Proyecto de Simulación: Agentes

Adrian González Sánchez C412
https://github.com/adriangs1996

November 30, 2020

Orientación

Se tiene un ambiente, representado por un tablero rectangular de tamaño N×M, con el que interactúa un agente. Las acciones que realizan los agentes ocurren por turnos. En un turno, los agentes realizan sus acciones, una sola por cada agente, y modifican el medio. En el siguiente, el ambiente puede variar. Si es el momento de cambio del ambiente, ocurre primero el cambio natural del ambiente y luego la variación aleatoria. En una unidad de tiempo ocurren el turno del agente y el turno de cambio del ambiente. Los elementos que pueden existir en el ambiente son obstáculos, suciedad, niños, el corral y los agentes que son llamados Robots de Casa. El Robot de Casa parte de un posición aleatoria y es el el que realiza el primer turno. Igual, se especifica el valor del tiempo de unidades de cambio (t).

El objetivo del Robot de Casa es mantener la casa (a.k.a el ambiente) limpia. Se considera la casa limpia si el 60% de las casillas vacías no están sucias. Se sabe que si la casa llega al 60% de casillas sucias el Robot es despedido e inmediatamente cesa la simulación. Si el Robot ubica a todos los niños en el corral y el 100% de las casillas están limpias también cesa la simulación. Estos son llamados estados finales. En caso de que no se logre uno de los estados finales del ambiente, la simulación debe detenerse cuando hayan transcurrido 100 veces t. Debe programar el comportamiento del robot por cada turno así como las posibles variaciones del ambiente.

Consideraciones

El ambiente es una matriz de casillas, y los posibles elementos que pueden existir en las casillas son: robot, niño, obstáculo, suciedad y corral. Se permite también que coexistan varios elementos en una misma casilla, por ejemplo, pueden existir un niño en un corral, el robot puede estar en una misma casilla con un niño y un corral, el robot puede coexistir con una basura, y puede que el robot cargue un niño y en ese momento se encuentre en una casilla con una basura o una suciedad. El ambiente es dinámico y el robot en todo momento tiene acceso a la información completa sobre el ambiente.

Dada la naturaleza dinámica del ambiente, el agente que se implementa es reactivo; aprovecha el conocimiento del ambiente para tomar una decisón de qué hacer en cada turno. La idea es que el robot nunca vaga sin objetivo por el ambiente, siempre tiene un propósito, o bien intenta reducir la suciedad del ambiente, o bien intenta llevar a un niño al corral. Exactamente cuándo limpia y cuándo lleva un niño, es la diferencia de implementación del agente.

El ambiente tiene 3 estados:

1. Menos del 60% de las casillas vacías están sucias, o hay algún niño fuera de corral.

2. Mas del 60% de las casillas vacías están sucias (Estado final).
3. No hay suciedad y todos los niños están en el corral (Estado final).

El robot prioriza nunca llegar al estado 2, pues en este caso se considera como fallo, aunque también intenta llegar al estado 3, pues es el caso ideal de su funcionamiento.

Agente: Always deliver

Este agente intenta llegar lo más rápido posible al estado 3, intentando de forma greedy llevar los niños al corral, mientras la suciedad no sobrepase un umbral definido. Este robot una vez que carga un niño, no lo suelta hasta que no lo ubica en un corral.

1. Si carga niño y está en un corral $\implies$ soltar al niño.
2. Si carga un niño $\implies$ moverse en dirección al corral más cercano.
3. Si no carga un niño y está en una celda con suciedad $\implies$ limpiar.
4. Si suciedad $<$ threshold y en celda con niño suelto $\implies$ coger al niño.
5. Si suciedad $<$ threshold $\implies$ moverse en dirección al niño más cercano.
6. Si suciedad $\geq$ threshold $\implies$ moverse en dirección a la suciedad más cercana.

Agente: Try to deliver if not dirty

Este agente utiliza reglas muy parecidas al anterior, la diferencia es que una vez que toma una decisión, esta puede variar, o sea, si carga un niño, y en su camino al corral, el nivel de suciedad sobrepasa el umbral, entonces suelta al niño, e intenta limpiar; quiere decir que este prioriza mantenerse en el estado 1 y si puede, intenta llegar al estado 3. Visto de un modo simplista, este robot intenta no perder, mientras que el otro intenta siempre ganar.

1. Si suciedad $\geq$ threshold y carga niño $\implies$ soltar niño.
2. Si suciedad $\geq$ threshold y suciedad en celda actual $\implies$ limpiar.
3. Si suciedad $\geq$ threshold $\implies$ moverse en dirección a la basura más cercana.
4. Si carga niño y está en un corral $\implies$ soltar al niño.
5. Si suciedad $<$ threshold y en celda con niño suelto $\implies$ coger al niño.
6. Si suciedad $<$ threshold $\implies$ moverse en dirección al niño más cercano.

Implementación

La simulacion consta de 3 módulos: **environment.py, robot.py, house.py**. El ambiente se representa en la clase Environment, el cual recibe parámetros como la cantidad de niños, porciento de obstáculos, porciento de suciedad y dimensiones. Este ambiente se genera automáticamente y de forma aleatoria, garantizando en cada momento que se cumplan las restricciones del ambiente. Esta inicialización es básicamente el llenado de una matriz de celdas cuyo contenido es especificado en la clase CellContent, que es un enum con los posibles valores que puede tomar una celda.

Además el ambiente es responsable de emular sus propios cambios, o sea, mover los niños, generar la suciedad, etc. Y cada cierto período de tiempo, todo se reordena utilizando el mismo método

de inicialización aleatorio. El ambiente es capaz de reportar la posición de todos sus elementos, dígase robot, niños, basuras, corrales, etc.

El agente, o en este caso el robot, se implementa con la clase **Robot**, la cual es la encargada de encapsular las posibles acciones que puede tomar el robot (moverse, soltar un niño, cargar un niño, limpiar). Principal en esta clase es el método **evalEnvironment** el cual percibe el ambiente y decide que plan seguir; es este método el que distingue a un agente de otro.

la simulación ocurre en el módulo house.py, donde se corren con el método **testRobot** simulaciones sobre un robot específico.

Resultados

Luego de correr las 30 simulaciones en 10 escenarios, obtenemos los siguientes promedios para cada agente:

Table 1: Agentes

Agente	**Éxito**	**Despido**	**Normal**
1	48	6	46
2	4	15	83

Este resultado es intuitivo, por la forma en que se disponen las reglas, el agente 2 siempre intenta permanecer en un estado neutral, mientras que el primero es más osado, intentando llevar los niños a su destino. También se pudo comprobar que el primer agente es mucho más efectivo cuando el entorno es grande y la cantidad de niños desminuye, pues cada niño que el robot logre cargar es uno menos ensuciando, y al ser un entorno grande, el porciento de suciedad que aumenta en cada turno es pequeño, lo que hace la estrategia greedy viable. Por otro lado, en ambientes con muchos niños, o en entornos pequeños, es preferible el agente más consevador, que si bien es cierto que no llega con frecuencia al estado final, también se mantiene en un promedio del 83% de los casos en un estado neutral, mientras que el número de despidos del agente 1 aumenta.

En un ambiente altamente dinámico como este, al parecer el mejor modelo sería un híbrido de estos agentes, donde se tenga en cuenta a la hora de planear una estrategia, el porciento de basura que representa recoger a un niño en una posición determinada, y por supuesto, como asumimos que el movimiento de los niños es aleatorio, no podemos hacer predicciones precisas de hacia donde se dirigen, por lo que un modelo de este tipo debería lidiar con decisiones diferentes en cada turno y potencialmente nunca llevar a cabo ningún plan. Quizás una mejor idea sea hacer esta valoración cuando el robot carga el niño, y en ese momento decidir no solo en base al nivel de suciedad actual, sino tambien al posible aumento de suciedad en su trayecto hacia el corral más cercano.