pr(|\overline{Y}_N| \geq t) &= \Pr(\overline{Y}_N \geq t) + \Pr(\overline{Y}_N \leq -t) \\ &= \Pr(\overline{Y}_N \geq t) + \Pr(-\overline{Y}_N \geq t) \end{aligned}.$$

Usando o método de Chernoff, obtém-se:

$$\begin{aligned} \Pr(\overline{Y}_N \geq t) &= \Pr\left(\sum_{i=1}^{N} Y_i \geq Nt\right) = \Pr\left(e^{s\sum_{i=1}^{N} Y_i} \geq e^{sNt}\right) \\ &\leq \frac{\mathrm{E}\left[e^{s\sum_{i=1}^{N} Y_i}\right]}{e^{sNt}} = \frac{\prod \mathrm{E}\left[e^{sY_i}\right]}{e^{sNt}} = \frac{\mathrm{E}\left[e^{sY_i}\right]^N}{e^{sNt}} \end{aligned}$$

Do lema de Hoeffding tem-se que $\mathrm{E}\left[e^{sY_i}\right] \leq e^{\frac{s^2(b-a)^2}{8}}$. Então:

$$\Pr(\overline{Y}_N \geq t) \leq \frac{e^{N\frac{s^2(b-a)^2}{8}}}{e^{sNt}},$$

e essa probabilidade é minimizada para $s = \dfrac{8t}{2(a-b)^2}$ resultando em:

$$\Pr(\overline{Y}_N \geq t) \leq e^{-\frac{2Nt^2}{(b-a)^2}}.$$

Repetindo o mesmo procedimento para o segundo termo $\Pr(-\overline{Y}_N \geq t)$ obtém-se o resultado. □

Corolário 1. *Seja $X_1, \ldots, X_N$ variáveis aleatórias independentes e identicamente distribuídas tal que $E[X_i] = \mu$ e $X_i \in [a, b]$. Então probabilidade pelo menos $1 - \delta$ tem-se:*

$$\Pr\left(|\overline{X}_N - \mu| \leq \sqrt{\frac{(b-a)^2}{2N} \log \frac{2}{\delta}}\right) \geq 1 - \delta.$$

4.2 Método de Monte Carlo

Métodos de Monte Carlo consideram amostragem de alguma variável aleatória X para estimar propriedades dessa variável aleatória. Os limites estudados na seção anterior são importantes para avaliar a qualidade das estimativas realizadas.

Quando se considera métodos de Monte Carlo, não se procura mais uma solução ϵ-ótima, como as obtidas pelos algoritmos estudados até o momento. Nesse caso, se procura uma solução correta provavelmente aproximada (Probably Approximately Correct - PAC).

Considere uma política π e a variável aleatória $R_{\pi,H,s_0} = \sum_{t=0}^{H-1} \gamma^t r_t$ que é o custo acumulado descontado obtido ao executar a política π por H passos a partir do estado inicial s_0. Lembre-se que o valor de uma política π para o estados s_0 é dado por:

$$V^\pi(s_0) = \lim_{H\to\infty} \mathrm{E}[R_{\pi,H,s_0}] = \lim_{H\to\infty} \mathrm{E}\left[\sum_{t=0}^{H-1} \gamma^t r_t | \pi, s_0\right].$$

Para obter uma aproximação para $V^\pi(s_0)$ pelo método de Monte Carlo pode-se utilizar o seguinte algoritmo:

1. para n entre 1 e N faça
   a) $R_n \leftarrow 0$
   b) $s \leftarrow s_0$
   c) para t entre 0 e $H-1$ faça
      i. $a_t \leftarrow \pi(s_t)$
      ii. $r_t \leftarrow R(s_t, a_t)$
      iii. $s_{t+1} \sim T(s_t, a_t, \cdot)$
      iv. $R_n \leftarrow R_n + \gamma^t r_t$
2. retorna $\frac{1}{N}\sum_{n=1}^{N} R_n$

Os limites estudados anteriormente permitem estudar a precisão do método de Monte Carlo e responder as perguntas:

- Qual é o erro máximo gerado com probabilidade $(1-\delta)$?
- Qual é a probabilidade que o erro não seja menor que ϵ?
- qual horizonte H devo escolher e quantas simulações N devo fazer para garantir um erro máximo ϵ com probabilidade $(1-\delta)$?

Dentre as inequalidades descritas na seção anterior, a inequalidade de Hoeffding nos vai ser bastante útil. A única exigência que ela faz sobre a variável aleatória é que tenha limite superior e inferior.

A primeira suposição é com relação a função de recompensa $R(\cdot)$ que deve ser limitada. Segundo, deve-se considerar o tipo de avaliação utilizada. Nos casos de avaliações que foram considerados no curso, o uso de horizonte finito com recompensa limitada garante que a variável aleatória de interesse seja limita, já o uso de horizonte

indeterminado não tem a mesma garantia. Por outro lado, o uso de fator de desconto garante que a recompensa acumulada descontada seja sempre finita e muitos trabalhos fazem análise nesse contexto.

Considere $R_{max} = \max_{s \in \mathcal{S}, a \in \mathcal{A}} R(s,a)$ e $R_{min} = \min_{s \in \mathcal{S}, a \in \mathcal{A}} R(s,a)$. Sob a avaliação de custo acumulado descontado, é fácil mostrar que os limites inferiores e superiores são:

$$V_{min} = R_{min} \frac{1}{1-\gamma} \qquad \text{e} \qquad V_{max} = R_{max} \frac{1}{1-\gamma}.$$

Partindo da mesma análise é fácil encontrar limites para o erro causado no valor de um estado ao considerar um custo acumulado descontado com horizonte finito H:

$$\mathrm{E}[R_{\pi,H,s}] + \gamma^H V_{min} \leq V^\pi(s) \leq \mathrm{E}[R_{\pi,H,s}] + \gamma^H V_{max}.$$

Note que existe um compromisso entre fator de desconto γ e horizonte H. Quanto mais próximo de 1 o valor de γ, maior deve ser o horizonte para garantir um erro mínimo ϵ. Dada essa consideração os teoremas abaixo fazem referência ao valor $V^{\pi,H}(s_0) = \mathrm{E}[R_{\pi,H,s_0}]$

Teorema 16. *Em um MDP, considere a variável aleatória $V^{\pi}_{H,s0,N}$ obtida ao avaliar o valor $V^\pi(s_0)$ utilizando o método de Monte Carlo com horizonte H e N amostras. Então:*

- *A probabilidade de o erro ser maior que ϵ tem limite superior:*

$$\delta = \frac{2}{\log \frac{2N\epsilon^2}{V_{max}^2}} = \frac{2}{\log 2 + \log N + 2\log \epsilon - 2 \log V_{max}};$$

- *Com probabilidade δ o erro tem limite superior:*

$$\epsilon = V_{max} \sqrt{\frac{1}{2N} \log \frac{2}{\delta}};$$

e

- *Para garantir que o erro seja no máximo de ϵ com probabilidade menor que δ, a quantidade máxima de amostras necessária é de:*

$$N = \left(\frac{V_{max}}{\epsilon}\right)^2 \frac{1}{2} \log \frac{2}{\delta}.$$

4.3 MCTS

Na seção anterior estudou a estimativa para o valor de uma política, nesta seção será visto como o método de Monte Carlo pode ser utilizado para construir uma política ϵ-ótima.

Considere o algoritmo de Iteração de Valor para horizonte finito visto no capítulo 2. Esse algoritmo baseia-se na seguinte recursão:

$$V(s,d) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') V^*(s', d+1) \right\},$$

onde $V(s,T) = 0$.

Considere as variáveis aleatórias $S^{s,a} \in \mathcal{S}$ para todo par (a,s) que é obtida da distribuição $T(s,a,\cdot)$. Para um par (s,a), considere N amostras $s_1^{s,a}, \ldots, s_N^{s,a}$. Pode-se construir a seguinte variável aleatória:

$$\widehat{V}(s,d) = \max_{a \in \mathcal{A}} \left\{ R(s,a) + \frac{1}{N} \sum_{i=1}^{N} \widehat{V}(s_i^{s,a}, d+1) \right\}.$$

Diferentemente dos algoritmos LAO* e LRTDP que dependem do operador de Bellman para calcular uma função valor. O Método de Monte Carlo exige apenas um modelo generativo.

Definição 18 (Modelo Generativo)**.** *Um modelo generativo para um MDP é um algoritmo aleatório que, recebe como entrada um par estado-ação (s,a) e retorna uma recompensa r e um próximo estado s', onde o estado s' foi sorteado aleatoriamente de acordo com as*

probabilidades de transição $T(s, a, \cdot)$ e a recompensa r foi sorteada (usualmente, determinista) da função de recompensa $R(s, a)$.

Lembre-se que o interesse é apenas na decisão a ser tomada em s_0. O algoritmo Sparse Sampling[1], baseado no algoritmo de Iteração de Valor, implementa o operador de Bellman aplicado a aproximações sucessivas apenas nos estados alcançáveis probabilisticamente a partir do estado s_0. As aproximações são realizadas com a média entre C execuções. Note que, a quantidade de estados alcançáveis a partir de s_0 com horizonte finito H e C simulações por estado alcançado é dada por:

$$\frac{(|\mathcal{A}|C)^{H+1} - |\mathcal{A}|C}{|\mathcal{A}|C - 1}.$$

O principal resultado do algoritmo Sparse Sampling é dado pelo seguinte teorema.

Teorema 17 (Sparse Sampling)**.** *Existe um algoritmo aleatório que, dado acesso a um modelo generativo para um MDP de k-ações, recebe como entrada um estado $s_0 \in \mathcal{S}$ e qualquer valor $\epsilon > 0$, retorna uma ação que satisfaz as seguintes condições:*

- *a complexidade computacional é $O((kC)^H)$, onde:*

$$\lambda = \frac{\epsilon(1-\gamma)^2}{4}, \qquad H = \left\lceil \log_\gamma \left(\frac{\lambda}{V_{max}} \right) \right\rceil$$

$$C = \frac{V_{max}^2}{\lambda^2} \left(2T \log \frac{kHV_{max}^2}{\lambda^2} + \log \frac{C_{max}}{\lambda} \right)$$

[1]No artigo "A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes" os autores chamam o algoritmo apenas por algoritmo $\mathcal{A}$, aqui aproveita-se o nome do artigo para dar nome ao algoritmo.

- *(Quase-Otimalidade) O valor V da função da política estocástica implementada por tal algoritmo satisfaz simultaneamente para todo $s \in \mathcal{S}$:*

$$|V(s) - V^*(s)| \leq \epsilon.$$

Algoritmo Sparse Sampling

1. calcule $H = f_H(\gamma, \epsilon, R_{max})$
2. calcule $C = f_C(\gamma, \epsilon, R_{max})$
3. para cada $a \in \mathcal{A}$
   a) Sorteie C amostras $s_1^{s_0,a}, \ldots, s_C^{s_0,a}$
   b) Para cada $s_i^{s_0,a}$
      i. $\widehat{V}(s_i^{s_0,a}) = EstimateV(s_i^{s_0,a}, C, H, 1)$
   c) $\widehat{Q}(a) = R(s_0, a) + \frac{1}{C}\gamma \sum_{i=1}^{C} \widehat{V}(s_i^{s_0,a})$
4. retorna $\arg\max_{a \in \mathcal{A}} \widehat{Q}(a)$

EstimateV(s, C, H, d)

1. se $d = H$ retorna 0
2. para cada $a \in \mathcal{A}$
   a) Sorteie C amostras $s_1^{s,a}, \ldots, s_C^{s,a}$
   b) Para cada $s_i^{s,a}$
      i. $\widehat{V}(s_i^{s,a}) = EstimateV(s_i^{s,a}, C, H, d+1)$
   c) $\widehat{Q}(a) = R(s,a) + \frac{1}{C}\gamma \sum_{i=1}^{C} \widehat{V}(s_i^{s,a})$
3. retorna $\max_{a \in \mathcal{A}} \widehat{Q}(a)$

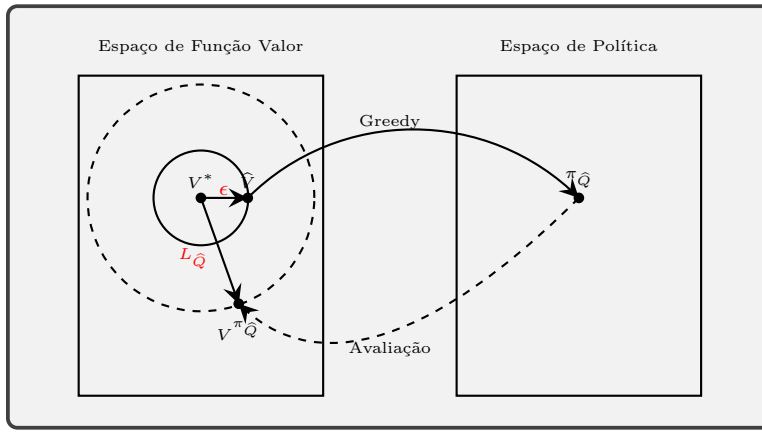
Algoritmos baseados no método de Monte Carlo baseados em Iteração de Valor estima uma função valor $\widehat{Q}$ e a partir dessa função valor constrói-se uma política $\pi_{\widehat{Q}}$.

A função valor $\widehat{Q}(s, a, d)$ estima o valor da função qualidade:

$$Q(s, a, d) = R(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s')V^*(s', d+1),$$

que designa o valor de executar a ação a no estado s na profundidade d e seguir dali para frente executando a política ótima. Dessa forma, a função qualidade permite avaliar qual ação deve ser executada em cada estado.

Pode-se analisar esses algoritmos sob dois aspectos: qualidade da função valor $\widehat{V}(s, d) = \max_{a \in \mathcal{A}} \widehat{Q}(s, a, d)$ estimada e qualidade da política gerada $\pi_{\widehat{Q}}$. O seguinte teorema discorre sobre a relação entre um erro máximo ϵ garantido na função valor e a distância $L_{\widehat{Q}}$ entre o valor ótimo e o valor da política gerada $\pi_{\widehat{Q}}$.



Teorema 18. *Seja V^* a função valor ótima para um MDP com fator de desconto $\gamma < 1$. Seja $\widehat{Q}$ uma função tal que $|\widehat{Q}(s,a) - Q^*(s,a)| \leq \epsilon$ para todo $s \in \mathcal{S}$ e $a \in \mathcal{A}$. Considere a política $\pi_{\widehat{Q}}$ obtida de forma gulosa a partir de $\widehat{Q}$ com valor $V^{\pi_{\widehat{Q}}}$. Então para todo $s \in \mathcal{S}$:*

$$|V^{\pi_{\widehat{Q}}}(s) - V^*(s)| \leq \frac{2\epsilon}{1-\gamma}$$

Demonstração. Defina $L_{\widehat{Q}}(s) = |V^{\pi_{\widehat{Q}}}(s) - V^*(s)|$. Existe um estado z que alcança a perda máxima, isto é, $L_{\widehat{Q}}(z) \geq L_{\widehat{Q}}(s)$ para todo $s \in \mathcal{S}$. Para esse estado z, considere a ação ótima $a = \pi^*(z)$ e a ação especificada por $\pi_{\widehat{Q}}$, isto é, $b = \pi_{\widehat{Q}}(z)$. Então:

$$\begin{aligned} Q^*(z,a) - \epsilon &\leq Q^*(z,b) + \epsilon \\ R(z,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V^*(s') - \epsilon &\leq R(z,b) + \gamma \sum_{s' \in \mathcal{S}} T(s,b,s') V^*(s') + \epsilon \end{aligned}$$

logo

$$R(z,a) - R(z,b) \leq 2\epsilon + \gamma \sum_{s' \in \mathcal{S}} (T(s,b,s') V^*(s') - T(s,a,s') V^*(s'))$$

Então:

$$\begin{aligned}
L_{\widehat{Q}}(z) &= V^*(s) - V^{\pi_{\widehat{Q}}}(s) \\
&= R(z,a) - R(z,b) + \gamma \sum_{s' \in \mathcal{S}} (T(s,a,s')V^*(s') - T(s,b,s')V^{\pi_{\widehat{Q}}}(s')) \\
&\leq 2\epsilon + \gamma \sum_{s' \in \mathcal{S}} (T(s,b,s')V^*(s') - T(s,a,s')V^*(s')) \\
&\quad + \gamma \sum_{s \in \mathcal{S}} (T(s,a,s')V^*(s') - T(s,b,s')V^{\pi_{\widehat{Q}}}(s')) \\
&= 2\epsilon + \gamma \sum_{s' \in \mathcal{S}} (T(s,b,s')(V^*(s') - V^{\pi_{\widehat{Q}}}(s')) \\
&\leq 2\epsilon + \gamma \sum_{s' \in \mathcal{S}} (T(s,b,s')L_{\widehat{Q}}(z) = 2\epsilon + \gamma L_{\widehat{Q}}(z) \\
&\leq \frac{2\epsilon}{1-\gamma}
\end{aligned}$$

□

Enquanto o teorema anterior nos dá informação sobre a política derivada de uma função aproximada que contém um erro ϵ, os métodos de Monte Carlo só garantem esse erro com probabilidade $1 - \delta$. Cada vez que um algoritmo é executado, erros e políticas diferentes são obtidas, podendo ser considerada uma política probabilística. O teorema seguinte permite estabelecer propriedades sobre essa política probabilística.

Teorema 19. *Assuma que $\pi_{\widehat{Q}}(s)$ é uma política gulosa resultante de um algoritmo baseado no método de Monte Carlo com base na estimativa $\widehat{Q}$, isto é, $\pi_{\widehat{Q}}(s)$ é uma variável aleatória. Se para todo $s \in \mathcal{S}$ é verdade que:*

$$\Pr(|\widehat{Q}(s,\pi^*(s)) - \widehat{Q}(s,\pi_{\widehat{Q}})| \leq \epsilon) \leq 1 - \delta,$$

então:

$$V^*(s) - V^{\pi_{\widehat{Q}}}(s) \leq \frac{2\epsilon + 2\delta(2-\delta)V_{max}}{1-\gamma}.$$

Demonstração. Considere que para um estado s qualquer, tem-se $z = \pi_{\widehat{V}}(s)$, então, com probabilidade $(1-\beta) = (1-\delta)^2$:

$$\begin{aligned} \widehat{Q}(s,z) &\geq \widehat{Q}(s,a)) \\ Q^*(s,z) + \epsilon &\geq Q^*(s,a) - \epsilon \\ Q^*(s,a) - Q^*(s,z) &\leq 2\epsilon \end{aligned}$$

Logo, tem-se:

$$\begin{aligned} \mathrm{E}[Q^*(s, \pi_{\widehat{Q}}(s))] &\geq (1-\beta)(Q^*(s,\pi^*(s)) - 2\epsilon) - \beta V_{max} \\ &\geq Q^*(s,\pi^*(s)) - 2\epsilon - 2\beta V_{max} \end{aligned}$$

Seguindo como na prova do teorema anterior:

$$L_{\widehat{Q}}(z) = V^*(s) - V^{\pi_{\widehat{Q}}}(s) \leq \frac{2\epsilon + 2\beta V_{max}}{1-\gamma} = \frac{2\epsilon + 2\delta(2-\delta)V_{max}}{1-\gamma}.$$

□

4.4 Algoritmo UCT

Note que as simulações realizadas pelo algoritmo Sparse Sampling são uniformes para todas ações e estados alcançados no horizonte H, mesmo que as ações não sejam ótima. Por outro lado, os algoritmos LAO* e LRTDP procuram focar apenas nos estados alcançáveis a partir da política ótima. Ainda, como o algoritmo Sparse Sampling é recursivo, para ele ser utilizado em uma visão on-line, deve-se conhecer de antemão o tempo disponível por decisão e escolher a quantidade de simulações e profundidade restritos a esse tempo.

Em vez de considerar a recursão do algoritmo Sparse Sampling, nesta seção serão considerados algoritmos baseados em Rollouts. Um rollout é a simulação de um episódio inteiro a partir do estado inicial. A árvore é criada incrementalmente adicionando informação obtidas com cada rollout.

Algoritmo MC Rollout

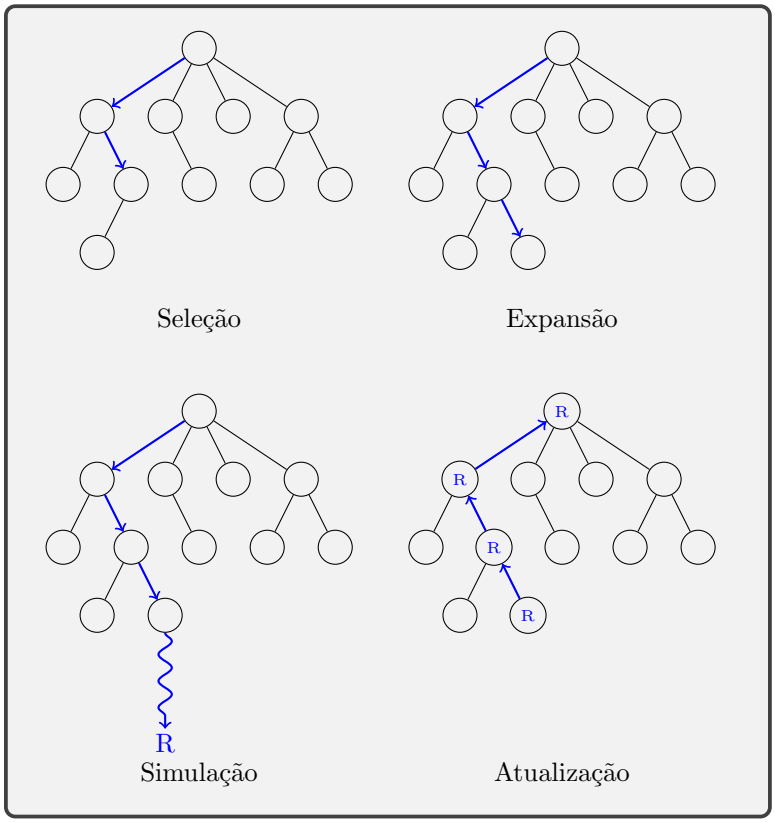
1. enquanto não *timeout*
   a) $search(s_0, 0)$
2. retorna $bestAction(s_0, 0)$

search(s,d)

1. se $Terminal(s)$ então retorne 0
2. se $Leaf(s, d)$ então retorne $Evaluate(s)$
3. $a = selectAction(s, d)$
4. $(s', r) = simulateAction(s, a)$
5. $q = r + \gamma \times search(s', d + 1)$
6. $updateValue(s, a, d, q)$
7. retorne q

Nesse algoritmo deve-se especificar: como a árvore é expandida ao encontrar um nó folha ($Leaf(s, d)$), como avaliar um nó folha ($Evaluate(s)$), como selecionar ações nos nós internos da árvore ($selectAction(s, d)$) e como os nós da árvore são atualizados ($updateValue(s, d)$). Um episódio finaliza quando um estado terminal é encontrado, que pode ser: uma meta, um horizonte que acabou, ou mesmo probabilístico; ou quando é realizada uma avaliação.

Em cada rollout o algoritmo passa pelas seguintes fases: Seleção, Expansão, Simulação e Atualização.



O algoritmo UCT é uma variação do esquema utilizando Monte Carlo com Rollout. A principal ideia desse algoritmo é amostrar as ações de forma seletiva para melhorar o desempenho. Em termos de desempenho deseja-se: um probabilidade pequena de gerar erros grandes se o algoritmo pare prematuramente, e convergência para a ação ótima se tempo suficiente é dado.

O algoritmo UCT realiza um compromisso entre: (i) testar alternativas que parecem ser as melhores naquele momento para me-

lhorar sua precisão; e (*ii*) testar alternativas que, no momento, parecem ser não ótimas para garantir que nenhuma alternativa ótima é descartada por conta de erros nas estimativas iniciais. Esse compromisso é colocado na literatura como o dilema entre *exploration* (descobrir as boas ações) e *exploitation* (executar a ação que se acredita ser melhor segundo a estimativa).

Nesse algoritmo, cada nó da árvore representa um par (estado,profundidade) e armazena:

- $n(s, a, d)$: número de vezes que a ação a foi executada no estado s no nível d;
- $n(s, d)$: número de vezes que o estado s foi visitado no nível d; e
- $\widehat{Q}(s, a, d)$: recompensa média recebida nas trajetórias executadas após executar a ação a no estado s no nível d.

Cada vez que um par (estado,profundidade) é alcançado e esse não pertence à árvore, um nó para esse par é adicionado à árvore e o estado s é avaliado. Essa avaliação é feita segundo uma política de avaliação, que pode ser simplesmente aleatória ou levando em conta alguma política inicial específica para o problema.

A política de avaliação tem um papel muito importante na qualidade da ação retornada pelo algoritmo para ser executada no estado inicial. A condição, para que uma política de avaliação gere um bom resultado, é que o valor dos estados segundo essa política possam ser diferenciados entre si. Por isso, o algoritmo UCT é bastante utilizado em jogos de tabuleiro, onde pode-se considerar sempre um lado vencedor e um lado perdedor, garantindo que recompensas diferentes sejam sempre geradas para o treinamento mesmo sob políticas aleatórias.

As ações para nós internos à árvore são selecionadas segundo a expressão abaixo:

$$\pi_{UCT}(s,d) = \arg\max_{a \in \mathcal{A}} \left\{ Q(s,a,d) + \beta \sqrt{\frac{\ln n(s,d)}{n(s,a,d)}} \right\}.$$

Nela, uma ação é avaliada realizando o compromisso entre *exploitation*, indicada pela estimativa da função qualidade $\widehat{Q}(s,a,d)$, e *exploration*, indicada pelo termo $\sqrt{\frac{\ln n(s,d)}{n(s,a,d)}}$; o fator β pondera esses dois termos. O termo de exploração é baseado no algoritmo UCB1 (*Upper Confidence Bounds*) e indicada um limite superior.

Algoritmo UTC

1. enquanto não *timeout*
   a) $search(s_0, 0)$
2. retorna $bestAction(s_0, 0)$

search(s,d)

1. se $Terminal(s)$ então retorne 0
2. se $Leaf(s, d)$ então retorne $Evaluate(s)$
3. $a = \arg\max_{a \in \mathcal{A}} \left\{ Q(s, a, d) + \beta \sqrt{\frac{\ln n(s, d)}{n(s, a, d)}} \right\}$
4. $(s', r) = simulateAction(s, a)$
5. $q = r + \gamma \times search(s', d + 1)$
6. $\widehat{Q}(s, a, d) \leftarrow \frac{n(s, a, d)Q(s, a, d) + q}{n(s, a, d) + 1}$
7. $n(s, a, d) \leftarrow n(s, a, d) + 1$
8. $n(s, d) \leftarrow n(s, d) + 1$
9. retorne q

O algoritmo UCT garante que a política obtida é quasi-ótima e que a complexidade do algoritmo não depende da quantidade de estados. Esse resultado é colocado no seguinte teorema.

Teorema 20. *Consider a finite-horizon MDP with rewards scaled to lie in the* $[0, 1]$ *interval. Let the horizon of the MDP be* D*, and*

the number of actions per state be K. Consider algorithm UCT such that the bias terms of UCB1 are multiplied by D. Then the bias of the estimated expected payoff, $\bar{X}_n$, is $O(log(n)/n)$. Further, the failure probability at the root converges to zero at a polynomial rate as the number of episodes grows to infinity.

4.5 Multi-Armed Bandit e o algoritmo UCB1

O algoritmo UCB1 (*Upper Confidence Bounds*) é uma solução para o problema de Multi-Armed Bandit. Considere k variáveis aleatórias $X_1, \ldots, X_k$ (Arm Bandits) e respectivas esperanças $\mu_1, \ldots, \mu_k$; sendo que essas esperanças são desconhecidas. Em cada tempo $t \in \{0, 1, 2, \ldots\}$, apenas a amostra de uma dessas variáveis pode ser observada. Então, no tempo t, cada variável aleatória X_i foi amostrada s_i vezes, sendo que $\sum_{i=1}^{k} s_i = t$, e tem uma esperança estimada $\overline{X}_{i,s_i}$. Utilizando a inequalidade de Hoeffding, tem-se:

$$\Pr\left(\mu_i \geq \overline{X}_{i,s_i} + \sqrt{2\frac{\ln t}{s_i}}\right) \leq t^{-4}.$$

Um algoritmo para o problema de Multi-Armed Bandit escolhe qual variável aleatória observar com o objetivo de minimizar o arrependimento (*regret*) de não ter escolhido a variável aleatória ótima, isto é, com maior esperança. Considere uma estratégia $a(\cdot) \in \{1, \ldots, k\}$ de escolher uma variável aleatória para observar; tal estratégia em geral depende das amostras realizadas antes do tempo t. O arrependimento é uma variável aleatória que é dada por:

$$\max_{i \in \{1,\ldots,k\}} t\mu_i - \sum_{i=0}^{t-1} \mu_{a(i)}.$$

O algoritmo UCB1 escolhe a variável aleatória que maximiza a seguinte função de avaliação:

$$q(i) = \overline{X}_{i,s_i} + \sqrt{2\frac{\ln t}{s_i}},$$

e garante que o arrependimento esperado cresce apenas de forma logarítmica com o tempo t, isto é,

$$\max_{i\in\{1,\ldots,k\}} t\mu_i - \mathrm{E}\left[\sum_{i=0}^{t-1} \mu_{a(i)}\right] < A\log t + B,$$

onde A e B são constantes. Além disso, é provado que o crescimento logarítmico é o mais lento possível.