NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta a)

Queremos demostrar que:

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot q_\pi(s,a)$$

Demostración:

La definición de $v_\pi(s)$ es:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s]$$

Expresando el valor esperado condicional como una suma sobre las posibles acciones en el estado s, usando la propiedad de la esperanza total (probabilidades totales):

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \mathbb{P}(A_t = a|S_t = s) \cdot \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Ahora usando la definición $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$, se tiene:

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Finalmente aplicando la definición $q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$, se tiene lo que se quería demostrar:

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \cdot q_\pi(s,a)$$

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta b)

Se pide demostrar que:

$$q_\pi(s,a) = \sum_{r\in\mathcal{R}} \sum_{s'\in\mathcal{S}} p(r,s'|s,a)\cdot(r+\gamma v_\pi(s'))$$

Demostración:

En clases vimos que:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')]$$

Pero también en el punto **a)** demostramos recién que:

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)} \pi(a|s)\cdot q_\pi(s,a)$$

Luego si igualamos estas dos tenemos que:

$$\sum_{a\in\mathcal{A}(s)} \pi(a|s)\cdot q_\pi(s,a) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')]$$

Si quitamos la sumatoria y nos fijamos en una acción a (no imposible, es decir $\pi(a|s) > 0$) en particular obtenemos:

$$\pi(a|s)\cdot q_\pi(s,a) = \pi(a|s) \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')]$$

Y dividiendo por $\pi(a|s)$:

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')]$$

Lo cual es precisamente lo que se quería demostrar.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta c)

Las dos políticas del enunciado repiten los siguientes ciclos:

- π_l: $s_0 \to s_l$ (recompensa 1), luego $s_l \to s_0$ (recompensa 0).
- π_r: $s_0 \to s_r$ (recompensa 0), luego $s_r \to s_0$ (recompensa 2).

Repiten dichos ciclos indefinidamente cada dos pasos.

Calculando el retorno esperado de cada política (con la suma geométrica):

$$G_{\pi_l} = 1 + 0 \cdot \gamma + \gamma^2 + 0 \cdot \gamma^3 + \gamma^4 + \cdots = 1 + \gamma^2 + \gamma^4 + \cdots = \sum_{k=0}^{\infty} \gamma^{2k} = \frac{1}{1-\gamma^2}$$

$$G_{\pi_r} = 0 + \gamma \cdot 2 + 0 \cdot \gamma^2 + \gamma^3 \cdot 2 + 0 \cdot \gamma^4 + \gamma^5 \cdot 2 + \cdots = \gamma \cdot 2 + \gamma^3 \cdot 2 + \gamma^5 \cdot 2 + \cdots = 2\gamma \sum_{k=0}^{\infty} \gamma^{2k} = \frac{2\gamma}{1-\gamma^2}$$

Luego viendo que el denominador de ambas fracciones es el mismo, se puede ignorar y comparar el numerador únicamente:

- Si $2\gamma > 1$, entonces π_r es mejor.
- Si $2\gamma < 1$, entonces π_l es mejor.
- Si $2\gamma = 1$, entonces son iguales.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta d)

A continuación se muestran los resultados de ejecutar los experimentos solicitados en el enunciado:

Problema	**Valor de $V(s_0)$**	**Tiempo (s)**
Grid (3x3)	-8.000	0.002
Grid (4x4)	-18.000	0.008
Grid (5x5)	-37.333	0.022
Grid (6x6)	-60.231	0.045
Grid (7x7)	-93.496	0.089
Grid (8x8)	-131.193	0.163
Grid (9x9)	-180.001	0.282
Grid (10x10)	-233.879	0.456
Cookie (3x3)	5.380	0.199
Cookie (4x4)	2.477	0.605
Cookie (5x5)	1.291	1.453
Cookie (6x6)	0.729	2.905
Cookie (7x7)	0.437	5.434
Cookie (8x8)	0.273	8.802
Cookie (9x9)	0.177	13.777
Cookie (10x10)	0.118	20.670
Gambler ($p = 0.25$)	0.067	0.064
Gambler ($p = 0.4$)	0.284	0.091
Gambler ($p = 0.55$)	0.612	0.126

Table 1: Resultados de Iterative Policy Evaluation con $\theta = 10^{-10}$ para política uniformemente aleatoria.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta e)

Esto es porque el tiempo que toma el algoritmo depende principalmente de dos factores:

- La cantidad de estados posibles, es decir, el tamaño del espacio de estados
- Velocidad de convergencia. Esto quiere decir, que si los valores cambian muy lentamente, entonces el tiempo que tomará hasta que se llegue a estabilizar bajo el umbral de θ será mayor.

El problema más lento fue **CookieProblem** con *size* de 10, con un tiempo total de 20 segundos.
La razón de esto es que El **CookieProblem** tiene un espacio de estados mucho más grande que el **GridProblem**. Cada estado incluye la posición del agente y la posición de la galleta, por lo que el número de estados es aproximadamente $N^2 \cdot N^2 = N^4$, donde N es el tamaño de la grilla.

Esto genera una explosión combinatoria del espacio de estados y, por tanto, muchas más actualizaciones en cada iteración. Además, como la política es aleatoria y las recompensas son esporádicas (solo al comer la galleta), los valores convergen más lento.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta f)

Si bajamos el valor de γ se esperaría que el algoritmo se demore **menos** tiempo debido a que el agente sería más míope. Esto hace que no se tengan tanto en cuenta valores a más largo plazo, permitiendo al agente converger solo por los valores más inmediatos sin tener que considerar todas las opciones lejanas.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta g)

A continuación se muestran los resultados de ejecutar los experimentos solicitados en el enunciado, junto con una pequeña explicación de por que se cree que la política *greedy* es óptima o no:

Problema	$V(s_0)$ **(Greedy)**	**Tiempo (s)**	**¿Es óptima?**
Grid (3x3)	-2.000	0.000	Sí. La política greedy toma el camino más corto al estado terminal, que es el comportamiento óptimo esperado en este problema.
Grid (4x4)	-2.000	0.000	
Grid (5x5)	-4.000	0.000	
Grid (6x6)	-4.000	0.000	
Grid (7x7)	-6.000	0.000	
Grid (8x8)	-6.000	0.000	
Grid (9x9)	-8.000	0.000	
Grid (10x10)	-8.000	0.001	
Cookie (3x3)	48.760	0.081	Probablemente sí. Los valores aumentan fuertemente respecto a la política uniforme, indicando que la política greedy recoge galletas con mayor eficiencia. Para confirmar optimalidad se requiere comparar con Value Iteration.
Cookie (4x4)	35.907	0.273	
Cookie (5x5)	28.197	0.706	
Cookie (6x6)	23.063	1.494	
Cookie (7x7)	19.402	2.743	
Cookie (8x8)	16.661	4.708	
Cookie (9x9)	14.535	7.713	
Cookie (10x10)	12.838	11.764	
Gambler ($p = 0.25$)	0.250	0.001	Sí. Los valores en s_0 coinciden o son muy cercanos a las probabilidades de éxito teóricas, lo cual indica que la política greedy es óptima o casi óptima.
Gambler ($p = 0.4$)	0.400	0.002	
Gambler ($p = 0.55$)	0.730	0.025	

Table 2: Evaluación de la política greedy inducida desde los valores de la política uniformemente aleatoria.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta h)

Los resultados de ejecutar los experimentos solicitados con la nueva política de *value iteration* son los siguientes:

Problema	**Valor óptimo $V^*(s_0)$**	**Tiempo (s)**
Grid (3x3)	-2.000	0.000
Grid (4x4)	-2.000	0.000
Grid (5x5)	-4.000	0.000
Grid (6x6)	-4.000	0.001
Grid (7x7)	-6.000	0.001
Grid (8x8)	-6.000	0.001
Grid (9x9)	-8.000	0.002
Grid (10x10)	-8.000	0.002
Cookie (3x3)	48.760	0.194
Cookie (4x4)	35.907	0.632
Cookie (5x5)	28.197	1.597
Cookie (6x6)	23.063	3.338
Cookie (7x7)	19.402	6.309
Cookie (8x8)	16.661	10.419
Cookie (9x9)	14.535	16.705
Cookie (10x10)	12.838	25.774
Gambler ($p = 0.25$)	0.250	0.025
Gambler ($p = 0.4$)	0.400	0.029
Gambler ($p = 0.55$)	1.000	2.604

Table 3: Resultados de Value Iteration con $\theta = 10^{-10}$ para obtener $V^*(s_0)$.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta i)

Como resultado del experimento, se observó que el número total de políticas óptimas posibles en el **GamblerProblem** puede ser extremadamente alto (del orden de 10^{25} en algunos casos). Esto se debe a que en varios estados no terminales existen múltiples acciones que entregan el mismo valor esperado al redondear los valores q(s,a) al quinto decimal, como exige el enunciado. La presencia de estas múltiples acciones óptimas por estado genera un crecimiento exponencial en la cantidad total de combinaciones posibles de políticas.

Debido a esto, por razones prácticas de visualización, en los gráficos presentados a continuación se muestra solo una muestra aleatoria representativa de políticas óptimas extraídas del conjunto completo.

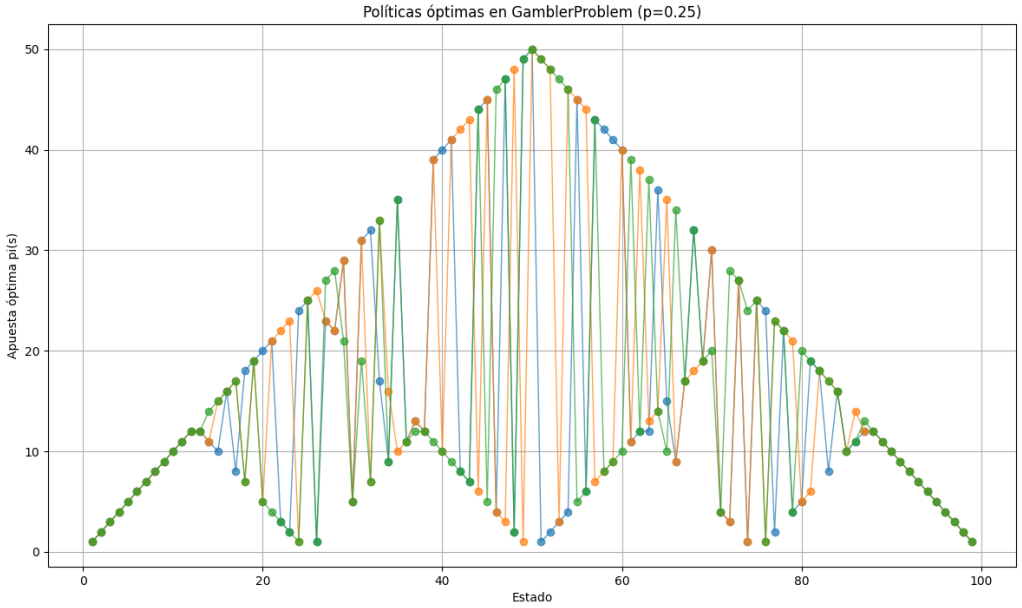

Figure 1: Tres políticas óptimas aleatorias con $p = 0.25$.

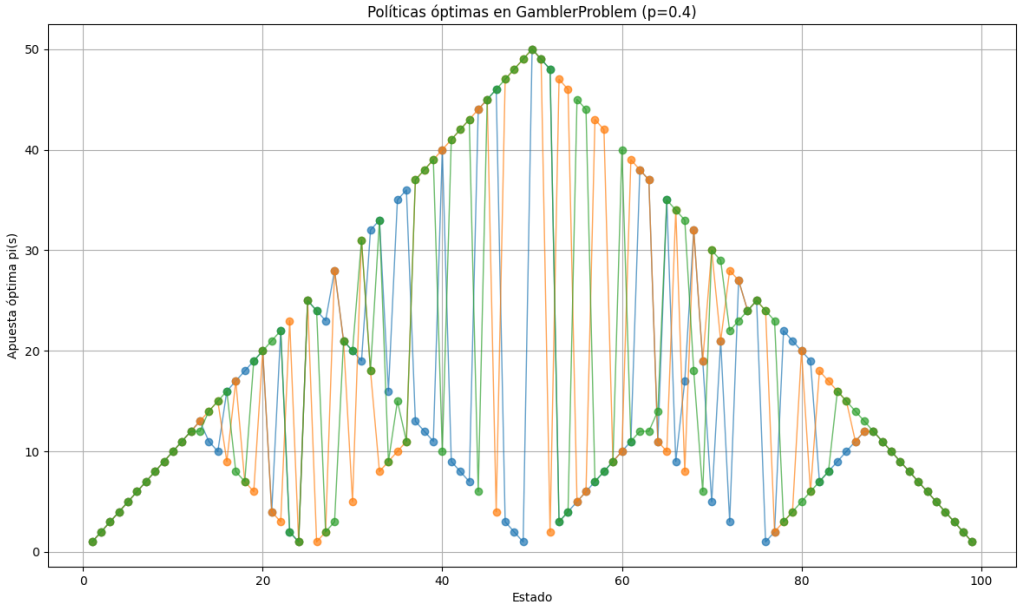

Figure 2: Tres políticas óptimas aleatorias con $p = 0.4$.

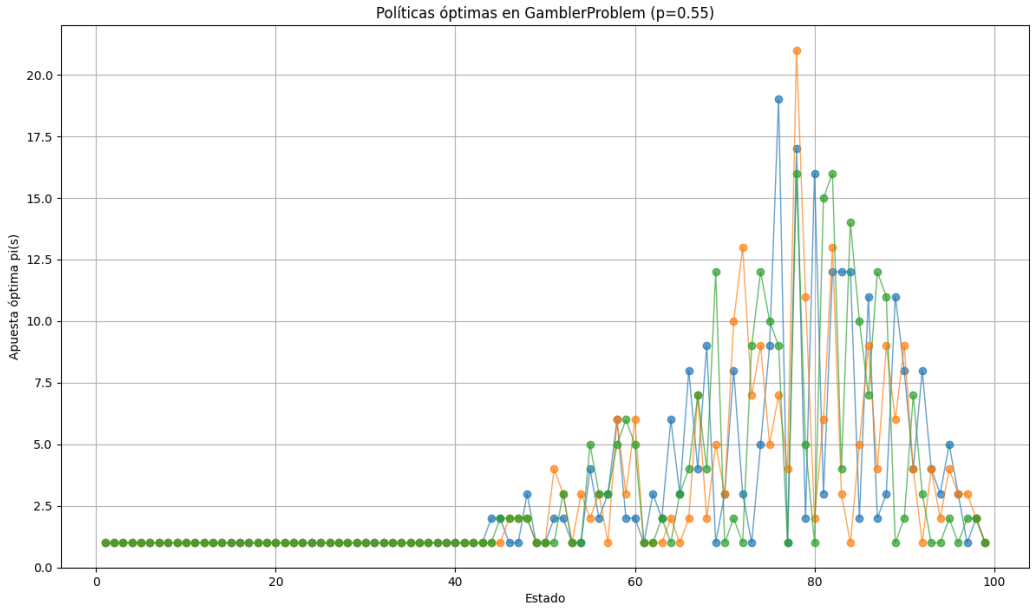

Figure 3: Tres políticas óptimas aleatorias con $p = 0.55$.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta j)

Los resultados de realizar los experimentos solicitados en el enunciado son los siguientes:

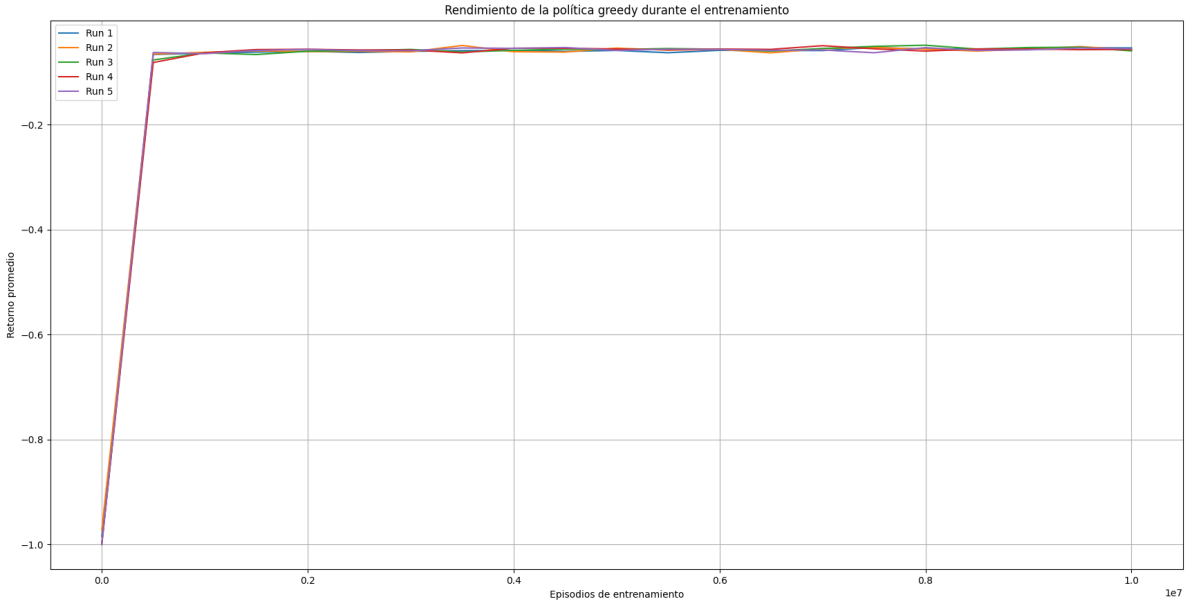

Figure 4: Rendimiento de la política *greedy* de **Blackjack** Monte Carlo ϵ-*soft control*.

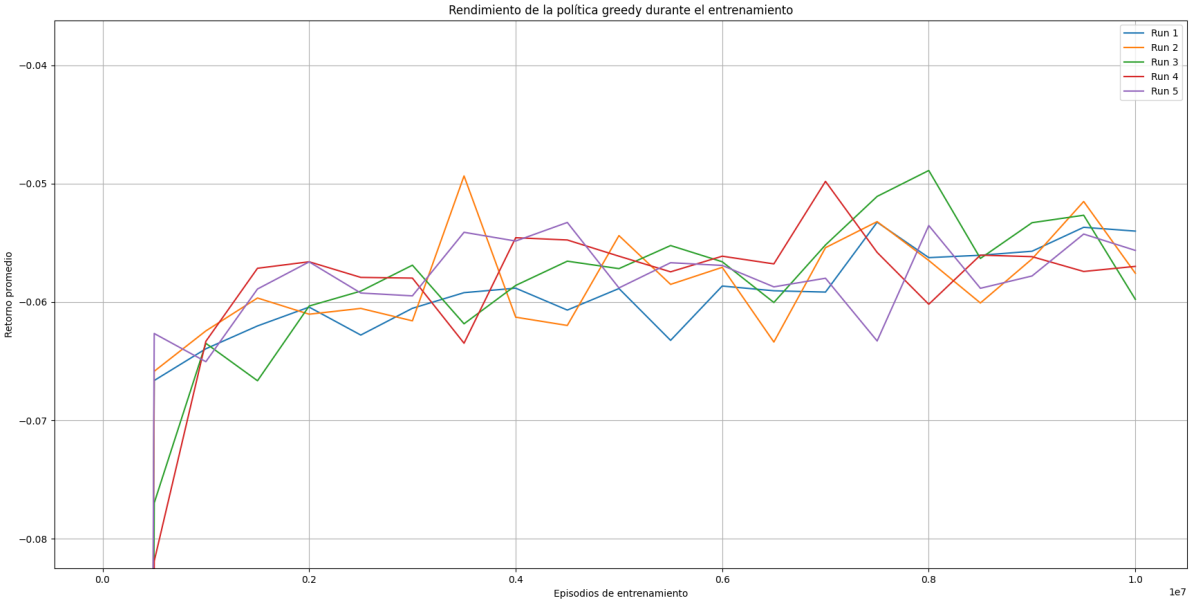

Figure 5: Rendimiento de la política *greedy* de **Blackjack** Monte Carlo ϵ-*soft control* vista con zoom.

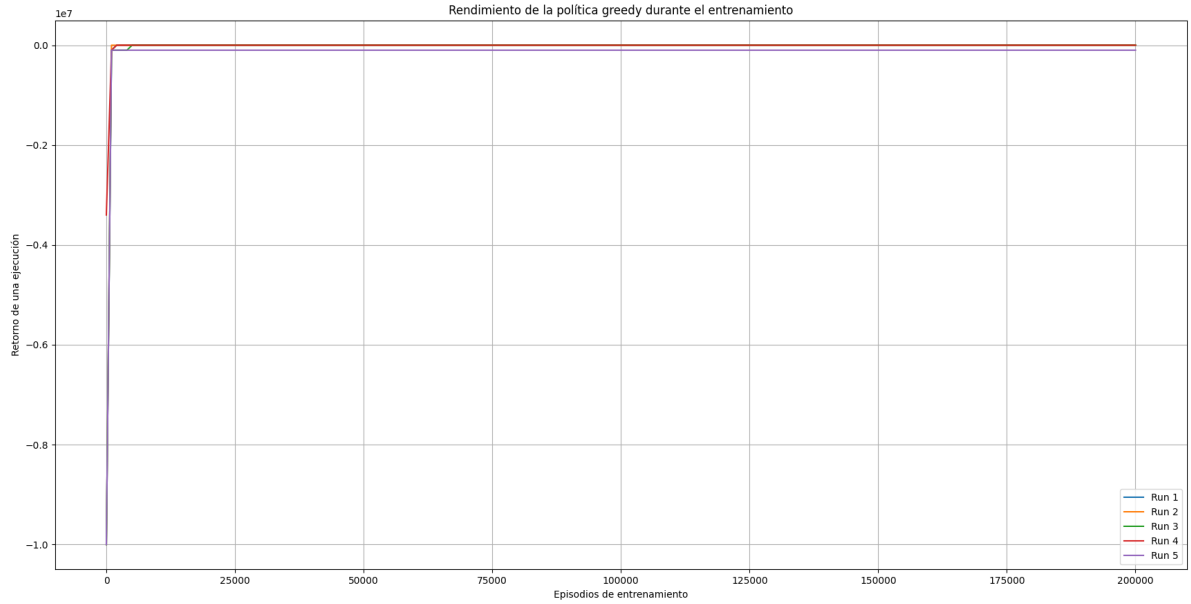

Figure 6: Rendimiento de la política *greedy* de **Cliff** Monte Carlo ϵ-*soft control*.

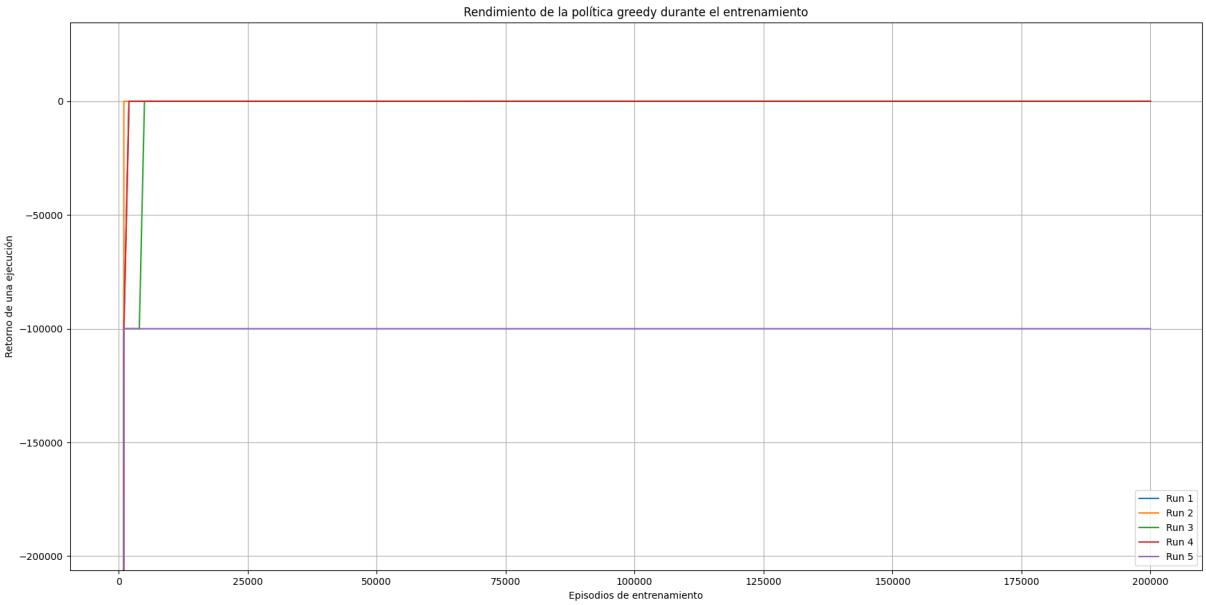

Figure 7: Rendimiento de la política *greedy* de **Cliff** Monte Carlo ϵ-*soft control* vista con zoom.

Además se presentan los siguientes resultados complementarios a los gráficos anteriores:

Blackjack

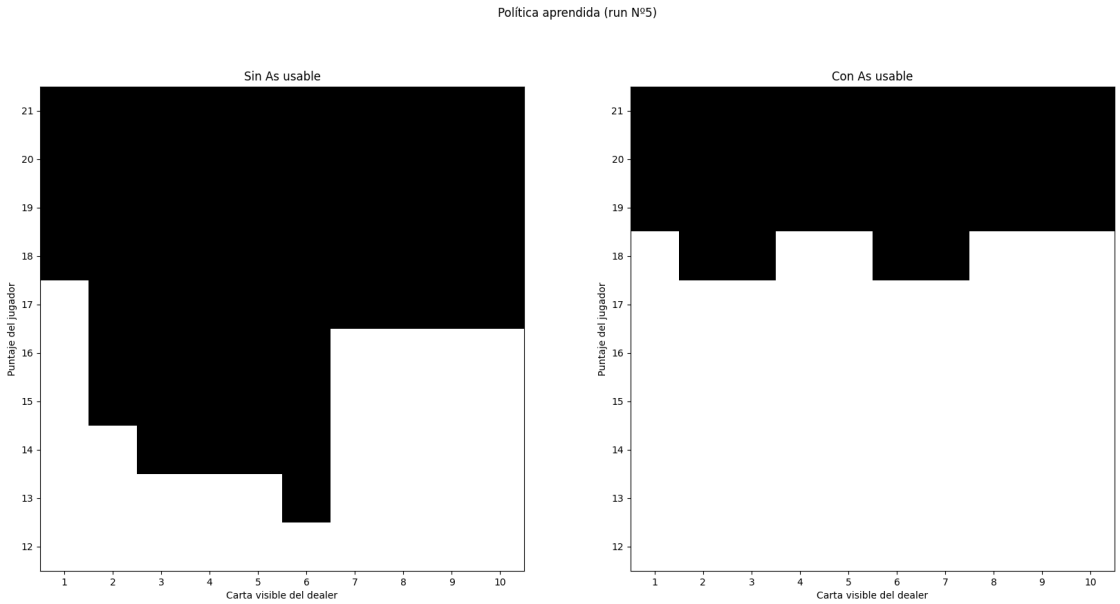

Figure 8: Ejemplo de política *greedy* de **Blackjack** Monte Carlo ϵ-*soft control*. Negro representa *'stick'*, blanco *'hit'*. Retorno aproximado de -0.055.

Cliff

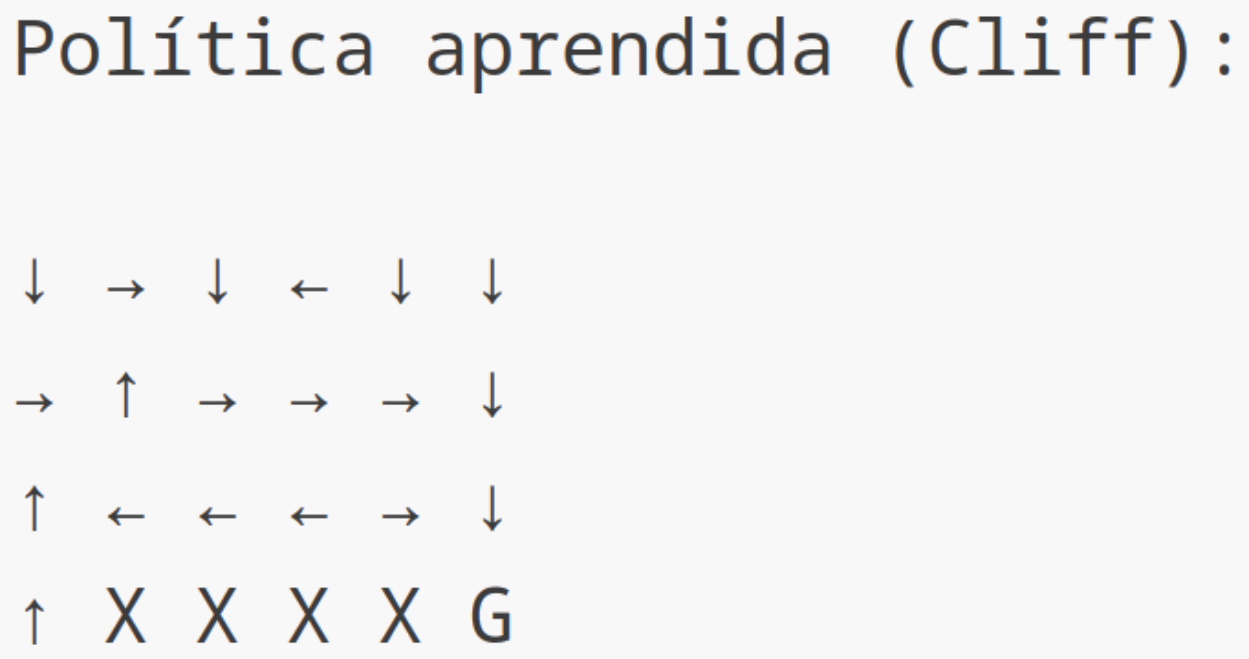

Figure 9: Ejemplo política *greedy* de **Cliff** Monte Carlo ϵ-*soft control*. Retorno de -11.

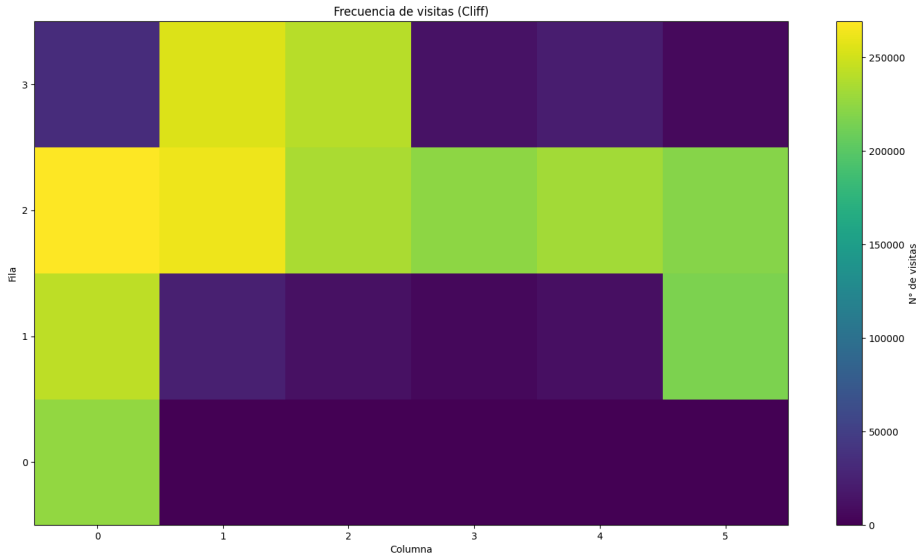

Figure 10: Estados más visitados por ejemplo de política *greedy* de **Cliff** Monte Carlo ϵ-*soft control*. Retorno de -11.

Finalmente se muestran las respectivas tablas con los resultados de los *tests* durante el entrenamiento:

Evaluación	**Corrida 1**	**Corrida 2**	**Corrida 3**	**Corrida 4**	**Corrida 5**
1	-0.98467	-0.97255	-0.99732	-1.00000	-1.00000
0.5M	-0.06663	-0.06585	-0.07697	-0.08188	-0.06266
1.0M	-0.06395	-0.06244	-0.06349	-0.06332	-0.06504
1.5M	-0.06202	-0.05966	-0.06665	-0.05715	-0.05890
2.0M	-0.06043	-0.06103	-0.06034	-0.05660	-0.05661
2.5M	-0.06279	-0.06054	-0.05908	-0.05792	-0.05924
3.0M	-0.06053	-0.06159	-0.05689	-0.05798	-0.05948
3.5M	-0.05921	-0.04935	-0.06184	-0.06348	-0.05411
4.0M	-0.05883	-0.06128	-0.05860	-0.05457	-0.05485
4.5M	-0.06068	-0.06198	-0.05655	-0.05476	-0.05328
5.0M	-0.05886	-0.05439	-0.05718	-0.05613	-0.05881
5.5M	-0.06324	-0.05851	-0.05523	-0.05745	-0.05668
6.0M	-0.05865	-0.05707	-0.05660	-0.05613	-0.05691
6.5M	-0.05905	-0.06338	-0.06004	-0.05678	-0.05872
7.0M	-0.05916	-0.05541	-0.05518	-0.04981	-0.05799
7.5M	-0.05326	-0.05321	-0.05108	-0.05581	-0.06329
8.0M	-0.05625	-0.05649	-0.04889	-0.06019	-0.05354
8.5M	-0.05605	-0.06008	-0.05632	-0.05604	-0.05884
9.0M	-0.05571	-0.05636	-0.05330	-0.05617	-0.05780
9.5M	-0.05369	-0.05151	-0.05267	-0.05742	-0.05426
10.0M	-0.05401	-0.05757	-0.05977	-0.05699	-0.05564

Table 4: Retorno promedio por evaluación para el entorno **Blackjack** usando MC Every-Visit *greedy*. Cada entrada corresponde al promedio de 0.1M de evaluaciones.

Evaluación	**Corrida 1**	**Corrida 2**	**Corrida 3**	**Corrida 4**	**Corrida 5**
1	-10000100	-10000100	-10000100	-3399968	-10000100
1k	-100001	-9	-100001	-100001	-100001
2k	-11	-9	-100001	-11	-100001
3k	-11	-9	-100001	-11	-100001
4k	-11	-9	-100001	-11	-100001
5k	-11	-9	-11	-11	-100001
6k	-11	-9	-11	-11	-100001
7k	-11	-9	-11	-11	-100001
8k	-11	-9	-11	-11	-100001
9k	-11	-9	-11	-11	-100001
10k	-11	-9	-11	-11	-100001
...	...	...	...	...	...
199k	-7	-9	-11	-11	-100001
200k	-7	-9	-11	-11	-100001

Table 5: Retorno por evaluación para el entorno **Cliff**. Cada evaluación corresponde a una única ejecución con política *greedy*.

Todas las columnas se quedan "pegadas" en el retorno que se observa desde 10 hasta 200 mil episodios, excepto por la Corrida 1, en la cual se pasó de -11 a -7 alrededor del episodio 80k.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta k)

Según los resultados obtenidos por Monte Carlo, la política aprendida indica que el jugador debería dejar de robar cartas (hacer *"stick"*) cuando tiene un puntaje de entre 18 y 19, aunque al tener un As el agente tiende a arriesgar más. Esta decisión es razonable, ya que las probabilidades de pasarse al pedir otra carta son muy altas, y el valor esperado de seguir actuando se vuelve negativo.
Por otro lado, se observa que para ciertas combinaciones "centrales" en la matriz, el agente tiende a hacer *"stick"* para valores más bajos de sus cartas. Se cree que esto puede deberse a que la probabilidad de que el *dealer* se pase cuando su carta inicial es entre 2 y 6 es más alta y no compensa arriesgarse.

Si bien hay pequeñas variaciones entre las cinco corridas debido a la aleatoriedad en la exploración, todas llegan a conclusiones similares en los estados clave, como aquellos con puntajes altos del jugador. Esto se debe a que en el Blackjack, los comportamientos son relativamente estables y fáciles de estimar cuando se dispone de suficiente exploración. Como consecuencia, Monte Carlo tiende a converger de forma consistente hacia políticas parecidas en este entorno.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta l)

En el entorno Cliff, el curso de acción más común en Monte Carlo es avanzar hacia arriba desde la posición inicial y luego bordear el precipicio por la parte superior, evitando las casillas peligrosas hasta llegar a la meta. Esta política permite alcanzar el objetivo con un costo de -9 o -11, evitando las penalizaciones severas por caer al cliff.
Está política no es óptima si el agente fuera completamente *greedy*, ya que en este caso conviene pasar pegado al cliff, ya que es una ruta más corta. Sin embargo, como se tenía $\epsilon = 0.1$, la posibilidad de caer es un riesgo que muchos agentes preferían no correr.

Este resultado no es completamente consistente entre las cinco corridas. En algunas ejecuciones el agente efectivamente descubre esta trayectoria "sub-óptima" pero segura, mientras que en otras se queda con rutas más cortas pero inseguras, llegando a retornos de -7. Esto ocurre porque Monte Carlo explora rutas aleatoriamente: si en las primeras etapas el agente cae muchas veces por el precipicio, tenderá a reforzar rutas conservadoras que, si bien evitan la penalización, también se alejan de la solución óptima. El aprendizaje en Cliff depende fuertemente del azar inicial, lo que afecta la consistencia del resultado.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta m)

Monte Carlo no puede considerarse un algoritmo estable en todos los contextos. En el entorno de **Blackjack**, mostró un comportamiento bastante consistente entre las distintas corridas, convergiendo en todas ellas hacia políticas similares con retornos estables y (se presume) cercanos al óptimo. Esto refleja cierta estabilidad, ya que el entorno es relativamente acotado y las trayectorias exitosas son exploradas con suficiente frecuencia.

En cambio, en **Cliff**, Monte Carlo fue mucho más sensible a la aleatoriedad inicial. Algunas corridas encontraron la política "óptima" (retorno -7), mientras que otras terminaron en políticas subóptimas con retornos de -9 y -11, pero más seguras. En algunos casos incluso se quedó atascado en *loops* que nunca llegaban al objetivo. Esta inestabilidad se debe a la dependencia del algoritmo en la exploración aleatoria: si no se descubre una buena trayectoria temprano, es difícil salir de caminos conservadores pero ineficientes. Por lo tanto, aunque puede aprender políticas óptimas, Monte Carlo no es estable en entornos con riesgos altos o trayectorias peligrosas, como **Cliff**, donde pequeñas diferencias en la exploración pueden llevar a políticas muy distintas.

NOMBRE: Bruno Farfán

SECCIÓN: 1

PUNTAJE:

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3675 — Aprendizaje Reforzado — 1' 2025

Tarea 2 – Respuesta Pregunta n)

Monte Carlo sí puede resolver el problema de Cliff cuando el largo de la grilla es 12, pero con mayor dificultad y menor probabilidad de converger a la política óptima. A medida que se incrementa el largo de la grilla, también aumenta el número de pasos necesarios para alcanzar la meta y la cantidad de trayectorias posibles, lo que hace más difícil que el agente explore eficazmente las rutas óptimas pegadas al precipicio.

Dado que Monte Carlo depende exclusivamente de la experiencia generada por episodios completos y no realiza generalización entre estados, necesita mucho más tiempo y suerte para encontrar y reforzar las trayectorias óptimas en un entorno más grande. Si en las primeras fases del entrenamiento el agente cae repetidamente al precipicio (por la exploración $\epsilon - greedy$), es muy probable que aprenda a evitar esas zonas, reforzando rutas largas y seguras. Incluso se puede quedar "pegado" en *loops* que lo mantienen lejos del *"cliff"*, pero que nunca llegan al objetivo.
Por eso, aunque en teoría Monte Carlo puede resolver este entorno más grande, en la práctica el aprendizaje se vuelve más lento, más inestable y menos eficiente.