Tarea 2: MDPs y Monte Carlo

Pontificia Universidad Católica de Chile

Integrante 1: Clemente Acevedo Integrante 2: Jorge Molina

9 de septiembre de 2025

Índice

1. Introducción

Breve resumen del objetivo de la tarea: resolver problemas de MDPs con programación dinámica (policy evaluation, value iteration) y Monte Carlo. Señalar estructura del informe y organización de resultados.

2. Preguntas Teóricas

2.1. (a) Demostración de $v_\pi(s)$

Objetivo. Probar la relación fundamental entre el valor de estado y el valor de acción:

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

*Intuición: Esta demostración conecta el valor de **estar** en un estado ($v_\pi(s)$) con el valor de **tomar una acción** desde ese estado ($q_\pi(s,a)$). La idea central es que el valor de un estado es simplemente el promedio de los valores de todas las acciones que podemos tomar, ponderado por la probabilidad de elegirlas según nuestra política π. Es una relación en el mismo instante de tiempo t.*

Definicione que usaremos.

- Valor de estado: $v_\pi(s) := \mathbb{E}_\pi\big[G_t \mid S_t = s\big]$.
- Valor estado-acción: $q_\pi(s,a) := \mathbb{E}_\pi\big[G_t \mid S_t = s,\ A_t = a\big]$.
- Política estocástica: $\pi(a \mid s) := \Pr\big(A_t = a \mid S_t = s\big)$.

Prueba paso a paso (caso discreto).

1. Comenzamos con la definición de valor para un estado s arbitrario:

$$v_\pi(s) = \mathbb{E}_\pi\big[G_t \mid S_t = s\big].$$

2. Expandimos la esperanza usando su definición fundamental para el caso discreto. La esperanza es una suma de cada posible valor (g) ponderado por su probabilidad:

$$\mathbb{E}_\pi\big[G_t \mid S_t = s\big] = \sum_g g\ \Pr_\pi\big(G_t = g \mid S_t = s\big).$$

3. Aplicamos la **Ley de Probabilidad Total** para descomponer la probabilidad, condicionando sobre todas las acciones posibles $a \in \mathcal{A}(s)$, que forman una partición del espacio de eventos:

$$\Pr_\pi\big(G_t = g \mid S_t = s\big) = \sum_{a \in \mathcal{A}(s)} \Pr_\pi\big(G_t = g \mid S_t = s, A_t = a\big)\ \Pr_\pi\big(A_t = a \mid S_t = s\big).$$

4. Sustituimos esta descomposición en la ecuación del paso 2 y reordenamos las sumatorias para agrupar los términos de manera conveniente:

$$\mathbb{E}_\pi\big[G_t \mid S_t = s\big] = \sum_{a\in\mathcal{A}(s)} \left[\sum_g g \; \Pr_\pi\big(G_t = g \mid S_t = s, A_t = a\big)\right] \Pr_\pi\big(A_t = a \mid S_t = s\big).$$

5. Ahora, reconocemos dos términos clave dentro de la expresión anterior por su definición:

   - La suma interna es precisamente la definición de la esperanza condicional de G_t dado $S_t = s$ y $A_t = a$, que es $q_\pi(s, a)$.

$$\sum_g g \; \Pr_\pi\big(G_t = g \mid S_t = s, A_t = a\big) = \mathbb{E}_\pi\big[G_t \mid S_t = s, A_t = a\big] = q_\pi(s, a).$$

   - El término de probabilidad es la definición de la política $\pi(a \mid s)$.

$$\Pr_\pi\big(A_t = a \mid S_t = s\big) = \pi(a \mid s).$$

6. Al juntar todo, llegamos a la expresión final que queríamos demostrar:

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)} \pi(a \mid s)\, q_\pi(s, a). \quad \blacksquare$$

2.2. (b) Demostración de $q_\pi(s, a)$

Objetivo. Probar la ecuación de Bellman para el valor de acción $q_\pi(s, a)$:

$$q_\pi(s, a) = \sum_{s'\in\mathcal{S}} \sum_{r\in\mathbb{R}} p(s', r \mid s, a)\,\big(r + \gamma\, v_\pi(s')\big).$$

*Intuición: Aquí derivaremos la famosa **ecuación de Bellman** para q_π. La herramienta clave será la definición recursiva del retorno, $G_t = R_{t+1} + \gamma G_{t+1}$. Esta fórmula nos permite expandir el valor de una acción y mirar un paso hacia el futuro: el valor de tomar la acción (s, a) hoy es igual a la recompensa que esperamos recibir mañana, más el valor descontado del estado al que lleguemos.*

Definiciones que usaremos.

- $q_\pi(s, a) := \mathbb{E}_\pi\big[G_t \mid S_t = s,\ A_t = a\big]$.
- $v_\pi(s') := \mathbb{E}_\pi\big[G_{t+1} \mid S_{t+1} = s'\big]$.
- Dinámica del entorno: $p(s', r \mid s, a) := \Pr\big(S_{t+1} = s', R_{t+1} = r \mid S_t = s,\ A_t = a\big)$.
- Definición recursiva del retorno: $G_t = R_{t+1} + \gamma\, G_{t+1}$.

Prueba paso a paso (caso discreto).

1. Partimos de la definición de valor-acción para un par (s, a) fijo:

$$q_\pi(s, a) = \mathbb{E}_\pi\big[G_t \mid S_t = s, A_t = a\big].$$

2. Sustituimos la definición recursiva del retorno y aplicamos la **linealidad de la esperanza**:

$$q_\pi(s,a) = \mathbb{E}_\pi\big[R_{t+1} \mid S_t = s, A_t = a\big] + \gamma\, \mathbb{E}_\pi\big[G_{t+1} \mid S_t = s, A_t = a\big].$$

3. Descomponemos cada una de las dos esperanzas usando la **Ley de la Esperanza Total**, condicionando sobre todos los posibles resultados del siguiente paso, es decir, el par (s', r):

$$\mathbb{E}_\pi\big[R_{t+1} \mid s, a\big] = \sum_{s',r} \mathbb{E}_\pi\big[R_{t+1} \mid s, a, s', r\big]\; p(s', r \mid s, a),$$
$$\mathbb{E}_\pi\big[G_{t+1} \mid s, a\big] = \sum_{s',r} \mathbb{E}_\pi\big[G_{t+1} \mid s, a, s', r\big]\; p(s', r \mid s, a).$$

4. Evaluamos las esperanzas condicionales internas, que ahora son más fáciles:

   a) La esperanza de la variable aleatoria R_{t+1} condicionada a que toma un valor específico r es simplemente ese valor.

$$\mathbb{E}_\pi\big[R_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r\big] = r.$$

   b) Por la **propiedad de Markov**, el retorno futuro G_{t+1} solo depende del estado futuro $S_{t+1} = s'$. El pasado $(S_t = s, A_t = a)$ no aporta información adicional una vez que conocemos s'.

$$\mathbb{E}_\pi\big[G_{t+1} \mid S_t = s, A_t = a, S_{t+1} = s'\big] = \mathbb{E}_\pi\big[G_{t+1} \mid S_{t+1} = s'\big] = v_\pi(s').$$

5. Finalmente, sustituimos estas esperanzas simplificadas (del paso 4) en las ecuaciones del paso 3 y reagrupamos los términos:

$$\begin{aligned} q_\pi(s,a) &= \sum_{s',r} r\, p(s', r \mid s, a) + \gamma \sum_{s',r} v_\pi(s')\, p(s', r \mid s, a) \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathbb{R}} p(s', r \mid s, a)\, \big(r + \gamma\, v_\pi(s')\big). \quad \blacksquare \end{aligned}$$

2.3. (c) MDP determinista: comparación de políticas

Planteamiento. Vamos a analizar un MDP determinista con tres estados: $\mathcal{S} = \{s_0, s_l, s_r\}$. Estando en s_0, tenemos dos opciones o políticas: ir a la izquierda (π_l) o ir a la derecha (π_r).

$$\pi_l(s_0) = a_l, \qquad \pi_r(s_0) = a_r.$$

La idea es identificar qué política es la mejor para distintos factores de descuento: $\gamma \in \{0, 0{,}5, 0{,}9\}$.

Método. Para evaluar cada política, vamos a usar la ecuación de Bellman. Primero, veremos cómo se simplifica para este caso, que es totalmente determinista.

Derivación de la Ecuación Simplificada: La ecuación de Bellman completa para $v_\pi(s)$ se ve así:

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

Pero como aquí todo es determinista, la cosa se pone más fácil:

1. **La política es determinista:** La primera suma (sobre las acciones) se va, porque la política π siempre elige una única acción $\pi(s)$ con probabilidad 1. Así que nos quedamos solo con el término de esa acción.
$$\sum_{a\in\mathcal{A}(s)} \pi(a|s)(\dots) \quad \longrightarrow \quad 1 \cdot \sum_{s',r} p(s',r|s,\pi(s))\big[r+\gamma v_\pi(s')\big].$$
2. **La dinámica es determinista:** La segunda suma (sobre estados y recompensas) también se va. Al tomar una acción, siempre llegamos al mismo estado s' y recibimos la misma recompensa r. La probabilidad de ese único resultado es 1.
$$\sum_{s',r} p(s',r|s,a)(\dots) \quad \longrightarrow \quad 1 \cdot \big[r(s,a)+\gamma v_\pi(s')\big].$$

Juntando todo, la ecuación de Bellman para nuestro problema queda súper simple:

$$v_\pi(s) = r(s,\pi(s)) + \gamma\, v_\pi(s').$$

Ahora, solo tenemos que resolver el sistema de ecuaciones que sale de esta fórmula.

Caso 1: Política "Izquierda" (π_l)

- En s_0, la acción a_l nos manda a s_l y nos da una recompensa de 1:
$$v_{\pi_l}(s_0) = 1 + \gamma\, v_{\pi_l}(s_l).$$
- Desde s_l, la única jugada posible nos regresa a s_0 con recompensa 0:
$$v_{\pi_l}(s_l) = 0 + \gamma\, v_{\pi_l}(s_0).$$
- Reemplazamos la segunda ecuación en la primera:
$$v_{\pi_l}(s_0) = 1 + \gamma(\gamma\, v_{\pi_l}(s_0)) = 1 + \gamma^2 v_{\pi_l}(s_0).$$
- Y ahora, a despejar $v_{\pi_l}(s_0)$:
$$v_{\pi_l}(s_0)(1-\gamma^2) = 1 \quad \Rightarrow \quad v_{\pi_l}(s_0) = \frac{1}{1-\gamma^2}.$$

Caso 2: Política "Derecha" (π_r)

- En s_0, la acción a_r nos manda a s_r con recompensa 0:
$$v_{\pi_r}(s_0) = 0 + \gamma\, v_{\pi_r}(s_r).$$
- Desde s_r, la única jugada nos regresa a s_0, pero con recompensa 2:
$$v_{\pi_r}(s_r) = 2 + \gamma\, v_{\pi_r}(s_0).$$
- De nuevo, reemplazamos la segunda en la primera:
$$v_{\pi_r}(s_0) = \gamma(2 + \gamma v_{\pi_r}(s_0)) = 2\gamma + \gamma^2 v_{\pi_r}(s_0).$$
- Y despejamos $v_{\pi_r}(s_0)$:
$$v_{\pi_r}(s_0)(1-\gamma^2) = 2\gamma \quad \Rightarrow \quad v_{\pi_r}(s_0) = \frac{2\gamma}{1-\gamma^2}.$$

Comparación y Conclusión

Ahora veamos qué política gana para cada valor de γ:

- Para $\boldsymbol{\gamma = 0}$ (si solo importa el ahora):

$$v_{\pi_l}(s_0) = 1, \quad v_{\pi_r}(s_0) = 0 \quad \Rightarrow \quad \pi_l \text{ es la mejor.}$$

- Para $\boldsymbol{\gamma = 0{,}5}$ (cuando el presente y el futuro importan por igual):

$$v_{\pi_l}(s_0) = \frac{1}{1-0{,}25} \approx 1{,}33, \quad v_{\pi_r}(s_0) = \frac{2(0{,}5)}{1-0{,}25} \approx 1{,}33 \quad \Rightarrow \quad \text{Da lo mismo, ambas son óptimas.}$$

- Para $\boldsymbol{\gamma = 0{,}9}$ (si somos pacientes y el futuro vale mucho):

$$v_{\pi_l}(s_0) = \frac{1}{1-0{,}81} \approx 5{,}26, \quad v_{\pi_r}(s_0) = \frac{2(0{,}9)}{1-0{,}81} \approx 9{,}47 \quad \Rightarrow \quad \pi_r \text{ es la mejor, lejos.}$$

En resumen: Como vemos, la política óptima depende totalmente del factor de descuento γ. Un agente "apurado" (gamma bajo) prefiere la recompensa inmediata de π_l. En cambio, un agente "paciente" (gamma alto) prefiere irse por el camino de π_r, que aunque no da nada al principio, entrega una recompensa más grande después.

3. Experimentos: Programación Dinámica

3.1. (d) ¿Qué valor tiene una política totalmente aleatoria?

Primero, implementamos *Iterative Policy Evaluation*. El objetivo era simple: calcular la función de valor $V(s)$ para una política donde siempre elegimos cualquier acción disponible con la misma probabilidad. Básicamente, un agente que actúa al azar.

Para que los resultados tengan sentido, es clave saber desde dónde partimos en cada problema:

- **GridProblem**: El agente empieza justo en el centro de la grilla.
- **CookieProblem**: El agente inicia en la esquina superior izquierda y la galleta en la inferior derecha.
- **GamblerProblem**: El jugador comienza con un capital de \$50.

A continuación, los resultados que obtuvimos, mostrando el valor del estado inicial $V(s_0)$ y cuánto se demoró el computador en calcularlo.

Cuadro 1: Resultados de Policy Evaluation para **GridProblem** ($\gamma = 1{,}0$).

Tamaño Grilla	$V(s_0)$	**Tiempo (s)**
3x3	-8.000	0.005
4x4	-18.000	0.015
5x5	-37.333	0.045
6x6	-60.231	0.114
7x7	-93.496	0.227
8x8	-131.193	0.440
9x9	-180.001	0.707
10x10	-233.879	1.129

Cuadro 2: Resultados de Policy Evaluation para **CookieProblem** ($\gamma = 0{,}99$).

Tamaño Grilla	$V(s_0)$	**Tiempo (s)**
3x3	5.380	0.491
4x4	2.477	1.490
5x5	1.291	2.607
6x6	0.729	5.061
7x7	0.437	9.195
8x8	0.273	15.247
9x9	0.177	23.749
10x10	0.118	35.763

Cuadro 3: Resultados de Policy Evaluation para **GamblerProblem** ($\gamma = 1{,}0$).

Prob. Cara (p_h)	$V(s_0)$	**Tiempo (s)**
0.25	0.067	0.121
0.40	0.284	0.180
0.55	0.612	0.242

3.2. (e) ¿Por qué algunos problemas tardan mucho más que otros?

Viendo las tablas, salta a la vista que el **CookieProblem es, por lejos, el más lento y exigente**.

Esto es por el tamaño de su **espacio de estados** ($|\mathcal{S}|$). Pensemos cómo crece en cada caso para una grilla $N \times N$:

- **GridProblem**: El estado es solo la posición del agente, así que $|\mathcal{S}| = N^2$.
- **CookieProblem**: El estado necesita saber la posición del agente **y** la de la galleta. Esto aumenta el tamaño a $|\mathcal{S}| = (N \times N) \times (N \times N) = N^4$.

Ese crecimiento es significativo. Para una grilla de 10x10, GridProblem maneja 100 estados, pero CookieProblem debe manejar 10,000. Como el algoritmo tiene que "recorrer" todos los estados en cada iteración, es obvio que se demore muchísimo más.

Aunque el tamaño del espacio de estados es el factor dominante, el costo por iteración también depende de otros detalles, como cuántas acciones hay por estado ($|\mathcal{A}(s)|$) y la complejidad de las transiciones. Pero la explosión combinatoria de N^4 es la razón principal.

3.3. (f) ¿Afecta el factor de descuento (γ) al tiempo?

La respuesta es un claro **sí, un factor de descuento γ más bajo acelera la convergencia**.

Esto se puede entender con una analogía. Un γ alto (cercano a 1) hace que el agente sea un "visionario", le importan mucho las recompensas futuras. Para calcular el valor de un estado, necesita que la información de recompensas muy lejanas se propague hacia atrás, lo que requiere muchas iteraciones.

En cambio, un γ bajo (cercano a 0) convierte al agente en un "miope", solo le importa el aquí y el ahora. El valor de un estado depende casi exclusivamente de sus recompensas inmediatas. Esto hace que los valores se estabilicen muy rápido, necesitando menos "pasadas" y acelerando todo el proceso.

3.4. (h) Encontrando la política óptima con Value Iteration

Ahora, usamos *Value Iteration* para encontrar la función de valor óptima $V^*(s)$, que nos dice lo mejor que se puede lograr desde cada estado.

Una observación interesante aparece en **GamblerProblem con** $p_h = 0{,}55$. El valor óptimo desde \$50, $V^*(s_0)$, es 1.000. Esto tiene mucho sentido, ya que con probabilidades a favor ($p_h > 0{,}5$), la probabilidad teórica de ganar (llegar a \$100) es altísima (cercana a 0.99996), y V^* en este problema representa justamente esa probabilidad de éxito.

Este valor teórico proviene de la fórmula problema de la *Ruina del Apostador*. Para un juego con probabilidad de ganar $p \neq 0{,}5$, la probabilidad $P(i)$ de alcanzar una fortuna de N antes de la bancarrota, comenzando con un capital i, se calcula así:

$$P(i) = \frac{1 - (q/p)^i}{1 - (q/p)^N}$$

Donde $p = 0{,}55$ es la probabilidad de ganar, $q = 1 - p = 0{,}45$, nuestro capital inicial es $i = 50$ y la meta es $N = 100$. Al reemplazar estos valores en la fórmula, obtenemos ese $\approx 0{,}99996$, lo que confirma que nuestro resultado de Value Iteration es correcto. Por si acaso, esta materia la vimos en Modelos Estocásticos el semestre pasado.

Cuadro 4: Resultados de Value Iteration para **GridProblem** ($\gamma = 1{,}0$).

Tamaño Grilla	$V^*(s_0)$	**Tiempo (s)**
3x3	-2.000	0.000
4x4	-2.000	0.000
5x5	-4.000	0.000
6x6	-4.000	0.001
7x7	-6.000	0.001
8x8	-6.000	0.002
9x9	-8.000	0.002
10x10	-8.000	0.003

Cuadro 5: Resultados de Value Iteration para **CookieProblem** ($\gamma = 0{,}99$).

Tamaño Grilla	$V^*(s_0)$	**Tiempo (s)**
3x3	48.760	0.297
4x4	35.907	0.994
5x5	28.197	2.480
6x6	23.063	5.259
7x7	19.402	9.696
8x8	16.661	16.345
9x9	14.535	26.782
10x10	12.838	41.617

Cuadro 6: Resultados de Value Iteration para **GamblerProblem** ($\gamma = 1{,}0$).

Prob. Cara (p_h)	$V^*(s_0)$	**Tiempo (s)**
0.25	0.250	0.038
0.40	0.400	0.046
0.55	1.000	4.357

3.5. (g) ¿Es Óptima la Política Derivada de una Evaluación Aleatoria?

Partiendo de la función de valor $V(s)$ de la política aleatoria (calculada en la sección 3.1), aplicamos un paso de **mejora de política** para obtener una nueva política determinista y *greedy*. Esta nueva política simplemente elige la acción que parece mejor según los valores iniciales. La pregunta clave es ¿este único paso de mejora es suficiente para alcanzar la optimalidad?

Para verificarlo rigurosamente, no basta con comparar el valor del estado inicial. Una política es óptima solo si elige una acción óptima en **todos los estados**. Por ello, comparamos la acción de nuestra política *greedy* para cada estado contra el conjunto de acciones óptimas verdaderas, las cuales derivamos de la función de valor óptima $V^*(s)$ encontrada con *Value Iteration*.

Los resultados fueron los siguientes:

- Para **GridProblem** y **CookieProblem**, de forma algo inesperada, la política *greedy* resultó ser la **óptima** en todos los casos. Esto sugiere que, aunque la evaluación de la política aleatoria produce valores imprecisos, estos fueron suficientes para ordenar correctamente las preferencias de acciones en todos los estados.
- Para **GamblerProblem con** $p_h = 0{,}55$, la historia fue diferente: **la política no fue óptima**.

Este experimento ilustra perfectamente el **Teorema de Mejora de Política** que vimos en clases, actuar de forma *greedy* sobre la función de valor de una política garantiza una política nueva que es igual o mejor, pero no necesariamente la óptima. En el caso de Gambler, la política mejoró, pero se necesitarían más ciclos de evaluación y mejora (es decir, el algoritmo completo de *Policy Iteration*) para converger a la verdadera estrategia óptima.

3.6. (i)Análisis de las Políticas Óptimas en Gambler's Problem

Finalmente, extrajimos las políticas óptimas completas para el Gambler's Problem. Para lograrlo, seguimos estrictamente la metodología indicada en el enunciado para manejar los errores de aproximación.

La Metodología: Redondeo al Quinto Decimal

El problema fundamental es que, tras muchos cálculos, dos acciones que teóricamente podrían ser igual de buenas pueden terminar con valores Q muy similares pero no idénticos, como '0.950000001' y '0.949999998'. Un 'argmax' ingenuo elegiría solo la primera y descartaría incorrectamente la segunda.

Para solucionar esto, el enunciado nos pide seguir un proceso de redondeo:

1. Para un estado s dado, calcular el valor $Q(s, a)$ para cada acción posible a.
2. **Redondear cada uno de estos valores Q** a 5 decimales.
3. Encontrar el valor máximo entre esta nueva lista de valores ya redondeados.
4. Considerar como óptimas a todas las acciones cuyo valor redondeado sea igual a este máximo.

De esta forma, si 'q(s, a1)' redondeado es igual a 'q(s, a2)' redondeado, y ambos son el máximo, ambas acciones se incluyen en el conjunto de políticas óptimas, tal como se espera. Este procedimiento nos asegura capturar todas las variantes de la estrategia óptima.

Análisis de las Políticas Resultantes

Con esta metodología, el análisis de las políticas revela que la estrategia óptima no es única, sino que depende drásticamente de si las probabilidades están a favor o en contra del jugador.

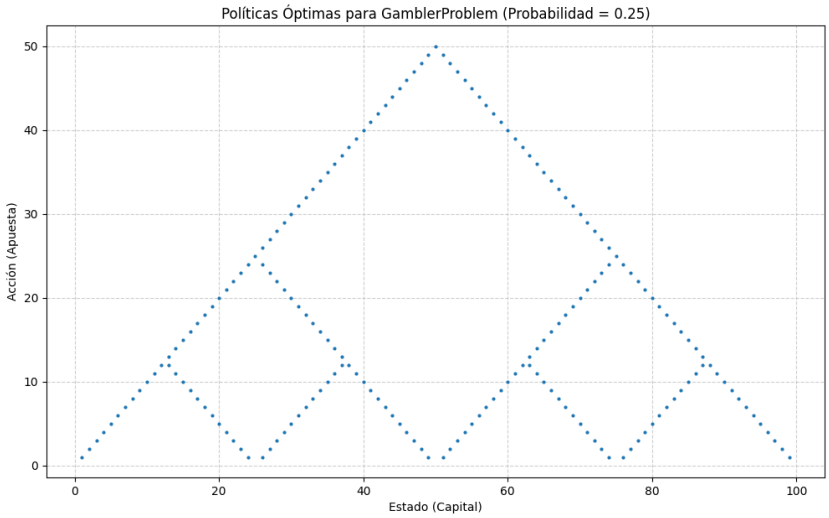


Figura 1: Política óptima para $p_h = 0{,}25$. Una estrategia de "juego audaz".

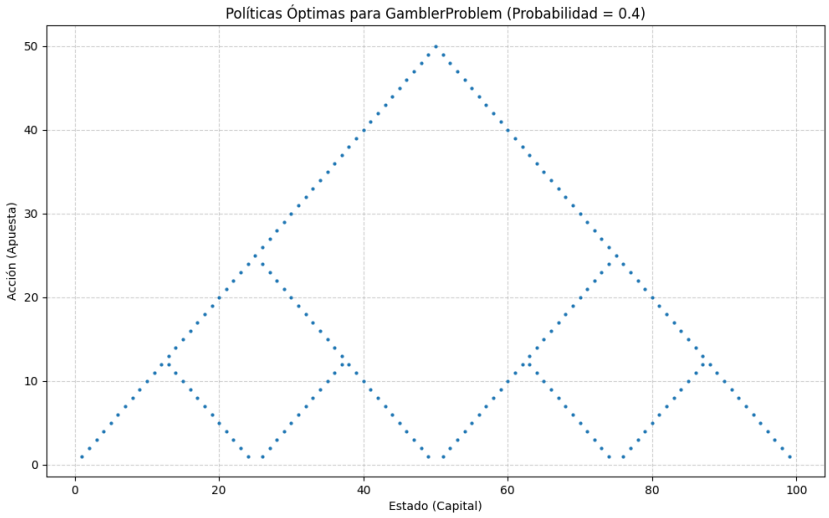


Figura 2: Política óptima para $p_h = 0{,}4$. Una estrategia de "juego audaz".

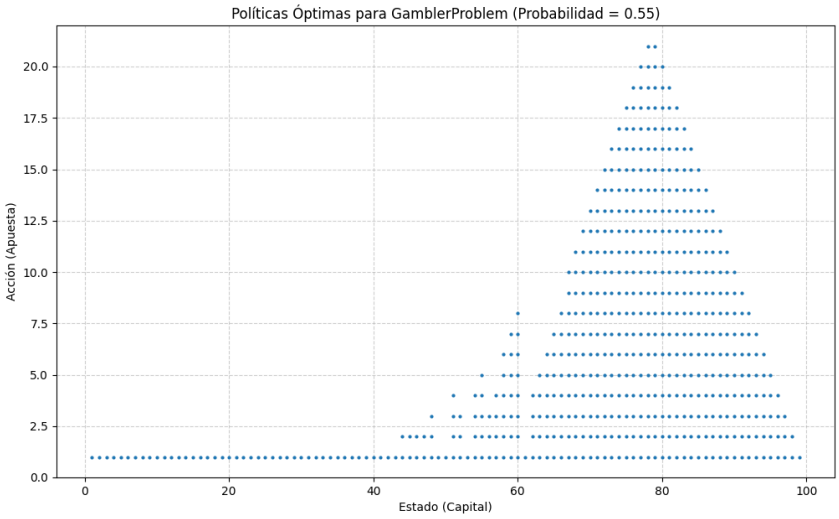


Figura 3: Política óptima para $p_h = 0{,}55$. Una estrategia principalmente conservadora.

El análisis de las políticas nos revela:

- **Con probabilidades en contra ($p_h \leq 0{,}4$):** La estrategia óptima es el **"juego audaz"**. El jugador hace apuestas grandes y arriesgadas, a menudo apostando todo lo que tiene (*all-in*) o lo justo para llegar a un punto seguro. La lógica es si las probabilidades están en nuestra contra, cada apuesta es una invitación a la ruina. La mejor opción es intentar ganar rápido con un golpe de suerte, minimizando el número de veces que nos exponemos a esas malas probabilidades.
- **Con probabilidades a favor ($p_h = 0{,}55$):** Aquí la estrategia es mucho más conservadora y métodica. La política óptima, sobre todo con poco capital, es **apostar lo mínimo posible ($1)**. Porque el tiempo y las probabilidades juegan a nuestro favor. El mayor riesgo es la bancarrota. Al hacer apuestas pequeñas, minimizamos el riesgo de perderlo todo y dejamos que la ganancia esperada positiva se acumule lentamente pero de forma segura. Es la estrategia de "lento pero seguro".

4. Experimentos: Monte Carlo

4.1. (j) On-policy Every-visit MC Control

Para resolver los dominios de Monte Carlo, se implementó el algoritmo de control *On-policy first-visit MC control*, pero con las dos modificaciones que se pedían en el enunciado.

Primero, en lugar de utilizar *first-visit*, se optó por ***every-visit***. Esto significa que, dentro de un mismo episodio, se consideran todas las visitas a un par estado-acción (s, a) para actualizar su valor Q, y no solo la primera.

Segundo, para que el algoritmo fuera computacionalmente tratable, la estimación de los Q-values se realizó a través de una **actualización incremental**, similar a la que usamos para el problema de los bandidos. En vez de guardar una lista con todos los retornos y promediarlos cada vez, se actualiza el valor de $Q(s, a)$ después de cada episodio usando una media incremental. La fórmula es:

$$Q_{k+1}(s, a) = Q_k(s, a) + \frac{1}{N(s, a)}(G_t - Q_k(s, a))$$

Donde G_t es el retorno obtenido desde la visita al par (s, a) y $N(s, a)$ es el contador de cuántas veces hemos visitado ese par. Esta aproximación evita tener que almacenar todos los retornos, haciendo el algoritmo mucho más eficiente en memoria y tiempo.

A continuación, se presenta el gráfico con los resultados obtenidos para el problema de **Blackjack**, siguiendo los parámetros del enunciado. Como se observa en el gráfico 4, el retorno promedio mejora drásticamente durante los primeros episodios, pasando de un valor inicial cercano a -0.4 a aproximadamente -0.07 en el primer medio millón de episodios. A partir de ahí, el aprendizaje se vuelve más fino y el rendimiento se estabiliza lentamente, convergiendo a un valor que ronda el -0.055. Las 5 corridas muestran un comportamiento muy similar, lo que sugiere que el agente está aprendiendo una política consistente y robusta.

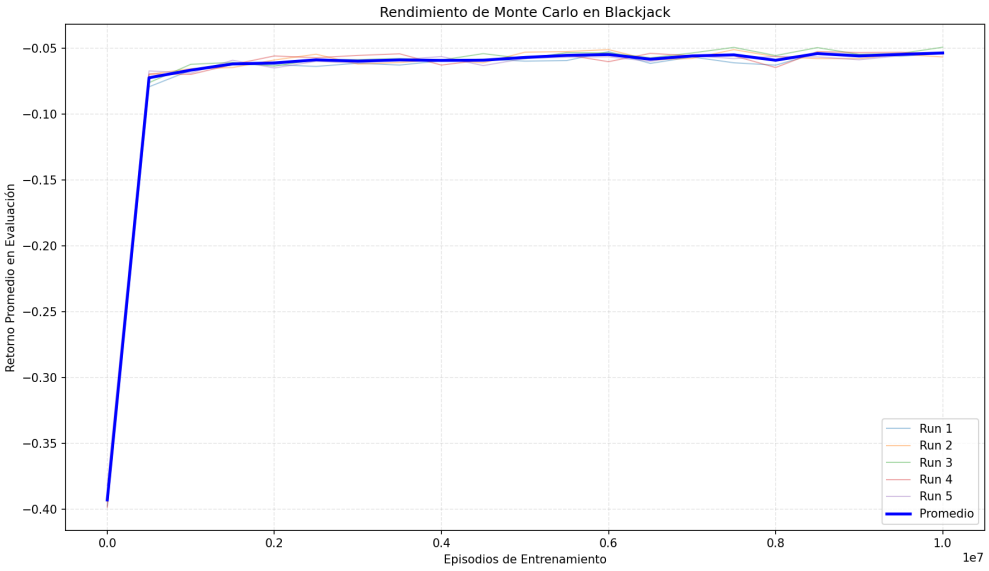


Figura 4: Rendimiento del agente de Monte Carlo en Blackjack a lo largo de 10 millones de episodios de entrenamiento. Se muestran 5 corridas independientes y su promedio.

(k) Estrategia en Blackjack

Analizando las políticas finales generadas por las 5 corridas del algoritmo, se puede extraer una estrategia general sobre cuándo dejar de pedir cartas en Blackjack. Las políticas completas para cada corrida se pueden encontrar en el **Anexo A**.

La conclusión principal es que un jugador debería **plantarse (stick) si su suma es de 19 o más**, sin importar la carta que muestre el dealer. Con una suma de 18, casi siempre es mejor plantarse, a menos que el dealer muestre una carta alta (como un 9, 10 o As). Para sumas inferiores, la decisión

depende mucho más de la carta del dealer. Generalmente, si la suma es 13 o más y el dealer tiene una carta baja (de 2 a 6), es una buena idea plantarse, ya que hay una mayor probabilidad de que el dealer se pase de 21. Por otro lado, si la suma es 12 o menos, casi siempre hay que pedir carta, ya que es imposible pasarse.

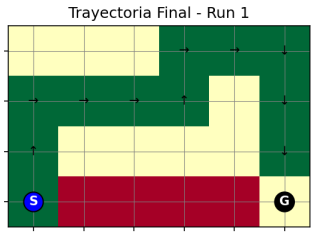
Ahora, ¿se llega siempre a la misma conclusión en las 5 corridas? **No, la política óptima no es exactamente la misma en las 5 corridas**. Si bien las estrategias generales son muy parecidas, existen diferencias en decisiones específicas. Al tener las tablas completas en el anexo, podemos señalar ejemplos concretos:

- **Mano dura de 17 vs. un As del dealer:** Si revisamos las tablas del anexo, veremos que en las corridas 3 y 4 la acción recomendada es Pedir ('H'), mientras que en las corridas 1, 2 y 5 la acción es Plantarse ('P').
- **Mano blanda de 18 vs. un 2 del dealer:** La estrategia también varía. Las corridas 2, 4 y 5 convergen a Pedir ('H'), pero las corridas 1 y 3 concluyen que es mejor Plantarse ('P').

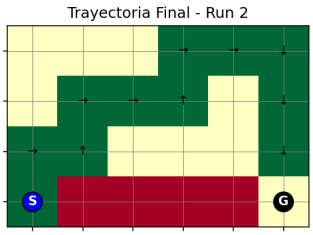
Estos ejemplos demuestran que, aunque el agente aprende una estrategia muy competente, los detalles finos pueden variar. Esto ocurre debido a la **naturaleza estocástica del método de Monte Carlo**. El algoritmo aprende a partir de la experiencia generada por episodios aleatorios. Si para un estado particular el valor esperado de pedir o plantarse es muy similar, pequeñas variaciones en las muestras aleatorias de una corrida a otra pueden hacer que el algoritmo prefiera una acción sobre la otra, resultando en políticas finales ligeramente distintas.

(l) Estrategia en Cliff Walking

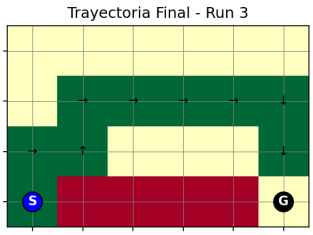
Para el problema de Cliff Walking, el objetivo es encontrar el camino más corto desde el inicio (S) hasta la meta (G) sin caer al precipicio. Caer al precipicio da una recompensa de -100 y te devuelve al inicio, mientras que cada paso normal cuesta -1. Teóricamente, el camino óptimo es pegarse al borde del precipicio, ya que es el más corto. Sin embargo, analizando los resultados, el mejor curso de acción que aprende el agente de Monte Carlo es tomar un **camino más largo pero seguro**, alejándose completamente del precipicio. A continuación se muestran las trayectorias finales para cada una de las 5 corridas.



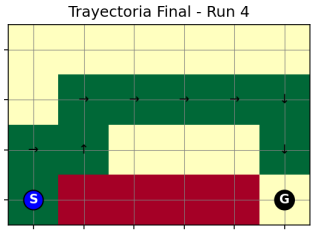
(a) Run 1



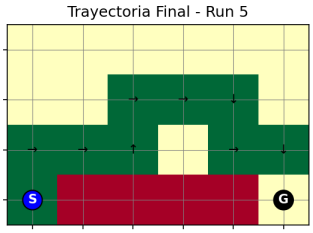
(b) Run 2



(c) Run 3



(d) Run 4



(e) Run 5

Figura 5: Trayectoria final de la política greedy aprendida en las 5 corridas del experimento Cliff Walking (width=6).

¿Es este resultado consistente en las 5 corridas? La respuesta es sí y no. **Sí, es consistente** en que todas las corridas aprenden a evitar el precipicio y optan por un camino seguro. **No, no es consistente** en el camino exacto que toman. Como se ve en la Figura 5, las corridas 3 y 4 encontraron el mismo camino, que es el más conservador posible. Sin embargo, las corridas 1, 2 y 5 encontraron rutas seguras diferentes y un poco menos óptimas.

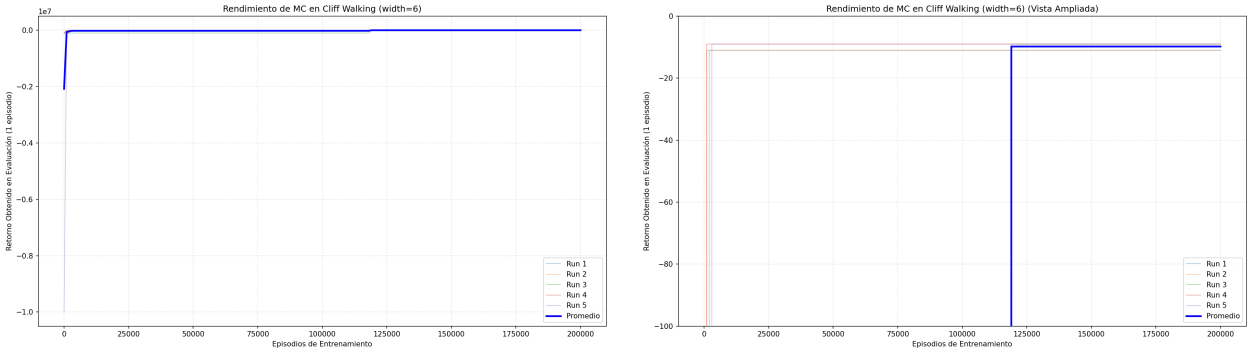
Esto ocurre por la forma en que aprende un agente *on-policy* como el nuestro. Durante el entrenamiento, la política ϵ-greedy hace que el agente explore. Inevitablemente, con $\epsilon = 0{,}1$, el agente eventualmente dará un paso hacia el precipicio y recibirá la penalización masiva de -100. Esta experiencia tan negativa hace que el valor de las acciones que llevan hacia el precipicio se vuelva extremadamente bajo. Como resultado, el agente aprende a ser muy averso al riesgo y prefiere dar un largo rodeo. Las diferencias entre las rutas seguras se deben a la **aleatoriedad de la exploración**. Dependiendo de en qué casillas exactas el agente tuvo sus "malas experiencias", desarrollará una aversión a diferentes zonas cercanas al precipicio, resultando en trayectorias finales ligeramente distintas.

(m) Estabilidad de Monte Carlo

Considerando los resultados de ambos experimentos, se puede decir que Monte Carlo es un algoritmo **relativamente estable en cuanto a rendimiento, pero no tanto en cuanto a la política final**.

En el caso de **Blackjack**, el gráfico de rendimiento muestra que las 5 corridas convergen a un retorno promedio muy similar y estable. En **Cliff Walking**, pasa algo parecido. La Figura 6 muestra la historia completa del aprendizaje. La gráfica de la izquierda (a) ilustra el rendimiento inicial, que es catastróficamente bajo (menor a -20,000) porque el agente aún no aprende a evitar el precipicio. La

gráfica de la derecha (b) es una vista ampliada que muestra cómo, una vez que el agente aprende la lección, todas las corridas convergen rápidamente a un retorno alto y estable.



(a) Vista General (b) Vista Ampliada (Zoom)

Figura 6: Rendimiento del agente en Cliff Walking. (a) Muestra la drástica mejora inicial tras aprender a evitar el precipicio. (b) Muestra la convergencia final a un retorno estable.

Aunque todas las corridas alcanzan un buen rendimiento, este no es idéntico, y como vimos en la Figura 5, las políticas finales varían. La razón de esta inestabilidad en la política es la **naturaleza estocástica del aprendizaje**. El método depende de las muestras de episodios generados. Si el valor real de dos acciones en un estado es muy parecido, pequeñas fluctuaciones aleatorias en los retornos observados durante el entrenamiento pueden hacer que una corrida prefiera una acción sobre la otra.

En conclusión, aunque puede que no converja a la misma política exacta cada vez, el algoritmo de Monte Carlo es lo suficientemente estable como para encontrar consistentemente políticas que logran un **rendimiento casi óptimo y muy similar** en cada ejecución.

(n) Cliff Walking con una Grilla de Largo 12

Para responder a esta pregunta, se configuró y ejecutó el experimento para una grilla de largo 12. Sin embargo, el tiempo de ejecución resultó ser demasiado largo, el algoritmo **demoraba una eternidad** en completar incluso una pequeña fracción de los episodios. Observando los resultados preliminares de lo poco que alcanzó a correr, el rendimiento **no era nada optimista** y no mostraba signos de aprendizaje convergente. Por esta razón, decidimos detener el experimento.

Teóricamente, este mal desempeño era esperable. El algoritmo de Monte Carlo **tiene muchas dificultades para resolver el problema** en estas condiciones debido a la **ineficiencia de la exploración en espacios de estados grandes con recompensas dispersas**. Las razones son:

1. **El espacio de estados crece:** Al pasar de un ancho de 6 a 12, el número de estados se duplica. Esto significa que se necesita una cantidad mucho mayor de episodios para explorar el ambiente de manera adecuada y visitar cada par estado-acción suficientes veces.

2. **Dificultad para encontrar la meta:** Monte Carlo aprende actualizando los valores solo al final de un episodio. En una grilla de 12x4, una política inicial completamente aleatoria tiene

una probabilidad muy baja de llegar a la meta (G) por pura casualidad. La mayoría de los episodios terminarán porque se alcanza el límite de 100,000 pasos, no porque se llegue a G.

3. **Sin señal de aprendizaje:** Si los episodios no terminan llegando a la meta, el agente nunca "verá"los estados que son buenos porque llevan a G. Sin episodios exitosos, el algoritmo no tiene ninguna señal para propagar hacia atrás y entender qué camino es el correcto. Solo aprenderá a vagar sin rumbo o a evitar caerse del precipicio, pero no a cómo llegar a la meta.

A. Políticas Finales de Blackjack

A continuación se muestran las políticas completas generadas en cada una de las 5 corridas. Leyenda: P = Plantarse (Stick), H = Pedir (Hit). La columna de la izquierda es la suma del jugador y la fila superior es la carta visible del dealer.

Run 1

Cuadro 7: Política para Manos Duras (Sin As Usable) - Run 1

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	P	P	P	P	P	P	P	P	P	P
17	P	P	P	P	P	P	P	P	P	P
16	H	P	P	P	P	P	H	H	H	H
15	H	P	P	P	P	P	H	H	H	H
14	H	H	H	H	P	P	H	H	H	H
13	H	H	H	H	P	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Cuadro 8: Política para Manos Blandas (Con As Usable) - Run 1

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	H	P	P	P	P	P	P	P	P	P
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Run 2

Cuadro 9: Política para Manos Duras (Sin As Usable) - Run 2

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	P	P	P	P	P	P	P	P	P	P
17	P	P	P	P	P	P	P	P	P	P
16	H	P	P	P	P	P	H	H	H	H
15	H	P	P	P	P	P	H	H	H	H
14	H	H	H	P	P	P	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Cuadro 10: Política para Manos Blandas (Con As Usable) - Run 2

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	H	P	P	P	P	P	P	P	H	H
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Run 3

Cuadro 11: Política para Manos Duras (Sin As Usable) - Run 3

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	P	P	P	P	P	P	P	P	P	P
17	H	P	P	P	P	P	P	P	P	P
16	H	P	P	P	P	P	H	H	H	H
15	H	P	P	P	P	P	H	H	H	H
14	H	H	P	P	P	P	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Cuadro 12: Política para Manos Blandas (Con As Usable) - Run 3

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	H	P	P	P	P	P	P	P	H	H
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Run 4

Cuadro 13: Política para Manos Duras (Sin As Usable) - Run 4

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	P	P	P	P	P	P	P	P	P	P
17	H	P	P	P	P	P	P	P	P	P
16	H	P	P	P	P	P	H	H	H	H
15	H	P	P	P	P	P	H	H	H	H
14	H	H	P	P	P	P	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Cuadro 14: Política para Manos Blandas (Con As Usable) - Run 4

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	H	H	P	P	P	P	P	P	H	H
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Run 5

Cuadro 15: Política para Manos Duras (Sin As Usable) - Run 5

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	P	P	P	P	P	P	P	P	P	P
17	P	P	P	P	P	P	P	P	P	P
16	H	P	P	P	P	P	H	H	H	H
15	H	H	P	P	P	P	H	H	H	H
14	H	H	H	P	P	P	H	H	H	H
13	H	H	P	H	P	P	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H

Cuadro 16: Política para Manos Blandas (Con As Usable) - Run 5

Suma	**A**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
21	P	P	P	P	P	P	P	P	P	P
20	P	P	P	P	P	P	P	P	P	P
19	P	P	P	P	P	P	P	P	P	P
18	H	H	H	H	P	P	P	P	H	H
17	H	H	H	H	H	H	H	H	H	H
16	H	H	H	H	H	H	H	H	H	H
15	H	H	H	H	H	H	H	H	H	H
14	H	H	H	H	H	H	H	H	H	H
13	H	H	H	H	H	H	H	H	H	H
12	H	H	H	H	H	H	H	H	H	H