Relacion1.pdf

Sanchez01

Inteligencia Artificial

2º Grado en Ingeniería Informática

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

Relación de Problemas 1 - Agentes Reactivos

Curso 2023/2024

Ejercicio 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.

a) Necesitamos una variable booleana FEROMONA que vale 1 si hay hormona, y 0 si no hay.
El sistema de producción sería:

```
FEROMONA → actF
1        → actTURN
```

b) La hipótesis es la misma, pero no puede girar más de dos veces en la misma dirección. Se podría solucionar con un contador de giros, ¿pero como se hace con nuestro código?¿Como se contempla el giro? Con memoria, necesitamos retroalimentación, que recuerde qué ha hecho en la última iteración.

Entonces, tendremos dos variables:
FEROMONA → percepción
GIRO → memoria

El sistema de producción sería:

```
FEROMONA → actF, GIRO =0
GIRO=0   → actTR, GIRO++
1        → actTL
```

Lo que se ha hecho serían los sistemas de producción. Ahora vamos a ver el código.

Para el apartado a, el fichero agent_hormiga.hpp sería:

```
#ifndef AGENT__
#define AGENT__

#include <string>
#include <iostream>
using namespace std;

// ------------------------------------------------------------
//              class Agent
// ------------------------------------------------------------
class Environment;
class Agent {
public:
   Agent(){
       FEROMONA_=false;
   }

   enum ActionType {
       actFORWARD,
       actTURN_L,
       actTURN_R,
       actIDLE
   };

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

private:
   bool FEROMONA_;
};

string ActionStr(Agent::ActionType);

#endif
```

Para el apartado a, el fichero agent_hormiga.cpp sería:

```
#include "agent_hormiga.h"
#include "environment.h"
#include <iostream>
#include <cstdlib>
#include <vector>
#include <utility>

using namespace std;
Agent::ActionType Agent::Think() {
   int accion = 0;
   if(FEROMONA_) accion = actFORWARD;
   else accion = actTURN_R;

   return static_cast<ActionType> (accion);
}
// -----------------------------------------------------------
void Agent::Perceive(const Environment &env) {
   FEROMONA_ = env.isFeromona();
}
// -----------------------------------------------------------
string ActionStr(Agent::ActionType accion) {
   switch (accion) {
   case Agent::actFORWARD: return "FORWARD";
   case Agent::actTURN_L: return "TURN LEFT";
   case Agent::actTURN_R: return "TURN RIGHT";
   case Agent::actIDLE: return "IDLE";
   default: return "????";
   }
}
```

Para el apartado b:

```
Agent::ActionType Agent::Think() {
   int accion = 0;

   if(FEROMONA_){
       accion =actFORWARD;
       GIRO180_=0;
   }else if(GIRO180_ == 0){
       accion = actTURN_R;
       GIRO180_++;
   }else accion =actTURN_L;

   return static_cast<ActionType> (accion);
}
```

Ejercicio 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.

Características del agente	Acciones del agente
- posición de la avispa: dentro/fuera del nido - posición del grillo: en el umbral/no umbral - estado del nido: inspeccionado/ no inspeccionado	- llevar grillo al umbral - entrar/salir del nido - arrastrar al grillo

Sistema de producción:

posicion_avispa == “fuera” && posicion_grillo == “umbral” → entrar_al_nido()
posicion_avispa == “dentro” && posicion_grillo == “umbral” → salir_nido()
posicion_avispa == “fuera” && inspeccion == “correcta” → arrastrar_grillo()
posicion_avispa == “dentro” && posicion_grillo != “umbral” → colocar_grillo_umbral()
posicion_avispa == “fuera” && posicion_grillo != “umbral” → colocar_grillo_umbral()

Ejercicio 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.

a)

Requiere 3 variables de memoria: para poder hacer el giro, para saber cuándo girar, para girar.

Una regla por defecto sería avanzar. Cuando he llegado por primera vez al límite hay que activar el giro. Una vez hecho eso, comienzo a avanzar y activo la variable de conteo.

Al profesor le han salido 5 reglas.

El fichero .hpp:

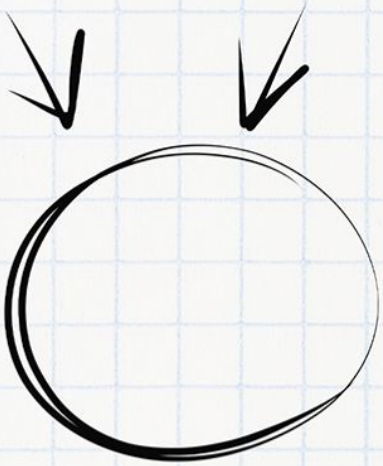
```
class Environment;
class Agent
```

Imagínate aprobando el examen

Necesitas tiempo y concentración

Planes	PLAN TURBO	PLAN PRO	PLAN PRO+
Descargas sin publi al mes	10	40	80
Elimina el video entre descargas	✓	✓	✓
Descarga carpetas	✗	✓	✓
Descarga archivos grandes	✗	✓	✓
Visualiza apuntes online sin publi	✗	✓	✓
Elimina toda la publi web	✗	✗	✓
Precios Anual	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo, ¿Qué nota vas a sacar?



WUOLAH

```
{
public:
    Agent(){
        CNY70_=false;
        BUMPER_=false;
        size_=0;
        count_=false;
        turn_=false;
    }
    enum ActionType
    {
        actFORWARD,
        actTURN_L,
        actTURN_R,
        actBACKWARD,
        actPUSH,
        actIDLE
    };

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

El fichero .cpp:

```
// ----------------------------------------------------------------------------------------------------
Agent::ActionType Agent::Think()
{
    int accion =0;
    if(turn_){
        accion = actTURN_R;
        turn_   =false;
        count_=false;
        size_  =1;
    }
    else{
        if( !CNY70_){
            accion = actFORWARD;
            if(count_)
                size_++;
        }
        else{
            if(count_){
                accion = actIDLE;
                cout << “Tamaño: “ << size_ + size_ << endl;
            }
            else{
                accion = actTURN_R;
                turn_=true;

            }
        }
    }
}
```

b) Encontrar un objeto

Idea política: hacer un barrido pero que **no** sea aleatorio.

```
BUMPER                  → actIDLE;
!BUMPER && !CNY70_  → actFORWARD;
!BUMPER && CNY70_   → actTURN_L;
```

c) Expulsar el objeto

```
BUMPER  &&  !CNY70_     → actPUSH;
BUMPER  &&  CNY70_      → actTURN_R;
!BUMPER &&  CNY70_      → actFORWARD;
```

Ejercicio 4

Está resuelto en Telegram

Ejercicio 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?

Mi agente en función de lo que conozca hace su parte.

Componente atractiva: $p_a(X) = k_1 d(X)^2$

Componente repulsiva: $p_r(X) = \frac{k_2}{d_0(X)^2}$

Potencial = $p_a(X) + p_r(X)$

Potencial en A sale a 16, que es la que más le aleja. B y C son parecidos. La mejor es la D.
En la casilla donde está el robot es 17.
Si calculamos F y G sale >20