Entrega Tarea 1:
Programación dinámica

Fecha de entrega: Martes 05 de abril, 23:59 hrs.

Estudiante: Francisco Molina L.
Profesor: Javier Ruiz del Solar
Auxiliar: Francisco Leiva C.
Semestre: Otoño 2022

1. Parte I

1. EL **MDP** $\langle \boldsymbol{S}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{R}, \boldsymbol{\gamma} \rangle$ de una grilla bidimensional de dimensiones H x W se define por:

   - El espacio de estados $\boldsymbol{S}$: Corresponde a cada una de las posiciones que puede tomar el agente en la grilla bidimensional. En este caso hay tres tipos de estados: válidos, muros y terminales (diferenciados por su recompensa). Los estados válidos son las casillas por donde el agente puede transitar, los muros son casillas hacia las cuales el agente no puede moverse, y si lo intenta, permanece en su estado anterior, y los estados terminales corresponden a las casillas objetivo.

     De manera arbitraria se enumeran los estados desde 0 hasta $H * W$, comenzando por la esquina superior izquierda y terminando por la esquina inferior derecha. De modo que un estado se puede representar por sus coordenadas en la grilla o por su número asignado.

   - Las acciones $\boldsymbol{A}$: Corresponden a las direcciones que puede tomar el agente en la grilla, en este caso se representan por números enteros del 0 al 3: arriba (0), abajo (1), derecha (2) e izquierda (3).

   - La matriz de transición de estados $\boldsymbol{P}$: En la literatura $\mathcal{P}^a_{ss'}$, corresponde a la probabilidad de transicionar desde el estado s, hacia el estado s' dada una acción a. Dada la enumeración escogida, se debería representar por un *array* de dimensiones ($H * W$, $H * W$, 4), es decir, 4 matrices bidimensionales, una para cada acción, donde cada valor representa la probabilidad de pasar desde el estado s (índice de fila) hacia el estado s' (índice de columna), dado que se ejecuta una acción a (índice de profundidad).

     Dado que dicho *array* tendría muchos elementos con probabilidad 0 para cada acción, pues cada estado puede a lo más alcanzar otros 4 estados, se opta por implementar la matriz de transición de estados de manera distinta en el código. Para cada estado válido se crea un diccionario que contiene un diccionario para cada estado sucesor posible, los cuales contienen las probabilidades de llegar dada la acción, de la forma:

     $$s = \{s'_1 : \{0 : \_, 1 : \_, 2 : \_, 3 : \_\}, \ \dots, s'_4 : \{0 : \_, 1 : \_, 2 : \_, 3 : \_\}\} \tag{1}$$

     De esta manera sólo se almacenan las probabilidades útiles.

   - La función de recompensa $\boldsymbol{R}$: En la literatura $\mathcal{R}^a_s$, corresponde a la recompensa que recibe el agente por ejecutar la acción a desde el estado s, se representa por una matriz bidimensional del mismo tamaño que la grilla, donde cada valor representa la recompensa que recibe el agente al llegar a dicha casilla. En este caso la función de recompensa se utiliza como un descuento ($-1{,}0$ en cada casilla valida) para que así el agente minimice la cantidad de acciones que le toma llegar a la casilla objetivo. Para los muros se usa como recompensa NaN y para las terminales 0.

   - El factor de descuento $\boldsymbol{\gamma}$: Representa la incertidumbre sobre el futuro para el agente, hace que se valoren más o menos las recompensas futuras dependiendo del valor escogido.

2. La función `policy_evaluation` itera sobre todos los estados válidos, calcula su nuevo valor y lo almacena en la matriz `new_value`. Cuando se termina el barrido de los estados se revisa la diferencia entre la función de valor actual y la anterior mediante la norma infinito:

$$\Delta = \max(\Delta, max_{s \in \mathcal{S}} |v_{k+1}(s) - v_k(s)|) \tag{2}$$

   La función `policy_improvement` itera sobre todos los estados válidos y para cada uno calcula la mejor acción posible según su función de valor actual. Si la nueva política es igual a la de la iteración anterior, se interrumpe el algoritmo.

   Las funciones asociadas a *policy_evaluation* se encuentran completamente documentadas en el código adjunto.

3. La iteración de política logra converger a la política óptima en tan solo 7 iteraciones, produciendo la función de valor y la política mostradas en las **Figuras 1 y 2**:

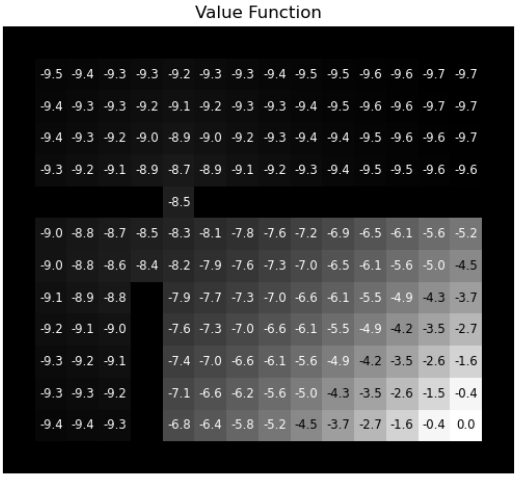

Figura 1: Función de valor PI base

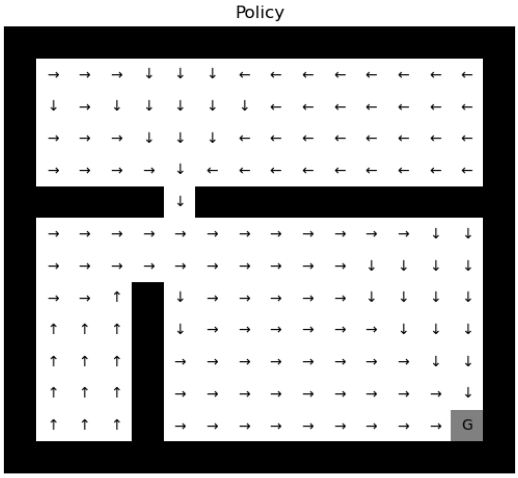

Figura 2: Política PI base

2. Parte II

1. La función `run_value_iteration` itera sobre todos los estados posibles, para cada uno actualiza su valor y la política, computando todos los posibles valores y escogiendo el mejor. Al igual que en `policy_evaluation` se termina el ciclo cuando Δ (2) se vuelve muy pequeño. Las funciones asociadas a *run_value_iteration* se encuentran completamente documentadas en el código adjunto.

2. La iteración de valor converge en tan solo 45 iteraciones a la política óptima, entregando la función de valor y la política presentadas en las **Figuras 3 y 4**:

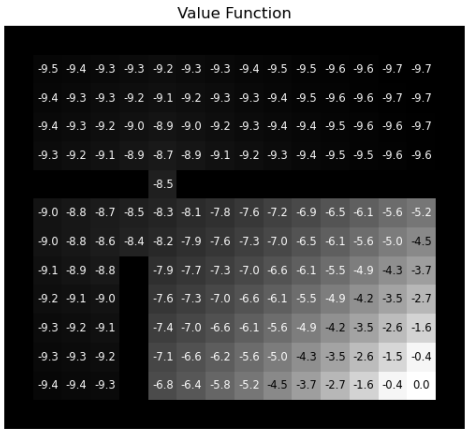

Figura 3: Función de valor VI base

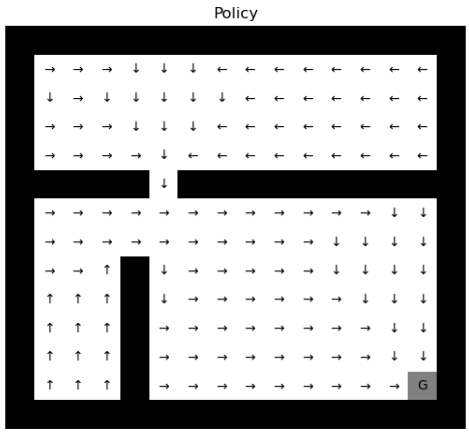

Figura 4: Política VI base

3. Llevar la probabilidad $1 - p$ a 0,0 es equivalente a llevar p a 1,0, con esto la iteración de política produce en 1000 iteraciones la función de valor y política presentadas en las **Figuras 5 y 6**:

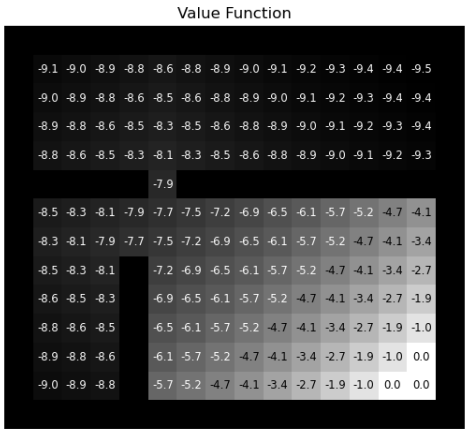

Figura 5: Función de valor PI con $p_{dir} = 1$

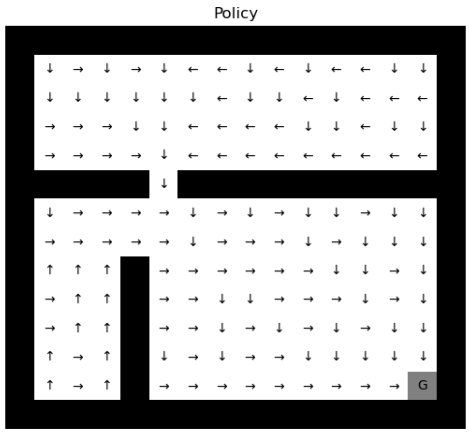

Figura 6: Política PI con $p_{dir} = 1$

Mientras que la iteración de valor produce en tan solo 28 iteraciones la función de valor y política presentadas en las **Figuras 7 y 8**:

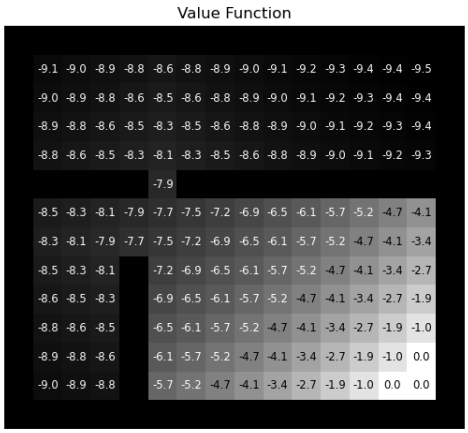

Figura 7: Función de valor VI con $p_{dir} = 1$

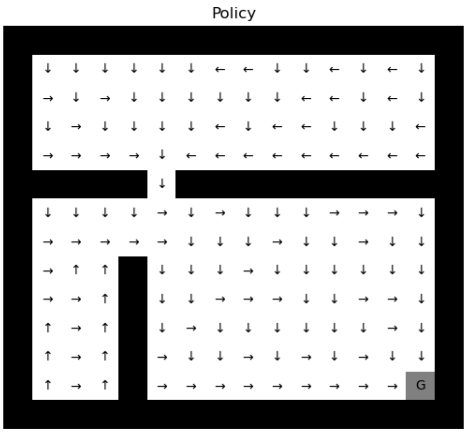

Figura 8: Política VI con $p_{dir} = 1$

Este cambio en la probabilidad elimina la incertidumbre sobre si el agente efectivamente va a llegar a la celda que seleccionó. Esto se traduce a valores más uniformes en las mismas diagonales, como se puede ver en las **Figuras 1 y 3** las celdas más cercas de los muros tienen valores menores que las de su misma diagonal, mientras que en las **Figuras 5 y 7** los valores en una misma diagonal son iguales. Este cambio también se ve medianamente reflejado en las políticas, en las **Figuras 2 y 4** (idénticas entre sí) se puede ver que el agente tiende a mantener su dirección, mientras que en las **Figuras 5 y 7** el agente sólo intenta acercarse al objetivo, escogiendo una dirección al azar en los empates.

También es importante destacar que la iteración de política se demora el máximo de las iteraciones puesto que se queda atascada en un ciclo entre más de una política óptima, debido a todas las celdas en las que hay empates. Esto se podría arreglar añadiendo un par de banderas para revisar si ya se ha obtenido la misma política previamente, pero esto se escapa del objetivo de la tarea.

4. Fijar $1 - p$ a 0.4 es equivalente a llevar a p a 0,6. Con un factor de descuento de 0.2, *value iteration* entrega en 7 iteraciones la función de valor y la política mostradas en las **Figura 9 y 10**:

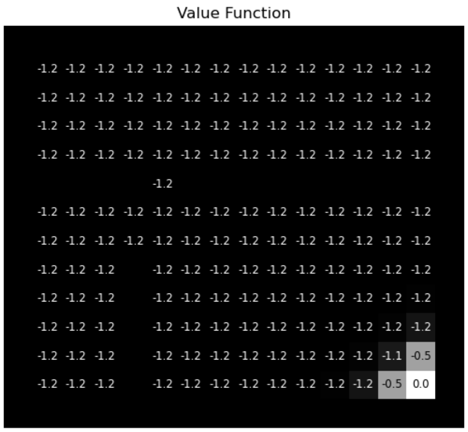

Figura 9: Función de valor

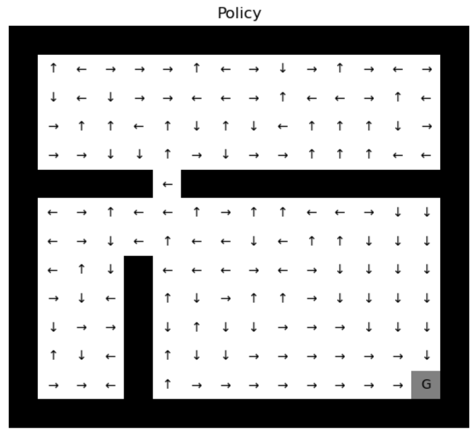

Figura 10: Política

Mientras que con un factor de descuento de 1.0 se obtiene en 96 iteraciones la función de valor y la política mostradas en las **Figura 11 y 12**:

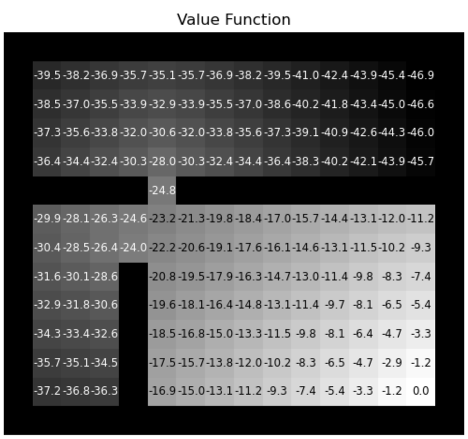

Figura 11: Función de valor

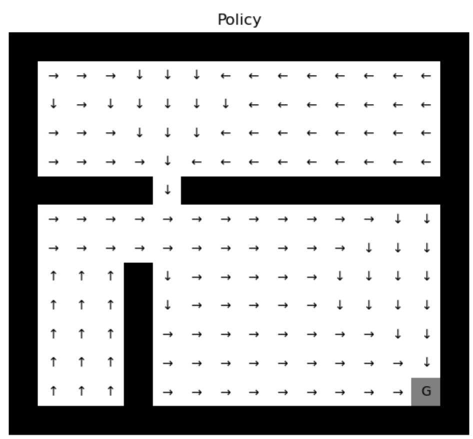

Figura 12: Política

Cuando el factor de descuento es muy cercano a 0,0, el MDP se vuele miópico, es decir, se preocupa sólo de las recompensas inmediatas. Como se puede ver en la **Figura 9** sólo las celdas cercanas al objetivo tienen un valor distinto al mínimo, esto es porque el agente sólo obtiene recompensas inmediatas en esas celdas, lo que se traduce a la política de la **Figura 10** en la que el agente no sabe qué hacer lejos del objetivo. Cabe destacar que en la **Figura 9** todos los valores de casillas que están a más de 3 del objetivo son iguales, esto es porque para el agente es igual si está a 3 o a 25 casillas del objetivo, si no obtiene una recompensa inmediata esa casilla es igual de mala.

Mientras que un factor de descuento muy cercano a 1,0, el MDP tiene muy buena vista, es decir, contempla todas las recompensas que va a obtener desde el tiempo inicial hasta el final. Esto implica en la función de valor, **Figura 11**, que los valores de cada casilla son proporcionales a su distancia al objetivo, incluso aquellas que están detrás de los muros (a diferencia del caso de la **Figura 7**). Dicha función de valor produce la mejor política posible, mostrada en la **Figura 12**.

5. Dado que la recompensa es de $-1{,}0$ por cada transición de casilla, cuando $\gamma = 1$ la función de valor representa literalmente la distancia restante hacia el objetivo. La función de valor mostrada en la **Figura 11** tiene valores decimales que no se condicen exactamente con la distancia, pero esto es porque hay sólo un 60 % de probabilidades de que la acción seleccionada se ejecute, por tanto cada casilla tiene como valor su distancia más un pequeño valor correspondiente a la incertidumbre de no llegar a la mejor casilla.

   Para comprobar empíricamente que en este caso los valores corresponden a la distancia, basta con fijar p en 1,0, con lo cual se obtiene la función de valor mostrada en la **Figura 13**:

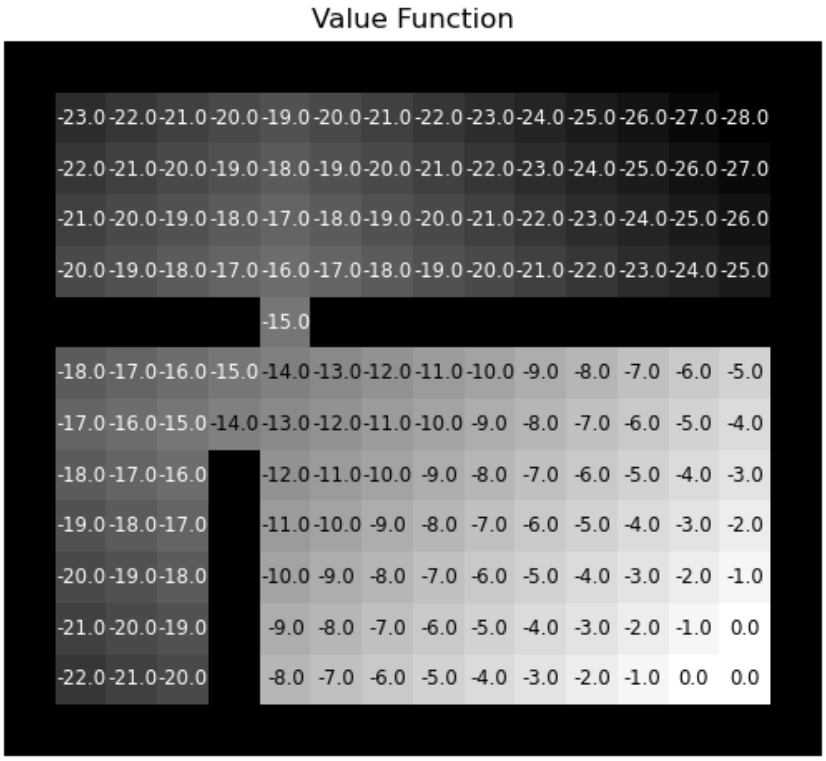

Figura 13: Función de valor $\propto$ distancia