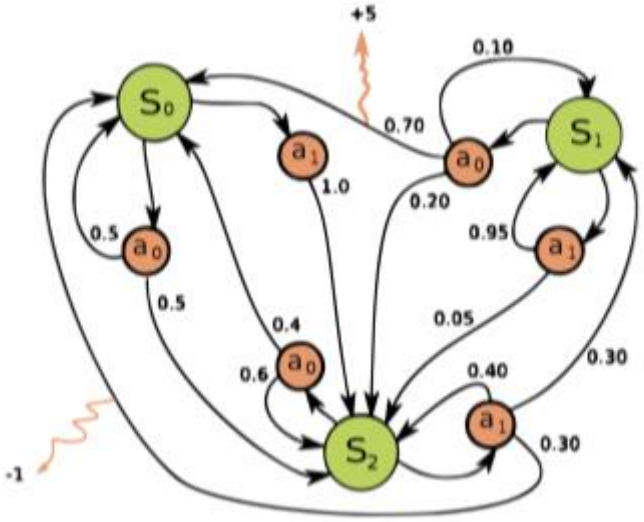
Planificación en Entornos con Incertidumbre

Ingeniería Informática del Software – Inteligencia Artificial



José Manuel Ruiz-Mateos

dpto. Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Sevilla, España

josruirui6@alum.us.es

Raúl García de la Poza

dpto. Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Sevilla, España

raugarde@alum.us.es

Resumen: Realización del trabajo "Planificación en Entornos con Incertidumbre". En la siguiente memoria observaremos el proceso seguido por los miembros del grupo, su exitosa realización y los aspectos aprendidos por los mismos.

Introducción

Nuestra meta trata en realizar exitosamente una implementación que permita mostrar y resolver los resultados de un problema planteado como POMDP mediante uno de los dos algoritmos. Estos dos algoritmos son POMCP(Partially Observable Monte-Carlo Planning) o PBVI(Point-Based Value Iteration). También tendremos que mostrar el resultado de dos formas distintas. La primera forma consistirá en una simulación interactiva y la otra en una simulación silenciosa o también conocida como benchmark.

1. Preliminares

1.1. Estudio del Enunciado

Tras la asignación aleatoria del trabajo el equipo decidió unirse para estudiar y comprender los objetivos del proyecto. Decidimos que lo mejor sería declarar un horario de trabajo y unos plazos no absolutos para desarrollar el trabajo y poder cumplir exitosamente la elaboración del mismo. Para ello, primero hicimos un análisis sobre los Procesos de Decisión de Markov para posteriormente enfocarnos en los Procesos de Decisión de Markov Parcialmente Observables (POMDP).

Nos dimos cuenta de que los POMDP no se conoce el estado actual. Trata de una probabilidad sobre el conjunto de estados que es representado a través de su creencia o belief. En resumen, cada vez que realizamos una acción, se desconoce el estado que se alcanza, pero se obtiene una observación que permite actualizar el belief.

Sobre la instalación e implementación del software destacar el entorno que vamos a utilizar “PyCharm”. Tras instalar el entorno de desarrollo accedimos y estudiamos la librería “PyPOMDP” con el objetivo de conocer su utilidad. Con la ayuda del “Readme.md” importamos la librería en nuestro entorno y los respectivos paquetes de Python.

2. Metodología

Al empezar nos dimos cuenta de que algo fallaba, existía el problema del tigre de 2 puertas estando solo planteado no se ejecutaba bien. Con la ayuda del problema del tigre de 3 puertas pudimos corregir el formato del documento.

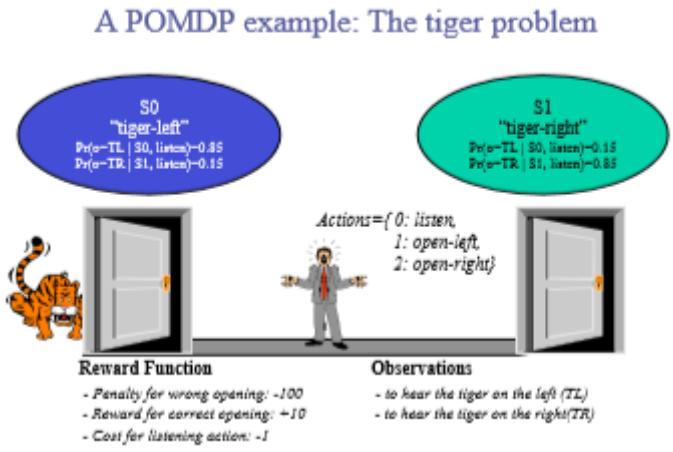
Tras comprobar el código pudimos ejecutar la instrucción principal. En los resultados de la consola apreciamos que la salida que aparece se mostraba un número determinado de veces sin parar variando entre los estados y las acciones disponibles, dado el valor por defecto (100) del parámetro "max_play" de la clase "main.py". Cuando creíamos que lo teníamos bien nos surgió el error de que el algoritmo no finalizaba.

Fragmento de la salida inicial:

```
# Step = 4025
Taking action: listen
Observation: tiger-left
Reward: -1.0
New Belief: [0.956679, 0.043321]
```

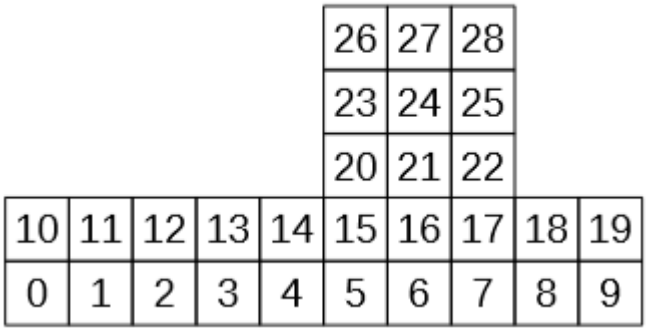
Una vez analizada la estructura de la librería, realizamos la implementación y en este documento mostramos la metodología empleada para parar el algoritmo cuando alcance uno de los estados finales. Además, también mostraremos cómo hemos implementado la simulación interactiva y silenciosa.

2.1. Problema del Tigre



Para detener el problema del tigre es necesario indicar a la librería cual es la acción de parada. En nuestro caso, cuando el sujeto abre una de las puertas, independientemente cual sea. Para ello identificamos en la librería la clase “pomdp_runner.py” que itera sobre los pasos y en concreto la función “run” que ejecuta cada uno de ellos teniendo en cuenta la acción de abrir.

2.2. Problema del Tag



En el problema del Tag para detenerlo es necesario indicar cual es el estado final. Cuando el estado indica que el robot T ha sido capturado el problema finaliza ya que significa que el robot S ha atrapado al robot T. Implementamos en la clase “pomdp_runner.py” y en concreto en la función “run” un método para parar la simulación como ya hemos hecho en el problema del tigre pero teniendo en cuenta el tagged actual.

Tenemos que tener en cuenta los límites del tablero y que para el robot le sea imposible salir de este. Por eso debemos poner una restricción para que no salga.

```
# Step = 648
*** 35 random particles are added ***
Taking action: East
Observation: Orv0rh7
Reward: -1.0
New state: Srv0rh7tv4th5
==================
# Step = 649
*** 35 random particles are added ***
Taking action: North
Observation: Orv0rh7
Reward: -1.0
New state: Srv0rh7tv4th5
```

En este ejemplo el Robot S aun encontrándose en una casilla del límite de la figura, más concretamente la casilla 28, definida por el estado. Podemos observar que la siguiente acción que realiza se sale del tablero. Con la implementación de unas restricciones, implementadas en el Anexo A, en las casillas limite en el método “get_legal_actions” de la clase “model.py” pero no se obtiene éxito tras intentarlos en muchas ocasiones ya que el algoritmo llamaba un número muy alto de veces a esta función y no lo interpretaba de forma correcta.

2.3. Problema de los Anuncios Web

Una vez más, y va a ser la regla a seguir en este proyecto, tendremos que definir el método de parada para el problema de anuncios web descrito en el Anexo B. Implementamos en la clase “pomdp_runner.py” y en concreto en la función “run” un método para parar la simulación como ya hemos hecho anteriormente, pero teniendo en cuenta la acción de anunciar.

2.4. Problema de Vuelo

La detención del problema de las tartas descrito en el Anexo C era indicar la acción final. Este problema termina cuando se decide a que lugar ir.

2.5. Simulación Interactiva, Silenciosa o Benchmark

Se nos presentaba una decisión a la hora de detener el algoritmo que es la posibilidad entre una simulación interactiva o una simulación silenciosa. Solo tuvimos que implementar la simulación silenciosa debido a que la librería usada ya mostraba por defecto en la salida el estado paso a paso.

Al tener dos opciones de alguna forma tenemos que diferenciarlas por lo que decidimos que una fuese implícita con la ejecución normal y la otra por defecto habría que declararla. Procedimos a asignar a la de benchmark como la que había que declararla, para ella añadimos un número atrás que indicase el número de simulaciones deseadas. A continuación, un ejemplo del mismo:

>python main.py pomcp –env Web.POMDP –benchmark X

Para ello, introducimos este nuevo parámetro en la clase”main.py” y en la función “init” de la clase ”RunnerParams.py”. Era necesario crear una nueva función llamada ”runBench” en la clase ”pomdp runner.py” que ejecutase los pasos necesarios para resolver cada situación (de la misma forma que lo hacía la función ”run” de la misma clase) pero esta vez sin mostrar los atributos de cada paso. Además, esta función almacenaría, cuando llegase a un estado final, en dos arrays llamados ”step list” y ”fReward list” el número de pasos y la recompensa acumulada de cada una de las simulaciones para obtener las respectivas desviaciones típicas utilizando la función ”statistics.stdev” de Python. De la misma forma, también almacenaría en los atributos ”steps” y ”fReward” el número total de pasos y la recompensa acumulada de todas las simulaciones con el objetivo de calcular posteriormente los valores medios.

3. Resultados

Muestra de los resultados obtenidos tras ejecutar los siguientes casos para cada problema. La columna ”Silenciosa X” representa el número (X) de simulaciones silenciosas que se han realizado. En esta columna el valor que aparece para la fila ”Pasos” y ”Recompensa” representa el valor total de pasos realizados y la recompensa acumulada total en dicha batería.

3.1. Problema del Tigre

	POMCP		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30
Pasos	4	2	113
Recompensa	7.0	-101.0	107
Valor medio de Pasos			3.76
Valor medio de recompensa			3.56
Desviación típica de pasos			2.17
Desviación típica de recompensa			19.67

	PBVI		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30
Pasos	5	3	112
Recompensa	6.0	8.0	64.0
Valor medio de Pasos			3.73
Valor medio de recompensa			2.13
Desviación típica de pasos			1.87
Desviación típica de recompensa			21.38

3.2. Problema del Tag

Para este problema hemos decidido mostrar tan solo 15 simulaciones silenciosas debido al numeroso tiempo que necesita para alcanzar el estado final.

	POMCP		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 15
Pasos	5	4	1383
Recompensa	6.0	-4.0	.1218.0
Valor medio de Pasos			92
Valor medio de recompensa			-81.2
Desviación típica de pasos			82.79
Desviación típica de recompensa			-82.79

3.3. Problema de los anuncios Web

	POMCP		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30

Pasos	2	2	87
Recompensa	1.9	-4.7	21.5
Valor medio de Pasos			2.9
Valor medio de recompensa			0.71
Desviación típica de pasos			1.49
Desviación típica de recompensa			1.91

	PBVI		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30
Pasos	5	3	76
Recompensa	6.0	8.0	12.4
Valor medio de Pasos			2.53
Valor medio de recompensa			0.41
Desviación típica de pasos			0.82
Desviación típica de recompensa			2.37

3.4. Problema del Vuelo

	POMCP		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30
Pasos	5	4	177
Recompensa	96.0	-4.0	2754.0
Valor medio de Pasos			5.9
Valor medio de recompensa			91.8
Desviación típica de pasos			2.14
Desviación típica de recompensa			18.22

	PBVI		
	Interactiva	Interactiva con 4 Jugadas	Silenciosa 30
Pasos	6	4	73
Recompensa	95.0	-4.0	39.0
Valor medio de Pasos			2.43
Valor medio de recompensa			1.3
Desviación típica de pasos			4.68
Desviación típica de recompensa			316.6

4. Conclusiones

Tras la realización del trabajo completo y a la vista de lo resultado obtenidos, hemos observado que todos los problemas que se ejecutan con el algoritmo PBVI se resuelven más rápido que con el algoritmo POMCP. Sin embargo, podemos apreciar en los resultados que el algoritmo POMCP obtiene una mayor recompensa si ejecutamos un amplio número de simulaciones.

A la hora de hacer pruebas con el problema tag y la creación del cuarto problema nos dimos cuenta de que las acciones las toma en función de la recompensa o reward, cosa que parece lógica, pero nosotros pensábamos que las acciones las tomaba en base a probabilidad partiendo del estado inicial y posteriormente tomaba la reward asociada a dicho movimiento. Podríamos penalizar enormemente los movimientos ”ilegales” del problema del tag de esta forma, pero supone un gasto de tiempo del que no disponemos ya que hay que contemplar para cada estado ”inicial” los posibles movimientos ilegales, todos los estados finales a los que se podría llegar para penalizar enormemente dichos movimientos ilegales. Lo cual implicaría desglosar todas y cada una de esas posibilidades. Teniendo en cuenta la cantidad de estados que tiene el problema del tag, es algo inviable para nosotros actualmente. No obstante, podemos perfeccionar el problema de las islas (diseñado por nosotros) debido a esto, ya que, inicialmente siempre lo resolvía con uno o dos movimientos pero tras darnos cuenta de lo mencionado anteriormente, ha aumentado el número de pasos y por tanto el ”realismo” del problema.

Como conclusión final, podríamos decir que ya que todo este entorno era desconocido para nosotros: tanto el lenguaje de programación, como el uso de librerías de Python y el funcionamiento de estos algoritmos; sentimos que el esfuerzo dedicado a este proyecto nos ha ayudado a ser más creativos en la resolución problemas que nunca antes habíamos afrontado.

5. Referencias

- PyPOMDP. Di, Lu. Data science software engineer at the University of Sydney.

  https://github.com/namoshizun/PyPOMDP

- Pycharm, Jet Brains.

  https://www.jetbrains.com/pycharm

- PyPOMDP. José Manuel Ruiz-Mateos.

  https://github.com/Pruizma/PyPOMDP

- Point-based value iteration: An anytime algorithm for POMDPs. Joelle Pineau, Geoff Gordon and Sebastian Thrun. Carnegie Mellon University

  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.8961&rep=rep1&type=pdf

6. Anexos

Anexo A: Comprobación de Validez

```
if 'Srv' in state:
    # Getting the column and row
    vertical = state[3:4]
    horiz = state[6:7]

    # FIRST ROW
    # If the robot A is in the first row of the map, he shouldn't be able to move to the North
    f_actions = self.actions
    if vertical == 0:
        #del f_actions[0]
        f_actions.remove('North')
        # Cell 26, cant go north nor west
        if horiz == 5:
            #del f_actions[3]
            f_actions.remove('West')
        # Cell 28, cant go north nor east
        if horiz == 7:
            #del f_actions[2]
            f_actions.remove('East')

    # SECOND ROW
    # If the robot A is in the second row may not be able to go west or east, depending on the cell
    if vertical == 1:
        # Cell 23, cant go west
        if horiz == 5:
            #del f_actions[3]
            f_actions.remove('West')
        # Cell 25, cant go east
        if horiz == 7:
```

```
            #del f_actions[2]
            f_actions.remove('East')

    # THIRD ROW
    # If the robot A is in the third row may not be able to go west or east, depending on the cell
    if vertical == 2:
        # Cell 20, cant go west
        if horiz == 5:
            #del f_actions[3]
            f_actions.remove('West')
        # Cell 22, cant go east
        if horiz == 7:
            #del f_actions[2]
            f_actions.remove('East')

    # FOURTH ROW
    # If the robot A is in the fourth row may not be able to go north, west or east, depending on the cell
    if vertical == 3:
        # Cells 10, 11, 12, 13, 14, 18 and 19 can't go north
        if horiz == 0 or horiz == 1 or horiz == 2 or horiz == 3 or horiz == 4 or horiz == 8 or horiz == 9:
            #del f_actions[0]
            f_actions.remove('North')
            # Cell 10, can't go west either
            if horiz == 0:
                #del f_actions[3]
                f_actions.remove('West')
            # Cell 19, can't go east either
            if horiz == 9:
                #del f_actions[2]
                f_actions.remove('East')

    # FIFTH ROW / LAST ROW
    # If the robot A is in the last row of the map, he shouldn't be able to move to the South
    if vertical == 4:
        #del f_actions[1]
        f_actions.remove('South')
        # Cell 0, can't go west either
        if horiz == 0:
            #del f_actions[3]
            f_actions.remove('West')

        # Cell 10, can't go east either
        if horiz == 9:
            #del f_actions[2]
            f_actions.remove('East')

    return f_actions
```

Anexo B: Problema de Anuncios de Web

Disponemos de una web que ofrece anuncios de cosméticos y de cuchillas de afeitar. Esta es visitada tanto por mujeres como hombres. El 70% de los usuarios que visitan nuestra web son hombres y el 30% mujeres. Para poder seguir manteniendo nuestra web necesitamos anunciar el producto adecuado en función del usuario que visite nuestra web. Asumimos que si anunciamos una cuchilla de afeitar a un hombre directamente realiza la compra al igual que sucedería con los cosméticos y las mujeres y en caso de que fallemos, no compraran nuestro producto. Sin embargo, no conocemos realmente que perfil está visitando nuestra web y tendremos que observar las consultas que realizan,

que pueden ser búsquedas de páginas sobre ballet o sobre videojuegos respectivamente para acertar con nuestro anuncio.

Anexo C: Problema de Vuelo

El capitán de un vuelo se enfrenta a un problema, el motor izquierdo del avión se ha sobrecalentado lo que le obliga a tener que hacer un aterrizaje para dejar enfriar el motor. Las únicas opciones a las que puede llegar son tres ciudades las cuales una dispone de una pista de aterrizaje que dispone de un aparato que enfría el motor al instante, en otra hay pista de aterrizaje, pero tendrían que esperar a que el motor se enfriase por su cuenta y por último esta la ciudad que no tiene pista de aterrizaje y obligaría al capitán a realizar un aterrizaje forzoso y muy arriesgado. Tienes que decidir donde aterriza el avión, pero no sabes cuál es la mejor ciudad así que puedes ayudarte del capitán que si puede verlas y así tomar la opción más segura.