Relacion-1-Ia.pdf

Anónimo

Inteligencia Artificial

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

ugr | Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial

DECSAI

Curso 2022/23

INTELIGENCIA ARTIFICIAL

Relación de Problemas 1

AGENTES REACTIVOS

1. Una hormiga artificial vive en un mundo bidimensional cuadriculado y desarrolla un comportamiento que le permite seguir un rastro de feromonas a lo largo de un conjunto de casillas previamente marcadas (el tamaño del rastro es de una casilla). La hormiga ocupa una sola casilla y puede encarar las casillas que se encuentran arriba, a la derecha, a la izquierda y debajo de la posición en la que se encuentra. La hormiga puede llevar a cabo tres acciones: moverse a una celda hacia adelante (actFORWARD), girar a la izquierda permaneciendo en la misma casilla (actTURN_L) y girar a la derecha permaneciendo en la misma casilla (actTURN_R). La hormiga puede percibir si la casilla que tiene delante (en el sentido del movimiento) tiene feromona.

a) Especificar un sistema de reglas para controlar el comportamiento de la hormiga en el seguimiento del rastro de la feromona. Suponer inicialmente a la hormiga en una casilla en la que puede percibir el rastro de feromona.

b) Resuelva el problema anterior con la siguiente restricción: la hormiga, una vez que llega a una nueva casilla, no puede girar más de 180 grados desde la posición inicial.

2. La avispa hembra del género Sphex, deja sus huevos dentro de un grillo que ha paralizado y ha llevado a su nido. Las larvas de la avispa salen del grillo y se alimentan de él. La avispa presenta el siguiente comportamiento: lleva el grillo paralizado a su nido, lo deja en el umbral del nido, entrar dentro del nido para ver si todo está correcto, sale, y entonces arrastra al grillo hacia su interior. Si el grillo se mueve cuando la avispa está en el interior haciendo la inspección preliminar, la avispa saldrá del nido, volverá a colocar el grillo en el umbral, pero no dentro, y repetirá el procedimiento de entrar en el nido para ver si todo está correcto. Si el grillo se mueve otra vez mientras la avispa está dentro del nido, ésta volverá a salir y colocar el grillo en el umbral, entrando de nuevo en el nido para realizar la inspección preliminar. En una ocasión, este procedimiento se repitió cuarenta veces. Define características y acciones para diseñar un agente reactivo que se corresponda con el comportamiento de la avispa.

Ejercicio 1:

a) Se haría sin la variable N_giros

.cpp

b)
```
Agent::ActionType Agent::Think() {
    ActionType accion = actIDLE;
    /* Reglas */
    if (FEROMONA){ // si hay feromona, vamos a por ella
        accion = actFORWARD;
        N_giros = 0;
    }
    else if (N_giros < 1){ //izquierda
        accion = actTURN_L;
        N_giros++;
    }
    else { // derecha
        accion = actTURN_R;
        N_giros ++;
    }
    return accion;
}
```

.h
```
class Agent {
public:
    Agent () {
        FEROMONA = false;
        N_giros = 0;
    }
private:
    bool FEROMONA;
    int N_giros;
};
```

Ejercicio 2:

Perceptores:

Sensor 1 → La avispa tiene o no el grillo
Sensor 2 → Si el grillo está o no en el umbral
Sensor 3 → Si el nido está o no bien
Sensor 4 → Si la avispa está o no en el umbral

Objetivo:

Meter al grillo en el nido

Memoria:

La avispa dice si el nido esta bien o no

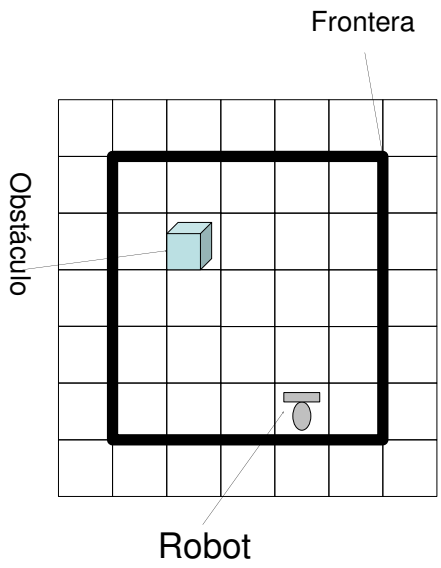
Acciones

// ~~Agarra → coger el grillo~~
Deja → Dejan al grillo
va_umbral → Ir al umbral
va_fuera → Fuera del nido
va_dentro → dentro del nido
encuentra → encontrar grillo

3. Supongamos que un agente trabaja sobre un tablero formado por NxN casillas. Sobre este tablero se definen dos zonas: una "zona interior" formada por un tablero de (N-2)x(N-2) casillas inscrito en el tablero general, y una "zona exterior" formada por el resto de las casillas.

Separando ambas zonas aparece una línea gruesa negra denominada *"Frontera"*. En la figura se muestra un ejemplo de la configuración de un tablero 7x7.

El cometido del robot consiste en llevar todos los obstáculos que se encuentren en la zona interior a la zona exterior. El robot siempre se debe encontrar en la zona interior, y no debe nunca traspasar la frontera.



Para realizar esta tarea, el robot dispone de 3 sensores, un sensor de choque "BUMPER" que le permite detectar el obstáculo, un sensor de infrarrojos "CNY70" que permite ver dónde está la línea de la Frontera, y una brújula digital "Brujula" que le indica su orientación en el avance. Los dos primeros sensores se encuentran situados en la parte frontal del robot. La brújula sólo devuelve 4 valores: 0, 1, 2 y 3, representando respectivamente Norte, Este, Sur y Oeste.

Las acciones que puede realizar el robot son las siguientes:

1. Avanzar: Avanza una casilla en la dirección que marca su brújula siempre que no tenga un obstáculo delante.
2. Retroceder: Retrocede una casilla en la dirección contraria a la que indica su brújula, siempre que no tenga un obstáculo detrás.
3. GirarI: Gira sin moverse de la casilla hacía la izquierda.
4. GirarD: Gira sin moverse de la casilla hacía la derecha.
5. Nada: No realiza ninguna acción
6. Empujar: Avanza una casilla en la dirección que marca su brújula. Para que está acción tenga efecto, debe estar activado el sensor de choque.

Se pide:

a) Definir las variables de estado (nombre e descripción) y las reglas de producción necesarias para diseñar un agente reactivo con memoria que partiendo de una casilla desconocida dentro de la zona interior de un tablero de dimensiones también desconocidas (nunca superiores a 99x99), sea capaz de calcular la dimensión de la zona interior, suponiendo que en el tablero no hay obstáculos.
b) Definir las variables de estado y las reglas de producción necesarias que permitan al robot localizar el obstáculo en el tablero.
c) Suponiendo que el robot se encuentra orientado hacía el obstáculo en una casilla adyacente (es decir, el sensor BUMPER está activado) y que el obstáculo se encuentra en una casilla interna del tablero que no es adyacente con ninguna casilla pegada a la frontera, definir las variables de estado y las reglas de producción necesarias que permitan al robot expulsar el obstáculo hacía la zona exterior, arrastrándolo por el camino más corto de casillas.

Ejercicio 2:

pierdo espacio

Necesito concentración

.h

```
class Agent {
public:
    Agent() {
        CNY70 = false;
        BUMPER = false;
        //variables de estado
        HeEstadoFrontera = 0;
        giran = 0;
        n_casillas = 0;
    }
    enum ActionType {
        actFORWARD ...
        // todas las que haya
    };
    void Percept (const Environment &env);
    ActionType Think ();
private:
    bool CNY70;
    bool BUMPER;
    int HeEstadoFrontera;
    int giran;
    int n_casillas;
};
```

.cpp

```
Agent::ActionType Agent::Think()
{
  ActionType accion = actIDLE;

/* ESCRIBA AQUI SUS REGLAS */
```

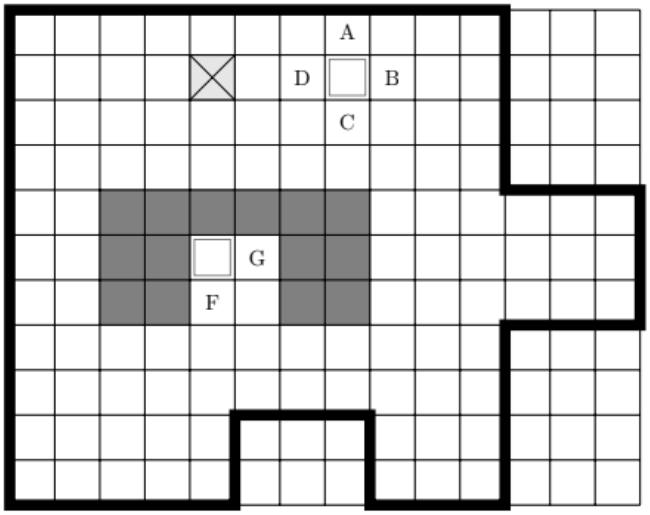
ali ali oooh esto con 1 coin me lo quito yo...

```
if(BUMPER_) {
        cout << "Regla 1: Obstaculo encontrado\n";
        accion=actIDLE;
}
else if(girando == 1 and CNY70_ and HeEstadoEnLaFrontera%2==0) {
         cout << "Regla 7: Estoy en una esquina girando a izquierda\n";
          accion = actTURN_L;
          girando = 0;
}
else if(girando == 1 and CNY70_ and HeEstadoEnLaFrontera%2==1) {
          cout << "Regla 2: Estoy en una esquina girando a derecha\n";
          accion = actTURN_R;
           girando = 0;
}
else if(girando == 1) {
         cout << "Regla 2: Avanzo en mitad de un giro\n";
        accion = actFORWARD;
        girando = 2;
}
else if(girando ==2 and HeEstadoEnLaFrontera%2==0){
        cout << "Regla 3: Terminar giro a la izquierda\n";
         accion = actTURN_L;
         HeEstadoEnLaFrontera++;
          girando =0;

}
else if(girando ==2 and HeEstadoEnLaFrontera%2==1){
       cout << "Regla 4: Terminar giro a la derecha\n";
       accion = actTURN_R;
      HeEstadoEnLaFrontera++;
       girando =0;

}
else if(CNY70_ and HeEstadoEnLaFrontera%2==0){
         cout << "Regla 5: Frontera par, empiezo giro a la izquierda\n";
         accion = actTURN_L;
        girando = 1;
}
else if(CNY70_ and HeEstadoEnLaFrontera%2==1){
      cout << "Regla 6: Frontera impar, empiezo giro a la derecha\n";
       accion = actTURN_R;
       girando = 1;
}
else{
       cout << "Regla 4: Avanza\n";
      accion = actFORWARD;
}


return accion;
```

4. Supongamos que tenemos un robot sobre un mapa bidimensional discreto de tamaño NxM. El robot puede realizar las acciones de Avanzar y Girar en el sentido de las agujas del reloj. El robot posee un sistema de posicionamiento sobre el mapa que le devuelve sus coordenadas absolutas "(robotX, robotY)" dentro del mapa.

Suponiendo que en el mapa hay obstáculos fijos (paredes), y que el robot se encuentra ubicado dentro de ese mapa en una posición concreta, definir un comportamiento reactivo para el mismo que le permita desplazarse hasta una coordenada objetivo "(ObjX, ObjY)". Para ello, definir las variables de estado necesarias y el sistema de reglas de producción que reproducen el comportamiento requerido.

5. Idear una función de potencial artificial (con componentes repulsivos y atractivos) que pueda ser utilizada para guiar un robot desde cualquier casilla del mundo bidimensional cuadriculado de la figura siguiente, a la casilla objetivo que está marcada con una X (suponer que las posibles acciones que puede ejecutar el robot son ir al norte, sur, este y oeste). ¿Tienen las componentes repulsivas y atractivas algún mínimo local? Si es así, ¿dónde?



Ejercicio 4:

.h

```
class Agent
{
public:
    Agent(int x, int y){
        RobotX_ = 0;
        RobotY_ = 0;
        OldRobotX_ = -1;
        OldRobotY_ = -1;
        ObjX_ = x;
        ObjY_ = y;
        Orientacion_ = -1;

      for (int i = 0; i < 10; ++i)
          for (int j = 0; j < 10; ++j)
              m[i][j] = 0;
}

enum ActionType
{
  actFORWARD,
  actTURN,
  actIDLE
};

void Perceive(const Environment &env);
ActionType Think();

private:
   int RobotX_, RobotY_, ObjX_, ObjY_, OldRobotX_, OldRobotY_, Orientacion_;

   int m[10][10];
};

string ActionStr(Agent::ActionType);
```

.cpp

```
Agent::ActionType Agent::Think() {
int accion = 0;
//reglas
return static_cast<ActionType>(accion);
```

pierdo espacio

Necesito concentración

ali ali oooh esto con 1 coin me lo quito yo...