Universidad de Granada

decsai.ugr.es

Práctica 2 de Inteligencia Artificial
Agentes Reactivos – El Robot trufero

Por:

Izquierdo Vera, Javier - B2
javierizquierdovera@gmail.com

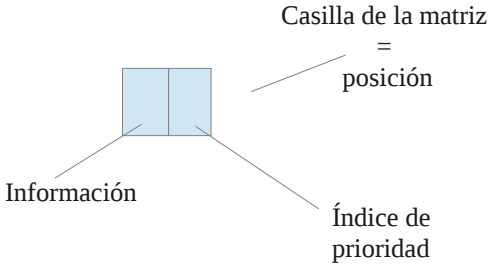
DECSAI
Departamento de Ciencias de la Computación e Inteligencia Artificial

0. Índice

1. Descripción de la solución planteada

1.1 Representación interna

La solución para el problema planteado del robot trufero consiste en un agente reactivo con memoria que utiliza una representación icónica, y algunos esquemas de subsunción.
El agente utiliza una estructura de datos basada en una matriz de pares, de modo que el agente va almacenando toda la información que percibe y va representando su propio modelo del mundo. Cada casilla de la matriz es una posición en el mundo, posición que es ocupada por el agente en los instantes de tiempo, y de modo que el primer valor del par nos indica qué conocemos acerca de esa posición y el segundo valor un índice de prioridad.



Los valores que toma la casilla de información son:

- 0 → Obstáculo
- 1 → Desconocido
- 2 → Camino
- 5 → Trufa
- 6 → Posición actual del agente

El índice de prioridad consiste en un valor entero que aumenta en función de la posición que vaya ocupando el agente, cada casilla que dejamos atrás tiene un valor de un entero menos que la nueva posición. De este modo el agente es capaz de reconocer las casillas que lleva más tiempo sin revisar.
Los valores que toma la casilla de prioridad son:

- -2 → Alerta →Este valor lo toman las casillas, en las que el agente conoce que se encuentra una trufa, cuando superan un valor de instantes de tiempo + nivel aromático * 2. De este modo el agente conoce que en esa posición se necuentra una trufa con cierto nivel estimado y puede llegar a situación crítica. (Esta opción se ha descartado en la práctica, pero se puede comprobar estableciendo valores en los umbrales de los defines).
- -1 → Por explorar → Prioridad con la que se inician las casillas en la representación interna. De este modo el agente se ve tentado a explorar el mundo y a lograr una temprana representación del mismo.
- N >= 0 → Camino → Prioridad que se le asigna al camino válido.
- 9999999 → Ignorar → Prioridad mínima, el agente no se fijará en esta casilla. Esta prioridad se le asigna a los obstáculos.

También se hace uso como método de representación interna de dos maps de la STL. “trufas” y “camino_pisado”. Se ha utilizado este tipo de dato para así poder acceder a los datos utilizando como clave las coordenadas.

- Trufas → Guarda la posición de las trufas, de modo que el agente trata de estimar el crecimiento de las trufas con el paso de los instantes de tiempo, de este modo se pueden fijar niveles de alerta y críticos de las trufas.
- Camino_pisado → Sirve para memorizar el camino que el agente utiliza para llegar hasta un objetivo

fijado, sirve de guía para no perderse.

Además de esto, el agente también guarda información acerca del último movimiento, de la orientación, del objetivo,... Todo esto sirve de ayuda para el movimiento.

1.2 Patrón de movimiento

Como se mencionaba anteriormente, el agente se mueve hacia las casillas con mayor prioridad, estas son las marcadas como objetivo, a la que va ignorando todo lo demás; la alerta de trufa, a la que va con preferencia si se encuentra a la distancia de una casilla; las casillas desconocidas, a las que prefiere ir si lo demás es camino u obstáculos, para de este modo explorar el mundo lo más pronto posible; el camino, al que da prioridad sobre los obstáculos; y los obstáculos, a los cuales le da la mínima prioridad, de modo que los ignora, nunca eligiéndolos como camino si los conoce. Usa esta estructura de subsunción para el movimiento.

Cuando el agente choca, antes de establecer la siguiente acción, resta la posición X o Y correspondiente a la última acción, de modo que la representación siga siendo la correcta y se marque la casilla del obstáculo con los valores que le corresponden.

El agente comienza a rastrear una vez que se supera comienzo_rastreo (para esta práctica fijado en 40, pero se puede definir en cambiando el valor del define), de este modo al principio no se preocupa por las trufas ya que aún no hay muchas o tienen poco nivel olfativo, dándole más importancia a explorar el mapa.
El agente mantiene una cuenta de los pasos que realiza, es decir, de las casillas a las que avanza. El agente se basa en esto para alternar avance con rastreo, de modo que rastrea y guarda información sobre las trufas de todas las casillas una vez comience a rastrear.

En el caso de marcar una casilla como objetivo, el agente trata de ir hacia ella ignorando todo lo demás. El nivel con el que el agente fija un objetivo, para esta práctica, se ha fijado muy alto, de modo que nunca llega a considerar una trufa como objetivo basándose en los instantes de tiempo. Esto es debido a que, tras realizar varias pruebas, se ha llegado a la conclusión de que el agente es capaz de recoger más trufas sin fijar objetivos críticos, ya que el mapa es pequeño y la frecuencia con la que aparecen es elevada. Esta funcionalidad se utiliza cuando el agente rastrea una trufa y detecta que su nivel olfativo es mayor que el valor entero asociado a “extractNow”, en ese caso fijo como objetivo la extracción inmediata de esa trufa.
Todos estos umbrales son configurables desde los defines de agent.cpp, de modo que se pueden realizar pruebas para distintas configuraciones del agente.
Si hay un objetivo fijado, puede haber dos posibilidades, que esté en él, o que tenga que avanzar hasta él. Si está en el objetivo, resetea los valores, elimina la trufa del map de trufas, y la extrae. Por el contrario, avanza hasta ella siguiendo otro patrón de movimiento.
El patrón de movimiento cuando se fija un objetivo se basa en otra estructura de subsunción dividida en cuatro partes:

- Primero comprueba si cierto avance está recomendado, esto es que se produzca una disminución en la diferencia de las coordenadaas que lo separan del objetivo, teniendo en cuenta que la casilla a la que se mueve sea correcta y no haya intentado ya ir por ella para este objetivo.
- Si nada de lo anterior es posible, comprueba si algún camino es válido y no pisado, aunque no se produzca mejora.
- Si nada de lo anterior es posible, trata de avanzar a una casilla válida siempre y cuando no sea por la que ha venido, pero sin tener en cuenta que ya haya podido estar. También se establece un límite fijado por “retirada” (en este caso con valor 10), el cual establece el máximo de veces que se puede tratar de buscar solución de este modo sin encontrar un camino adecuado, en el caso de que se supere y no se produzca mejora, se pasa al siguiente nivel.
- En este caso lo que trata es de alejarse para buscar el camino en otro lugar, ya que se ha producido una situación en el que se ha quedado atrapado sin encontrar un camino adecuado. En este nivel trata de alejarse, siguiendo para ello una dirección válida en la que no se esté chocando y no sea por la

que venía.

El agente mantiene información en todo momento de su posición actual, de su orientación, de su último movimiento, y de las prioridades y estimaciones que se han considerado, para poder realizar todo lo anterior.

1.3 Funciones

El agente cuenta con una serie de funciones para realizar distintas operaciones para estimaciones y prioridades, así como para proporcionar información de depuración.

- ActionType Arriba(); → Sea cual sea la posición del agente o la orientación, el agente se mueve arriba.
- ActionType Abajo(); → Sea cual sea la posición del agente o la orientación, el agente se mueve abajo.
- ActionType Derecha(); → Sea cual sea la posición del agente o la orientación, el agente se mueve a la derecha.
- ActionType Izquierda(); → Sea cual sea la posición del agente o la orientación, el agente se mueve a la izquierda

Las anteriores funciones, muchas veces no se pueden realizar en un solo tiempo, ya que pueden requerir giros. Por ello, mientras una de las acciones está en proceso, no se establece ningún otro movimiento y se devuelve el control a la función concreta. Esto facilita crear patrones de movimientos.

.

- void FijarObjetivo(int posXob, int posYob); → Establece un objetivo, para realizar con él las operaciones que se han mencionado anterioremente.
- void CompletarRastreo(); → Si acaba de rastrear, guarda el nivel aromático de la trufa en el map trufas, multiplicándolo por 2, y guarda la casilla en la que se encuentra la trufa. Aquí es donde se fija como objetivo si se supera el umbral extractNow. En el caso contrario, es un camino (ya que solo se accede a esta función si no ha chocado en la última acción), por lo tanto lo establece como tal, y actualiza la prioridad para tener en cuenta la antiguedad con la que ha sido comprobada la posición.
- void Revisar(); → Es la función que hace estimaciones para el nivel crítico o de alerta. Para ello recorre el map de trufas y aumenta en uno el valor tras instante de tiempo. De modo que cuando alcanza los umbrales establecidos en los defines, les da la prioridad adecuada o marca como objetivo. En el caso de que la trufa no sea crítica ni alerta, le actualiza el nivel de prioridad como si acabasemos de pasar por ella, ya que acaba de ser revisada.

Las siguientes son información para la depuración, y para poder ver los cálculos que realiza el agente.

- void Memoria(); → Muestra toda la representación del mundo en la consola.
- void Orientacion(); → Muestra información sobre la orientación del agente en la consola.
- void whereAreU(); → Marca en el mapa la posición actual del agente.

2. Resultados obtenidos por la solución aportada en los distintos mapas que se entregan con el agente

extractNow = 8

Mapa	Probabilidad → 0.01	Probabilidad → 0.02
Agent.map	1338	2810
Mapa1.map	1494	3114
Mapa2.map	1458	3030
Mapa3.map	1540	2890

extractNow = 10

Mapa	Probabilidad → 0.01	Probabilidad → 0.02
Agent.map	1358	2854
Mapa1.map	1530	3094
Mapa2.map	1440	2960
Mapa3.map	1526	2862

extractNow = 12

Mapa	Probabilidad → 0.01	Probabilidad → 0.02
Agent.map	1552	2726
Mapa1.map	1500	3026
Mapa2.map	1422	2988
Mapa3.map	1534	2742

Seleccionado → extractNow = 8

3. Descripción de otras estrategias descartadas

Cambiando los parámetros establecidos en los defines de agent.cpp se pueden establecer distintas estrategias. Algunas estrategias descartadas son:

- Fijar objetivo cuando la trufa lleva tiempo en memoria → Esta estrategia ha sido descartada para la práctica. Al tratarse de un mapa pequeño y de una constante aparición de trufas, fijar un objetivo puede llevar a perder instantes de tiempo valiosos, por lo que el número de trufas recolectadas es menor si se establece un umbral en el que se fija el objetivo en base a las estimaciones.
- Marcar trufas con nivel de alerta → Al dejar que el agente recolecte únicamente las que superan cierto nivel aromático, sin recurrir a intentar recoger las que puedan haber alcanzado un cierto nivel de crecimiento, el número de recolecciones es mayor. Una vez más, esto es debido a que el mapa no es lo suficiente grande, y el sistema de movimiento le permite volver a llegar a esas casillas antes de que la trufa se pierda.
- Otras estrategias se pueden basar en modificar el umbral con el que inicia el rastreo, o a partir del cual extrae directamente la trufa. Los umbrales actuales son los que mejores resultados han producido.

La conclusión principal de esta práctica, al menos en mi caso, es que la simplicidad es mejor. El mayor número de trufas se obtiene descartando en los defines las funciones comentadas.