Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

**Análisis de los resultados esperados**

**¿Cómo afecta una α baja (0.01) vs. una alta (0.9) a la velocidad con la que las curvas de recompensa se estabilizan?**

**Con α baja (0.01):**

* El aprendizaje es más lento y gradual.
* Las curvas de recompensa tardarán más tiempo en alcanzar valores altos.
* La convergencia es más estable pero lenta.
* El agente incorpora nueva información de manera muy conservadora, dando pequeños pasos hacia la política óptima.

**Con α alta (0.9):**

* El aprendizaje es mucho más rápido inicialmente.
* Las curvas de recompensa subirán más rápidamente en los primeros episodios.
* El agente incorpora nueva información muy agresivamente, actualizando sus Q-valores casi completamente con cada experiencia.

**¿Observas inestabilidad con valores altos de α? ¿Por qué podría ocurrir esto?**

Sí, con α alta (0.9) es probable que observes:

* Fluctuaciones grandes en la curva de recompensa incluso después de muchos episodios.
* Comportamiento errático donde el rendimiento puede empeorar repentinamente incluso después de haber sido bueno.

**Razones para esta inestabilidad:**

1. **Sobreajuste a experiencias recientes:** Con α alta, el agente descarta casi por completo su conocimiento previo y se adapta fuertemente a la última experiencia. Esto puede causar oscilaciones bruscas cuando experimenta diferentes secuencias de estados.
2. **Alta varianza en las actualizaciones:** Las grandes actualizaciones de los valores Q amplifican el ruido inherente al entorno y a la exploración ε-greedy, causando inestabilidad.
3. **Falta de promediado:** En el aprendizaje por refuerzo, queremos aproximarnos al valor esperado de las recompensas futuras. Con α alta, el agente no realiza un promedio adecuado de múltiples experiencias.

**Comparación de las Policy Grids finales para α=0.01 y α=0.5**

**Para α=0.01:**

* Probablemente verás una política menos definida o subóptima.
* Algunas zonas de la cuadrícula pueden mostrar decisiones inconsistentes.
* La ruta hacia la meta G podría no ser la más directa o eficiente.
* El aprendizaje está "incompleto" dado el número limitado de episodios (500).

**Para α=0.5:**

* Probablemente verás una política mucho más definida y clara.
* La ruta hacia la meta será más directa y consistente.
* Las decisiones en cada estado serán más decididas y óptimas.

**¿Sugiere esto que una α baja podría necesitar más episodios?** Definitivamente. Con una tasa de aprendizaje baja como 0.01, el agente necesita muchos más episodios para converger a una política óptima. Esto es porque cada actualización de los valores Q solo incorpora una pequeña fracción (1%) de la nueva información. Por lo tanto, para llegar a un mismo nivel de convergencia, un agente con α=0.01 necesitaría aproximadamente 50 veces más actualizaciones que uno con α=0.5.

**Conclusión**

La tasa de aprendizaje α representa un importante equilibrio entre:

* **Velocidad de aprendizaje:** α alta permite aprendizaje rápido.
* **Estabilidad:** α baja proporciona mayor estabilidad.

En la práctica, una buena estrategia suele ser comenzar con un α moderadamente alto (como 0.5) para aprender rápidamente y luego reducirlo gradualmente para refinar la política sin inestabilidad. Algunos algoritmos más avanzados implementan un α adaptativo que disminuye con el tiempo o con el número de visitas a cada par estado-acción.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

**1. Sobre la política de SARSA con gamma=0.90:**

Con un gamma (γ) más bajo de 0.90, la política de SARSA probablemente se vuelva **menos conservadora y más directa** comparada con γ=0.99. Esto ocurre porque:

* Un γ más bajo hace que el agente descuente más fuertemente las recompensas y penalizaciones futuras.
* La penalización grande (-100) por caer en el acantilado tiene menos influencia cuando está "descontada" por un γ menor.
* El agente se preocupa menos por lo que sucede varios pasos adelante, priorizando la ruta más corta.

Por ejemplo, para una penalización que está a 10 pasos de distancia:

* Con γ=0.99: El valor se multiplica por 0.99^10 = 0.904 (descuento del ~10%)
* Con γ=0.90: El valor se multiplica por 0.90^10 = 0.349 (descuento del ~65%)

**2. ¿Qué algoritmo se verá más afectado por el cambio en γ?**

**SARSA se verá más afectado** que Q-Learning por el cambio en γ en este entorno. Esto se debe a:

* **SARSA (on-policy)**: Aprende basándose en las acciones que realmente toma, incluyendo acciones exploratorias. Con γ alto es muy sensible al riesgo de caer durante la exploración, lo que lo lleva a comportarse conservadoramente. Con γ bajo, este efecto disminuye significativamente.
* **Q-Learning (off-policy)**: Aprende basándose en acciones óptimas futuras independientemente de su política exploratoria actual. Tiende a encontrar la ruta óptima teórica (borde del acantilado) en ambos casos, aunque con valores Q diferentes.

En CliffWalking, la diferencia en las políticas aprendidas debería ser mucho más notable en SARSA que en Q-Learning cuando se cambia el valor de gamma.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

**Análisis del decaimiento de epsilon (0.99 vs 0.9999)**

**Impacto en la curva de recompensa inicial**

Los datos muestran claramente la diferencia entre un decaimiento rápido (0.99) y lento (0.9999):

**Con decaimiento rápido (0.99)**:

* Q-Learning: La recompensa promedio mejora rápidamente, pasando de -2153.56 en el episodio 50 a -100.27 en el episodio 200.
* SARSA: Mejora aún más rápido, alcanzando -35.82 en el episodio 200.

**Con decaimiento lento (0.9999)**:

* Q-Learning: La recompensa permanece muy baja (-2876.92) incluso después de 500 episodios.
* SARSA: Mejora gradualmente pero sigue en -943.41 después de 2000 episodios.

Esto confirma que un decaimiento más rápido produce una mejora más rápida en las recompensas iniciales, ya que el agente comienza a explotar antes lo que ha aprendido. En cambio, con un decaimiento más lento, el agente sigue explorando y obteniendo malas recompensas durante mucho más tiempo.

**Estancamiento en soluciones subóptimas**

Observando las políticas aprendidas:

* **Q-Learning con decaimiento rápido (0.99)**: La política muestra un camino más "errático" con algunas direcciones contradictorias (→ ↑ ← →), lo que sugiere que se ha quedado en una solución subóptima. El agente dejó de explorar demasiado pronto (epsilon cayó por debajo de 0.1 después de 250 episodios).
* **SARSA con decaimiento lento (0.9999)**: Muestra un patrón más consistente, alejándose del borde del acantilado, lo que es característico de SARSA que aprende políticas más seguras.

El algoritmo Q-Learning con decaimiento rápido es claramente el más propenso a estancarse en soluciones subóptimas. Esto ocurre porque Q-Learning es off-policy (aprende la política óptima independientemente de la política que está siguiendo) y si deja de explorar demasiado pronto, puede quedarse con una estimación inexacta de los valores Q.

**Impacto de epsilon\_min (0.01 vs 0.0)**

**Diferencias en rendimiento final y política**

Comparando las políticas y recompensas:

**Para Q-Learning**:

* Con epsilon\_min = 0.01: Recompensa final = -18.53
* Con epsilon\_min = 0.0: Recompensa final = -15.27

La política de epsilon\_min = 0.0 muestra más variaciones, probablemente porque pudo converger más firmemente a los valores Q aprendidos sin la "interferencia" de exploraciones aleatorias continuas.

**Para SARSA**:

* Con epsilon\_min = 0.01: Recompensa final = -17.10
* Con epsilon\_min = 0.0: Recompensa final = -17.00

Las políticas de SARSA muestran un patrón similar en ambos casos: mantener una distancia segura del acantilado. Sin embargo, es notable que SARSA con epsilon\_min = 0.0 converge exactamente a -17.00 durante muchos episodios consecutivos, indicando que ha dejado completamente de explorar.

**Beneficios de mantener exploración mínima**

Los resultados confirman varios beneficios de mantener epsilon\_min = 0.01:

1. **Evitar óptimos locales**: Para Q-Learning con decaimiento rápido, mantener una exploración mínima podría haber ayudado a evitar algunos de los patrones erráticos en la política.
2. **Robustez ante incertidumbre**: El patrón más consistente en SARSA con epsilon\_min = 0.01 sugiere una política más robusta que puede adaptarse mejor a entornos estocásticos.
3. **Prevención de sobreajuste**: La exploración continua evita que el agente se sobreajuste a experiencias específicas, manteniendo cierta generalizabilidad.

**Conclusiones clave:**

1. **Sobre el decaimiento de epsilon**:
   * Un decaimiento rápido (0.99) mejora rápidamente las recompensas iniciales pero puede llevar a políticas subóptimas.
   * Un decaimiento lento (0.9999) requiere mucho más tiempo para convergir pero potencialmente puede encontrar mejores políticas a largo plazo.
2. **Sobre epsilon\_min**:
   * Establecer epsilon\_min = 0.0 puede proporcionar un mejor rendimiento final en términos de recompensa si el agente ya ha encontrado una buena política.
   * Mantener epsilon\_min = 0.01 proporciona mayor robustez y capacidad de adaptación, especialmente importante en entornos que pueden cambiar o donde es crítico evitar ciertas áreas de alto riesgo.
3. **Diferencias entre Q-Learning y SARSA**:
   * SARSA muestra consistentemente una política más segura, alejándose del borde del acantilado.
   * Q-Learning tiende a encontrar caminos más "audaces" que pueden acercarse más al acantilado, potencialmente obteniendo mejores recompensas pero con mayor riesgo.

En un entorno como CliffWalking, SARSA con un decaimiento más lento y un epsilon mínimo positivo parece ser la opción más robusta a largo plazo, aunque Q-Learning con decaimiento rápido puede proporcionar mejores recompensas si logra convergir a una buena política.

Interfaz de usuario gráfica, Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

**Respuestas esperadas basadas en los fundamentos teóricos**

1. **¿Consigue el agente SARSA llegar a la meta consistentemente durante el renderizado, o se queda atascado en bucles?**

Con solo 300 episodios de entrenamiento, es muy probable que el agente SARSA no logre llegar a la meta consistentemente durante el renderizado. SARSA podría mostrar comportamientos como:

* Quedarse atascado en bucles
* Acercarse peligrosamente al acantilado
* Tomar rutas ineficientes o incluso caer por el acantilado

1. **¿Por qué la falta de episodios afecta más a SARSA en este escenario para encontrar una ruta funcional?**

La diferencia fundamental está en cómo aprenden los dos algoritmos:

* **Q-Learning** es un algoritmo off-policy que aprende directamente la política óptima. Actualiza sus Q-valores usando el máximo Q-valor del siguiente estado, independientemente de la acción que realmente tome. Esto le permite aprender más rápidamente la política óptima incluso con exploración.
* **SARSA** es un algoritmo on-policy que actualiza sus Q-valores basándose en la acción real que tomará en el siguiente estado (incluyendo acciones exploratorias). Esto hace que SARSA sea más conservador y necesite más episodios para converger.

En el entorno CliffWalking específicamente:

* SARSA tiende a aprender rutas más seguras pero necesita más episodios para convergir a una solución
* Q-Learning aprende rápidamente a tomar el camino más corto, incluso si es arriesgado
* Con pocos episodios, SARSA no tiene tiempo suficiente para balancear exploración y explotación adecuadamente

La Policy Grid con 300 episodios para SARSA probablemente mostrará un patrón menos coherente y posiblemente rutas incompletas o peligrosas, comparada con la versión de 2000 episodios donde SARSA normalmente encuentra un camino más seguro pero eficiente.

**Análisis del estado (fila=2, col=5) en los Q-Value Heatmaps**

**Comparación de valores Q para la acción 'Abajo' (↓)**

En el estado (fila=2, col=5):

* **Q-Learning**: Tendría un valor Q significativamente más negativo para la acción 'Abajo'
* **SARSA**: Tendría un valor Q menos negativo para la misma acción

**Explicación basada en las ecuaciones de actualización**

Esta diferencia se explica por cómo cada algoritmo actualiza sus valores Q:

1. **Q-Learning** utiliza: target = reward + gamma \* max(Q(next\_state))
   * Es un método "off-policy" que siempre considera la acción óptima en el siguiente estado
   * Actualiza su valor Q basándose en lo que sería la mejor acción posible, independientemente de la política exploratoria
2. **SARSA** utiliza: target = reward + gamma \* Q(next\_state, next\_action)
   * Es un método "on-policy" que considera la acción real que se tomará según la política actual
   * Actualiza su valor Q basándose en la acción que realmente tomará, incluidas las acciones exploratorias

**Por qué Q-Learning tiene un valor más negativo para "Abajo"**

Si nos situamos en (fila=2, col=5) y consideramos la acción "Abajo":

* Esta acción nos llevaría a (fila=3, col=5), que es parte del acantilado
* Caer al acantilado resultaría en una recompensa muy negativa (típicamente -100)
* **Q-Learning**: Solo considerará esta acción "Abajo" cuando calcule que es la mejor según sus valores Q. A través del entrenamiento, rápidamente aprende que esta acción lleva a una recompensa muy negativa, y la penaliza fuertemente.
* **SARSA**: Considera las acciones exploratorias que podría tomar después. Durante el entrenamiento, ocasionalmente puede explorar y tomar acciones menos óptimas después de caer, lo que "suaviza" el valor negativo asociado con la acción de caer. Además, es más probable que SARSA evite completamente esta ruta durante el aprendizaje (eliminando parte de las muestras negativas).

**Descripción de las rutas en las Policy Grids**

**Q-Learning (off-policy)**

* **Características de la ruta**:
  + Típicamente sigue un camino muy cercano al acantilado
  + Es la ruta objetivamente más corta al objetivo
  + Tiene mayor riesgo de caer al acantilado si hay algún error o factor aleatorio
* **Explicación off-policy**:
  + Q-Learning aprende la política óptima independientemente de las acciones exploratorias
  + No considera el riesgo de exploración en su política final
  + Asume que el agente siempre seguirá la mejor acción (determinística)

**SARSA (on-policy)**

* **Características de la ruta**:
  + Típicamente sigue un camino más alejado del acantilado (por ejemplo, por la fila 2)
  + Es una ruta más larga para llegar al objetivo
  + Es significativamente más segura, con menor riesgo de caer al acantilado
* **Explicación on-policy**:
  + SARSA aprende considerando su política de exploración
  + Incorpora el riesgo de tomar acciones exploratorias en su cálculo de valores
  + Durante el entrenamiento, las caídas ocasionales por exploración "enseñan" a SARSA a mantener distancia del acantilado

**Comparación y análisis**

* **Ruta más corta**: Q-Learning, ya que optimiza puramente para la recompensa total esperada sin considerar los riesgos de exploración.
* **Ruta más segura**: SARSA, ya que incorpora la posibilidad de desviarse de la política óptima y por tanto evita regiones peligrosas.

Esta diferencia ilustra perfectamente la distinción fundamental entre algoritmos on-policy y off-policy:

* Los métodos **off-policy** como Q-Learning aprenden la política óptima asumiendo un comportamiento determinístico perfecto.
* Los métodos **on-policy** como SARSA aprenden una política óptima considerando el comportamiento real del agente, incluidas sus imperfecciones.

En el mundo real, donde la ejecución perfecta no está garantizada, un enfoque on-policy como SARSA puede ser más robusto, aunque menos óptimo en el sentido matemático estricto.

**Análisis Teórico de las Recompensas en Q-Learning y SARSA**

**Caídas en las Gráficas de Recompensa**

**¿Qué algoritmo muestra "caídas" más pronunciadas?**

**Q-Learning** típicamente muestra caídas más pronunciadas en la gráfica de recompensas durante el entrenamiento.

**¿A qué evento del entorno corresponden estas caídas?**

Estas caídas bruscas corresponden a los episodios donde el agente cae por el acantilado durante la evaluación. En el entorno CliffWalking, caer al acantilado resulta en una recompensa muy negativa (típicamente -100) y termina el episodio inmediatamente.

**Razones para las caídas más pronunciadas en Q-Learning:**

1. **Comportamiento optimista**: Q-Learning tiende a aprender un camino muy cercano al borde del acantilado porque es la ruta más corta y por tanto teóricamente óptima.
2. **Política greedy durante evaluación**: Durante la evaluación, Q-Learning sigue una política completamente greedy (escoge la mejor acción según sus valores Q).
3. **Falta de margen de seguridad**: Como Q-Learning no considera los errores de exploración en su política aprendida, no desarrolla un "margen de seguridad" y navega peligrosamente cerca del acantilado.

Cuando el agente Q-Learning se encuentra directamente al lado del acantilado, cualquier desviación o exploración residual puede resultar en una caída, causando esas grandes caídas en la gráfica de recompensas.

**Recompensa Promedio Final**

**¿Por qué la recompensa promedio final de Q-Learning podría no ser mejor que SARSA?**

Aunque Q-Learning aprende teóricamente la ruta óptima (la más corta), su recompensa promedio final puede no ser mejor (o incluso ser peor) que SARSA por varias razones:

1. **Riesgo vs. Recompensa**:
   * La ruta "óptima" de Q-Learning es más arriesgada al estar cerca del acantilado
   * Una sola caída (-100 de recompensa) puede anular los beneficios acumulados de tomar rutas cortas
2. **Exploración residual**:
   * Durante la evaluación, si queda algo de exploración (epsilon > 0), Q-Learning es más propenso a caídas catastróficas por su proximidad al acantilado
   * SARSA, que aprende un camino más seguro, es más resistente a los efectos de la exploración residual
3. **Varianza en el rendimiento**:
   * Q-Learning tiende a tener mayor varianza en su rendimiento
   * SARSA suele ser más consistente, lo que puede resultar en una mejor recompensa promedio
4. **Optimalidad vs. Robustez**:
   * Q-Learning optimiza para el caso ideal (seguir perfectamente la política óptima)
   * SARSA optimiza para el caso real (considerando errores y exploración)
5. **Promediado de recompensas**:
   * Al promediar las recompensas de múltiples episodios, unas pocas caídas al acantilado (-100 cada una) pueden impactar significativamente el promedio de Q-Learning
   * SARSA, aunque toma rutas más largas (con recompensas acumuladas ligeramente menores por paso), puede mantener un promedio más alto al evitar consistentemente las caídas catastróficas

Este fenómeno ilustra un principio importante en el aprendizaje por refuerzo: la política teóricamente óptima no siempre es la mejor en la práctica, especialmente cuando hay incertidumbre o exploración. SARSA, al considerar el comportamiento real del agente durante el aprendizaje, desarrolla una política que podría ser subóptima en el sentido estricto pero más robusta y con mejor desempeño promedio en la práctica.

**Análisis Teórico: Q-Learning vs SARSA en Entorno Determinista**

**¿Esperarías una gran diferencia en la política final aprendida por Q-Learning y SARSA?**

**No, en un entorno determinista como FrozenLake con is\_slippery=False, esperaría que tanto Q-Learning como SARSA converjan a políticas idénticas o extremadamente similares.**

**Razones teóricas:**

1. **Eliminación de la aleatoriedad en las transiciones**:
   * Con is\_slippery=False, cada acción lleva al agente de forma determinista al siguiente estado deseado
   * No hay "deslizamientos" aleatorios que puedan llevar al agente a estados no deseados
2. **Convergencia de las ecuaciones de actualización**:
   * En Q-Learning: target = reward + gamma \* max(Q(next\_state))
   * En SARSA: target = reward + gamma \* Q(next\_state, next\_action)

En un entorno determinista y con suficiente exploración, ambos algoritmos eventualmente seleccionarán la misma acción óptima en cada estado. Cuando esto ocurre, la diferencia fundamental entre sus ecuaciones de actualización desaparece, porque:

* + La next\_action en SARSA será eventualmente la acción con el máximo valor Q
  + Por lo tanto, Q(next\_state, next\_action) = max(Q(next\_state)) una vez convergidos

1. **Sin penalización por comportamiento exploratorio**:
   * La principal diferencia entre Q-Learning y SARSA surge cuando hay riesgo en la exploración
   * En un entorno determinista, no hay "riesgo" asociado con estar cerca de estados peligrosos
   * Sin esta diferencia en el manejo del riesgo, ambos algoritmos favorecen el mismo camino óptimo
2. **Garantía de optimalidad**:
   * En un MDP determinista con recompensas limitadas, ambos algoritmos convergen a la política óptima con suficiente exploración y un decaimiento adecuado de epsilon
   * La exploración probabilística asegura que todos los estados se visiten lo suficiente como para descubrir la ruta óptima

**Diferencias menores posibles:**

Aunque las políticas finales deberían ser idealmente idénticas, podrían observarse pequeñas diferencias debido a:

1. **Velocidad de convergencia**:
   * Q-Learning podría converger ligeramente más rápido por su naturaleza off-policy que propaga directamente valores óptimos
   * Estas diferencias de velocidad podrían provocar pequeñas variaciones si el entrenamiento se detiene antes de la convergencia completa
2. **Comportamiento en empates de valores Q**:
   * Si múltiples acciones tienen valores Q idénticos o muy similares, las políticas pueden diferir en cómo rompen estos empates
   * Esto es más probable en las primeras etapas de aprendizaje o en estados donde múltiples caminos son igualmente óptimos
3. **Inicialización y orden de experiencias**:
   * Diferencias en la inicialización o en el orden de experiencias pueden llevar a pequeñas diferencias, especialmente con pocos episodios

Sin embargo, estas diferencias serían principalmente cuestiones de implementación o entrenamiento, no diferencias fundamentales en el comportamiento de los algoritmos en un entorno determinista.

**Análisis Teórico: Expected SARSA vs SARSA y Q-Learning**

He implementado el agente ExpectedSarsaAgent y modificado el script principal para comparar los tres algoritmos (Q-Learning, SARSA y Expected SARSA) en el entorno CliffWalking. Ahora vamos a analizar teóricamente las diferencias esperadas entre ellos.

**Comparación de la curva de recompensas de Expected SARSA**

**¿Se parece más a alguno de los dos algoritmos?**

**Expected SARSA teóricamente combina características de ambos algoritmos, pero su curva de recompensas debería parecerse más a la de Q-Learning** por las siguientes razones:

1. **Actualización basada en expectativas**:
   * Expected SARSA utiliza un valor esperado sobre todas las posibles acciones en el siguiente estado
   * Esta expectativa es más estable que la elección de una única acción específica (como hace SARSA)
   * Se aproxima más a la maximización que hace Q-Learning sobre el siguiente estado
2. **Equilibrio entre políticas**:
   * Expected SARSA representa un punto intermedio entre SARSA (completamente on-policy) y Q-Learning (completamente off-policy)
   * Sin embargo, a medida que epsilon disminuye durante el entrenamiento, Expected SARSA se acerca más al comportamiento de Q-Learning
3. **Camino óptimo vs camino seguro**:
   * Al igual que Q-Learning, Expected SARSA tiende a aprender caminos más cercanos al óptimo
   * Pero incorpora cierto grado de "cautela" al considerar las posibles acciones exploratorias

**¿Por qué Expected SARSA podría tener menor varianza que SARSA?**

**Expected SARSA tiene menor varianza que SARSA por las siguientes razones fundamentales:**

1. **Eliminación de la aleatoriedad de muestreo**:
   * SARSA actualiza en base a una única acción muestreada según la política ε-greedy
   * Este muestreo introduce varianza aleatoria en cada actualización
   * Expected SARSA reemplaza este muestreo con un cálculo determinista del valor esperado sobre todas las acciones posibles
2. **Cálculo analítico vs muestreo estocástico**:
   * Expected SARSA calcula analíticamente la expectativa sobre todas las acciones, ponderando por su probabilidad
   * Este enfoque analítico elimina la varianza muestral inherente a SARSA, que depende de qué acción específica sea seleccionada en cada paso
3. **Actualización más suave**:
   * Las actualizaciones de Expected SARSA son más suaves y predecibles
   * Mientras que SARSA puede tener grandes fluctuaciones dependiendo de si las acciones exploradas llevan a estados con alta o baja recompensa
4. **Consistencia en el aprendizaje**:
   * Al considerar todas las acciones posibles ponderadas por su probabilidad, Expected SARSA mantiene mayor consistencia entre episodios
   * Esta consistencia se traduce en una curva de aprendizaje más suave con menos picos y valles

En términos matemáticos, la varianza de Expected SARSA es menor porque elimina una fuente de aleatoriedad: la selección de la siguiente acción. Mientras que SARSA introduce varianza tanto en la selección de estados (a través de la dinámica del entorno) como en la selección de acciones (a través de la política ε-greedy), Expected SARSA solo mantiene la primera fuente de varianza.

**Comparación de política final**

Esperaríamos que la política final de Expected SARSA:

1. Sea más cercana a la óptima que SARSA (similar a Q-Learning)
2. Pero ligeramente más cautelosa que Q-Learning en áreas de riesgo
3. Muestre un patrón de ruta que sea un equilibrio entre la ruta más corta (Q-Learning) y la ruta más segura (SARSA)

Esta combinación de características hace de Expected SARSA un algoritmo muy interesante que teóricamente combina lo mejor de ambos mundos: la optimalidad de Q-Learning con parte de la seguridad de SARSA, y todo ello con menor varianza en el aprendizaje.