Cuarta Parte:
Clase 3 – *Lookaheads*

Optimización Dinámica - ICS

Mathias Klapp

¿Qué hemos visto?

- Introducción a ADP

- Enfoque Miope (decisión reactiva)

Still,… mere reactions are not suitable. (Powell et al. 2000).

Recordemos

En cada etapa t y estado s_t buscamos una decisión tal que:

$$d_t(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x) + Q(y_t(s_t, x))\}$$

El enfoque miope (o reactivo) asume $Q_t(y_t) = 0$.

¿Cómo diseñar una política proactiva?

- *Políticas de lookahead* directo
- *Políticas de roll-out*
- *Políticas de horizonte rodante*
- *Gestión dinámica de recursos móviles*

Lookahead

- *Lookahead* = vistazo (aproximado) del futuro.

- Familia de enfoques ***online*** que intentan estimar la función Q mediante una **simulación online** del MDP desde el estado s_t y la etapa t en adelante durante p etapas.

Lookahead directo de una etapa ($p = 1$)

En etapa t y estado s_t se toma decisión x estimando a Q mediante el costo inmediato de la decisión en la etapa posterior. Es decir:

$$d_t^{1L}(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x) + \bar{Q}_t(y_t(s_t, x))\}$$

, donde:

$$\bar{Q}_t(y_t) = \mathbb{E}_{s_{t+1}} \left(\underset{x' \in \mathbb{X}_{t+1}(s_{t+1})}{max} r_t(s_{t+1}, x') \middle| y_t \right)$$

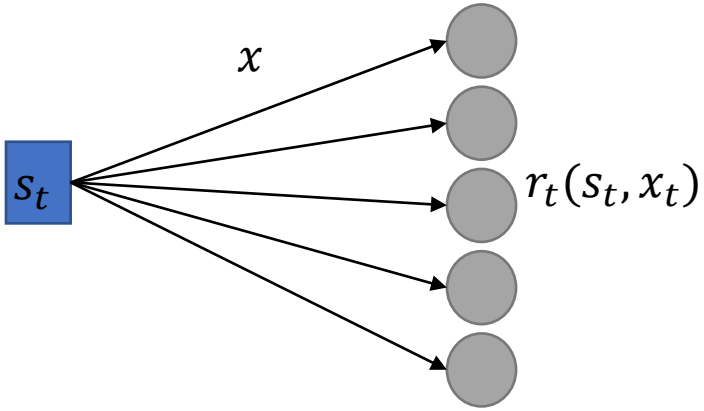
- Para evitar maldición de dimensionalidad, **simula en línea** una muestra $\omega \in \Omega'(y_t)$ de estados $s_{t+1}(\omega')$ para cada estado posible de post-decisión y_t :

$$\bar{Q}_t(y_t) \approx \frac{1}{|\Omega'(y_t)|} \sum_{\omega' \in \Omega'(y_t)} \left(\underset{x' \in \mathbb{X}_{t+1}(s_{t+1}(\omega'))}{max} r_t(s_{t+1}(\omega'), x') \right)$$

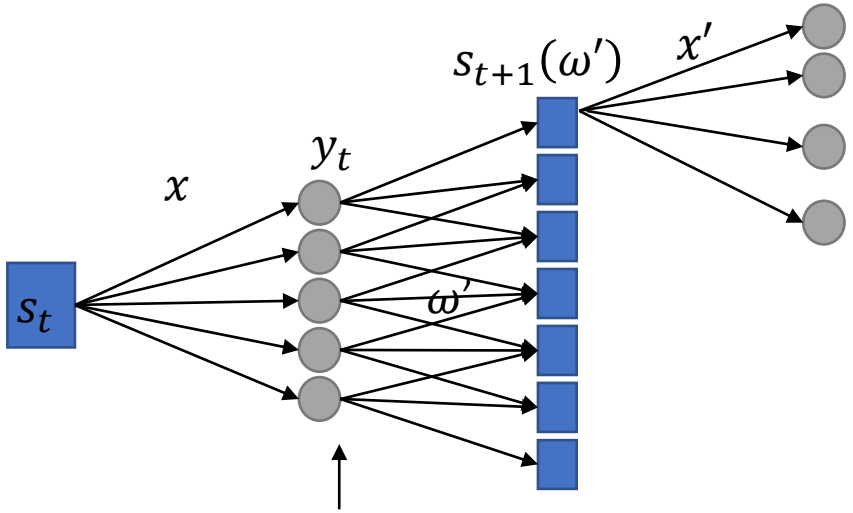
- En palabras simples, estima el *value-to-go* descartando ganancias posteriores a etapa $t + 1$.

Política de *lookahead* directo versus miope

Miope:



One-step direct lookahead:



simulación online para cada y_t potencial a visitar en s_t

- x' no es implementada directamente, sólo se computa para estimar Q. Podría incluso ser una solución infactible o relajada.

Simulación online v/s de ejecución

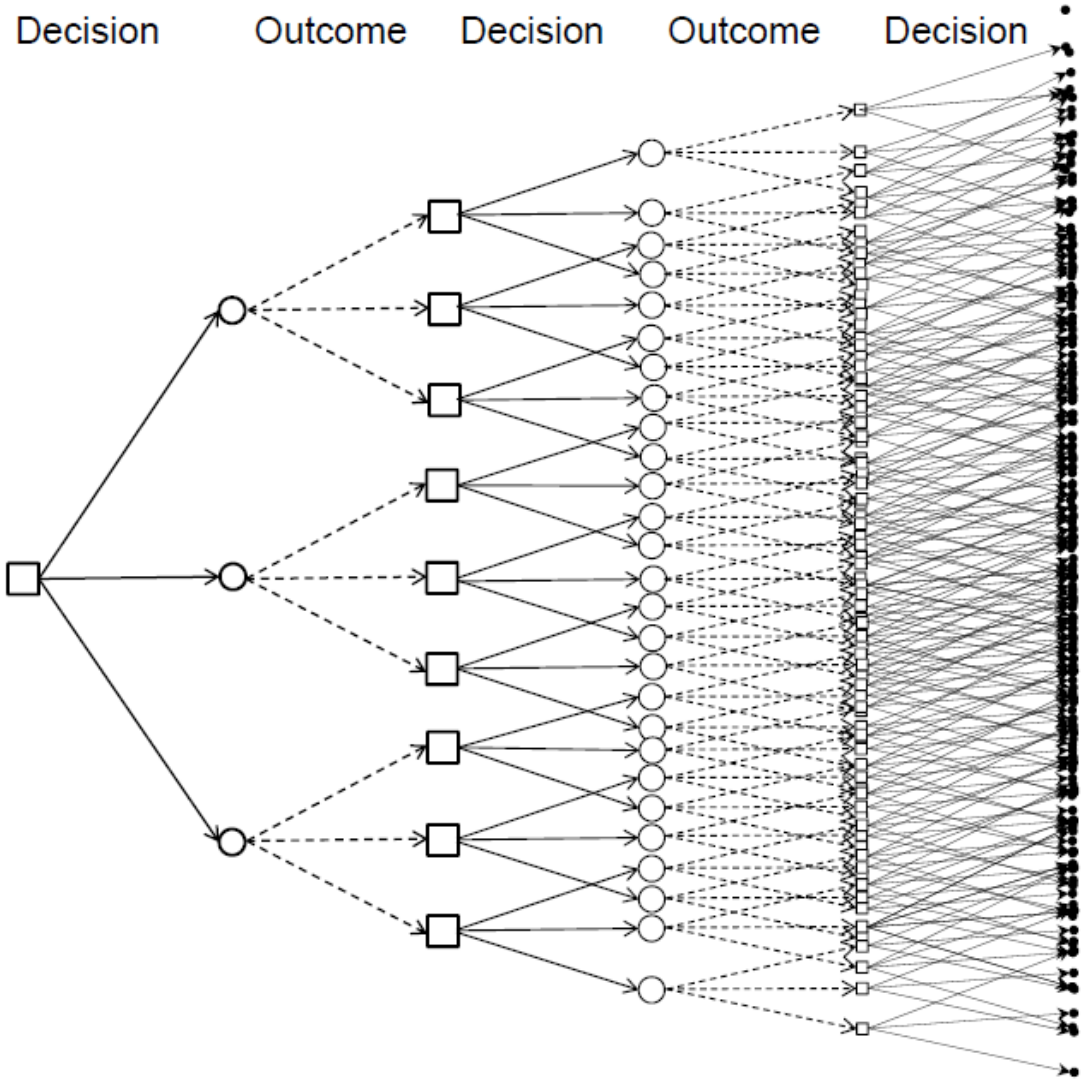
Simulación de ejecución: Simular ejecución del MDP aplicando una determinada política π para estimar su valor futuro esperado $\mathrm{V}^{\pi}(s_1)$ mediante $\frac{\sum_{\omega} \mathrm{V}_{\omega}^{\pi}(s_1)}{m}$.

- No es necesaria al ejecutar en la realidad. Solo para estimar costo.

Simulación online: Simular potenciales escenarios futuros en una ejecución particular del MDP desde una etapa t y estado s_t para estimar el value-to-go $\mathbb{E}_{s+1}(V_{t+1}(s_{t+1})|y_t)$ y evaluar una decisión-.

- Es una simulación ``dentro´´ de la ejecución.
- Necesaria para ejecutar en la realidad (desde ella se diseá la política).

2-step direct lookahead ($p = 2$)



p-step direct lookahead

- Simula $p \geq 1$ pasos hacia el futuro.
  - Cada paso implica aumento explosivo del requerimiento de simulación que depende de cardinalidad de estados de post-decisión:
    - 1 paso: $|\Omega||\mathbb{Y}_t|$ simulaciones *online*
    - 2 pasos: $|\Omega|^2|\mathbb{Y}_t||\mathbb{Y}_{t+1}|$ simulaciones *online*

- Adaptive ***multi-step direct lookahead:*** Tamaño de muestra variable por estapa y estado.

- Multistage scenario trees (*SDDP*)

- En esencia, es fuerza bruta truncada.
  - Optimiza ``aproximadamente'' el árbol del MDP simulado con p pasos de profundidad desde etapa t a la etapa $t + p$.
  - El MDP a optimalidad es un T*-step direct lookahead* sin simulación.

Direct Lookahead

- Ventajas:
  - Proactivo a corto plazo: Contabiliza impacto en futuro cercano $p - \text{etapas}$.
  - Permite paralelización computacional.
  - No requiere mayor calibración. Salvo p y el tamaño de muestra.
  - Toma decisiones óptimas en un MDP simulado y truncado.

- Desventajas:
  - Trunca el futuro.
  - 100% *on-line,* todo su cómputo es hecho en tiempo de ejecución.
  - N'umero de simulaciones crece exponencialmente con el número de pasos.

- Recomendación del Chef:
  - Funciona cuando impacto a futuro se diluye en pocas etapas y cuando el número de estados de post-decisión es manejable.
  - También funciona bien como heurística de DPs determinísticos

Ejemplo: Dos vecinos más cercanos para el TSP

$$d^{2NN}(i,S) \leftarrow \begin{cases} \underset{j \in S \setminus \{1\}}{\operatorname{argmin}} \{c_{ij} + \min_{k \in S \setminus \{1,j\}} \{c_{jk}\}\} \\ c_{i1} \qquad si\ S = \{1\} \end{cases}$$

Equivale a:

$$d^{2NN}(i,S) \in \left\{ j : (j,k) \in \underset{j,k \in S \setminus \{1\}}{\operatorname{argmin}} \{c_{ij} + c_{jk}\} \right\}$$

Ejecutable en $\mathcal{O}(n^3)$ operaciones, requiere hacer $\mathcal{O}(n^2)$ en cada decisión.

Ejemplos de *lookaheads* en MDPs

- Sistema de inventario infinito que planifica simulando online el costo mínimo esperado de los próximos p periodos de demanda a futuro.

- Sistema de generación Hidroeléctrica que considera p días de pronósticos futuros del tiempo para estimar acciones sobre represa.

- Decisiones de compra y venta de acciones con simulación futura de p etapas de escenarios futuros.

Roll-out (Lookahead indirecto)

- *roll-out:* desenvolver
- Es desenvolver una política fija a futuro desde una etapa en adelante como proxy al value-to-go.

Roll-out

- En etapa t y estado s_t ejecuta decisión:

$$d_t^{Roll}(s_t) \in \underset{x \in \mathbb{X}_t(s_t)}{\operatorname{argmax}} \{r_t(s_t, x) + \bar{Q}_t(y_t(s_t, x))\}$$

, donde $\bar{Q}_t(y_t) = \mathbb{E}_{s_{t+1}}\left(V^{\pi'}(s_{t+1}) \middle| y_t\right)$

- Es decir, aproxima el value-to-go $V(s_{t+1})$ con el valor $V^{\pi'}(s_{t+1})$ de una política base π' ejecutada desde la etapa $t+1$.

- Para evitar maldición de diemensionalidad, simula una muestra *online* $\Omega'(y_t)$ de múltiples corridas desde $t+1$ hasta T utilizando π':

$$\bar{Q}_t(y_t) \approx \frac{1}{|\Omega'(y_t)|} \sum_{\omega' \in \Omega'(y_t)} V_{t+1}^{\pi'}(\omega')$$

Roll-out

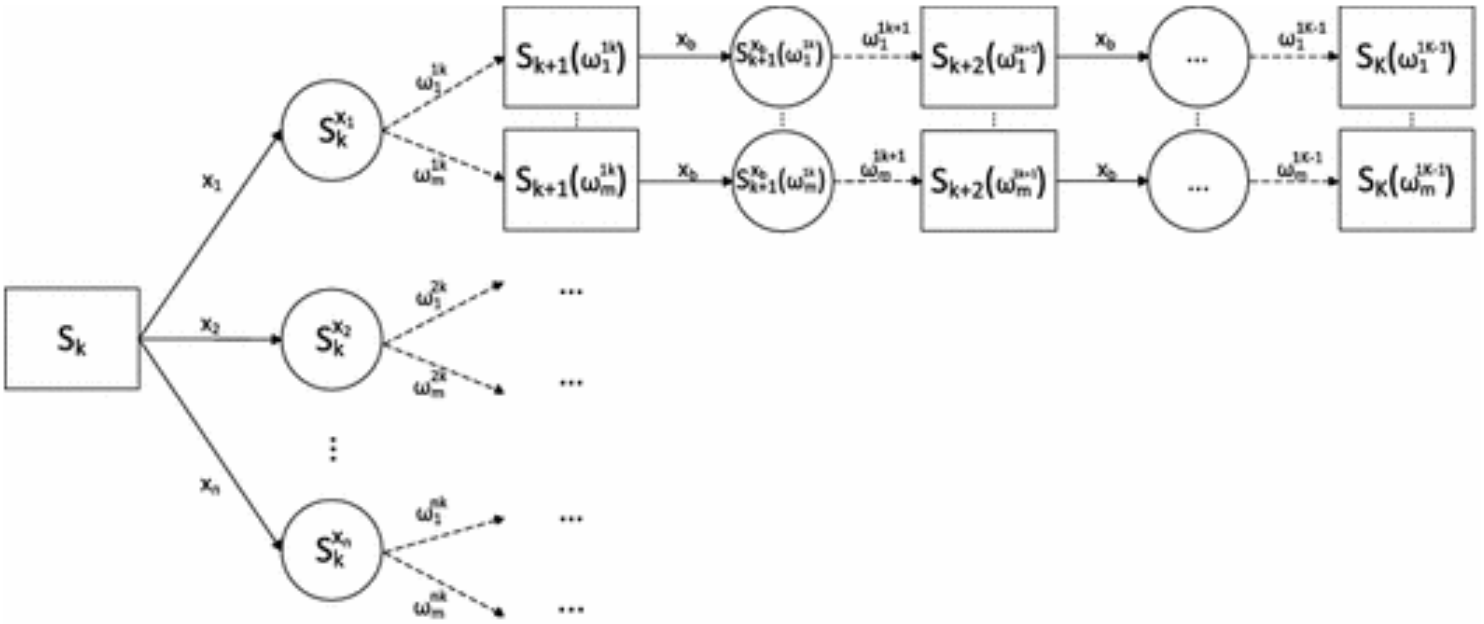


Fig. 6.1 Post-decision rollout algorithm

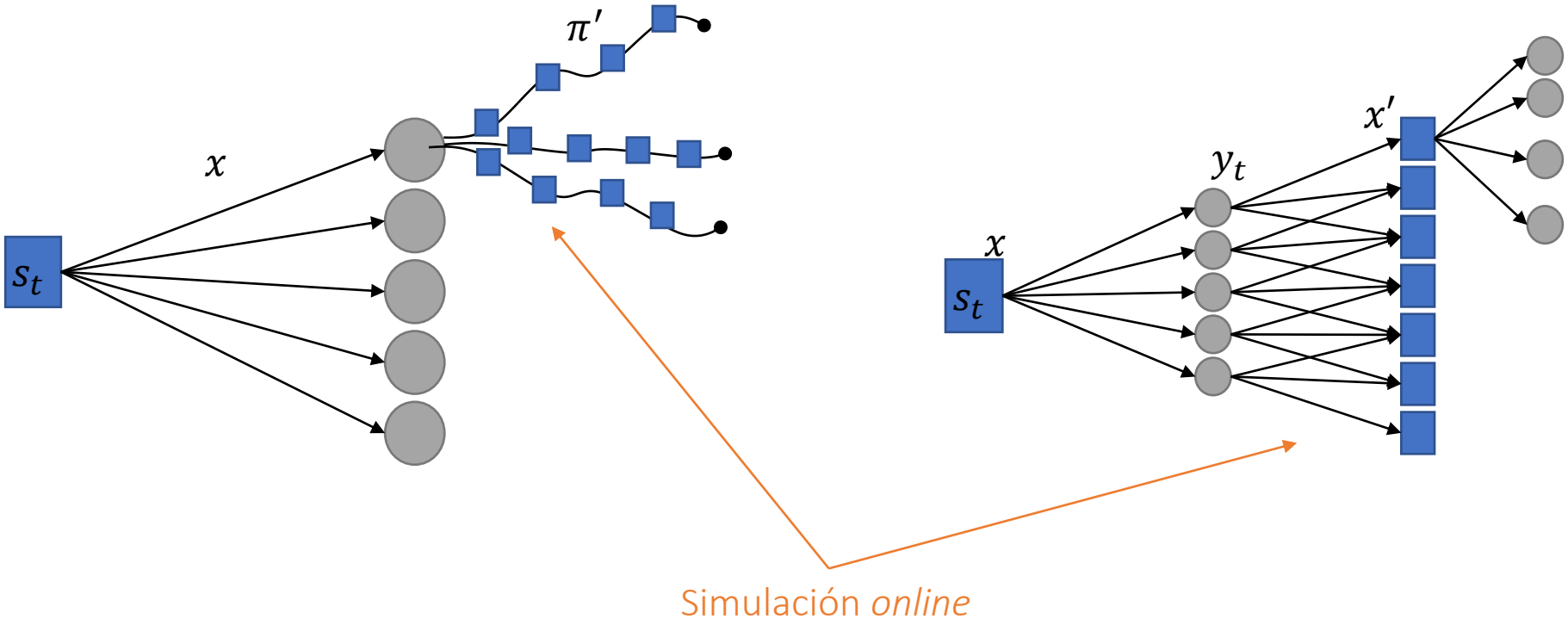
Fuente: Ulmer (2016)

En esta figura el estado de post-decisión $y_t(s_t, x_t)$ se denota como $s_t^{x_t}$

R*oll-out* versus *lookahead* directo

roll-out:

One-step direct lookahead:



- Política π' no es implementada, sólo se computa para estimar Q.
- *Roll-out* sería óptimo si π' fuese óptima y cálculo de esperanzas fuese exacto.

Roll-out

Ventaja:

- No trunca el futuro. Estima Q hasta la etapa terminal.
- Permite paralelización.

Desventaja:

- Requiere una buena política base π' como input (cercana a óptima).
- Depende de exigencias de cómputo de la política base π'**.**
- 100% *on-line,* todo su cómputo es hecho en tiempo de ejecución.

Recomendación del Chef:

- Cuando el impacto no se diluye rápidamente en el futuro
- Funciona bien si existe una regla simple que aproxima razonablemente el value-to-go.

Resultado teórico del *roll-out* (Bertsekas)

Sea π^{Roll} la política resultante de un roll-out sobre la política base π'.

Un *roll-out* nunca es peor que su política base π'

Para todo $t \in \{1, \dots, T\}$ y $s_t \in \mathbb{S}_t$: $V_t^{\pi^{Roll}}(s_t) \geq V_t^{\pi'}(s_t)$

Nota: Resultado assume estimación exacta del value-to-go de su política base

Demostración por inducción:

TERMINAL: $V_T^{\pi^{Roll}}(s_T) = V_T^{\pi'}(s_T) = r_T(s_T)$

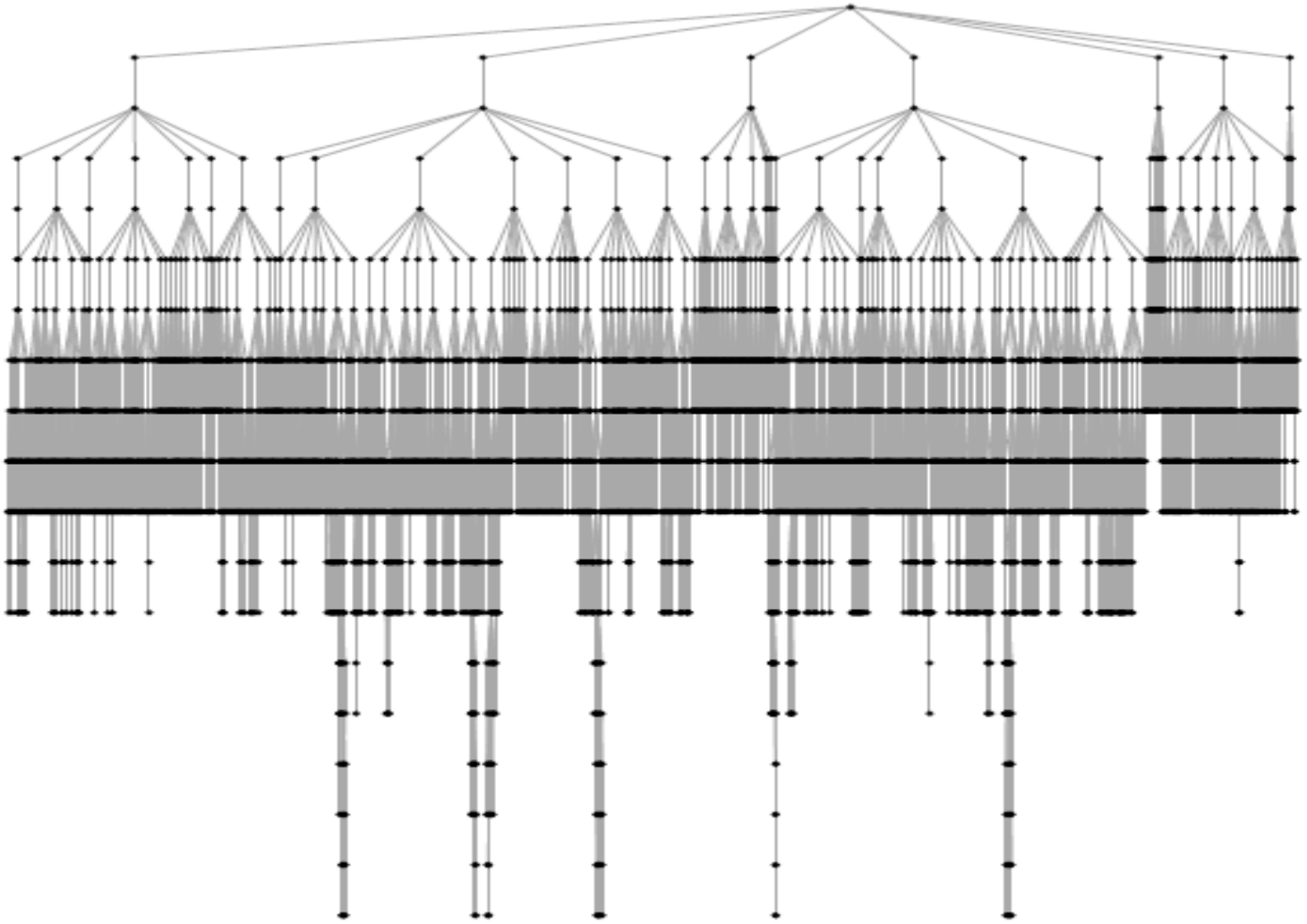
HI: $V_{t+1}^{\pi^{Roll}}(s_{t+1}) \geq V_{t+1}^{\pi'}(s_{t+1})$

PRUEBA:

$$
\begin{aligned}
V_t^{\pi^{Roll}}(s_t) &= r_t\left(s_t, x_t^{\pi^{Roll}}\right) + \mathbb{E}_{s_{t+1}}\left[V_{t+1}^{\pi^{Roll}}(s_{t+1}) \middle| y_t\left(s_t, x_t^{Roll}\right)\right] \\
&\geq r_t\left(s_t, x_t^{\pi^{Roll}}\right) + \mathbb{E}_{s_{t+1}}\left[V_{t+1}^{\pi'}(s_{t+1}) \middle| y_t\left(s_t, x_t^{Roll}\right)\right] \\
&= \max_{x \in \mathbb{X}_t(s_t)} \left\{ r_t(s_t, x) + \mathbb{E}_{s_{t+1}}\left[V_{t+1}^{\pi'}(s_{t+1}) \middle| y_t(s_t, x)\right]\right\} \\
&\geq r_t\left(s_t, x_t^{\pi'}\right) + \mathbb{E}_{s_{t+1}}\left[V_{t+1}^{\pi'}(s_{t+1}) \middle| y_t(s_t, x_t')\right] \\
&= V_t^{\pi'}(s_t)
\end{aligned}
$$

Simulación computacional puede distorsionar esta propiedad.

Lookahead híbrido: *lookahead directo* **+** *roll-out* terminal



Aplicación de MCTS a Go

https://deepmind.com/research/alphago/

ALPHAGO: programa equipado con MCTS que juega *Go*.

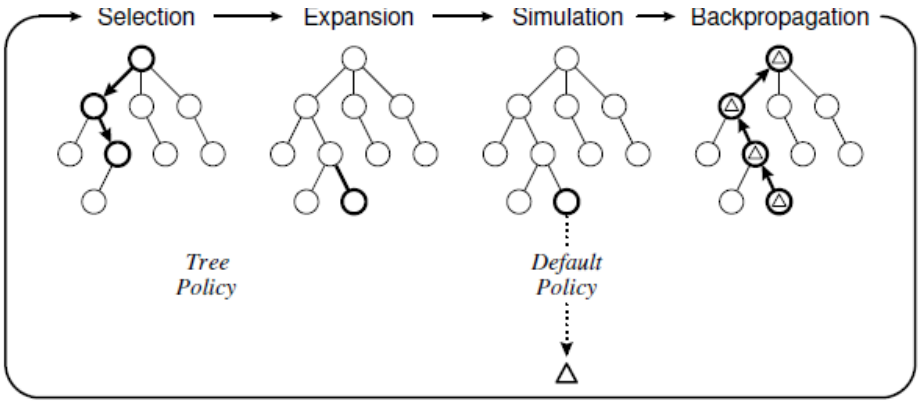
- En 2015 fue el primer computador en ganarle 5-0 a un jugador profesional (campeón Europeo).
- Actualmente, AlphaGo ha derrotado varias veces al campeón mundial y es considerado el mejor jugador de Go de todos los tiempos.
- Combina estas heurísticas y una gran capacidad de computo.
- Comput*o* masivo: 1920 CPUs y 280 GPUs, $3000USD de cuenta eléctrica por juego.

Monte Carlo tree search (MCTS)

En escencia, es un ***lookahead* híbrido adaptativo.** Funciona bien para problema de muchas etapas y decisiones discretas, *e.g. juegos de mesa*.

Explicado de forma simple evalúa así una estado en alguna parte del árbol simulado de *lookahead*:

- Si estado no se ha visitado antes en la historia se el asigna el valor de un rollout (aleatorio o por defecto).
- Si estado se ha visitado antes se simula expansión.
- Al cerrar el árbol se evalúa el valor la acción inicial (Backpropagation)



Ver https://link.springer.com/referenceworkentry/10.1007%2F978-3-319-08234-9_12-1

Monte Carlo tree search (MCTS)

Recomiendo este video si quiere aprender:

https://www.youtube.com/watch?v=UXW2yZndl7U

https://www.youtube.com/watch?v=Fbs4lnGLS8M

Cuarta Parte:
Clase 3 – *Lookaheads*

Optimización Dinámica - ICS

Mathias Klapp