Resolución Hoja de Ejercicios y Problemas tipo examen

Ismael Sagredo Olivenza

Problema 01

1 REAS

Sensores:

- Temperatura => T: real
- Personas => P: bool
- Animales => A: bool
- Hora => H: Dia|Noche = (0,1)
- VentanaAbierta => V: bool

Actuadores:

- Calefacción => Cal:bool
- Aire Acondicionado => AA:bool
- Alarma => Alam:bool
- Modificar Temperatura objetivo => SetTO(float)

Entorno:

- Temperatura Objetivo: TO
- Temperatura OBjetivo Establecida: TOE
- Ventanas
- Calefacción
- Aire Acondicionado
- Personas y Animales
- Temperatura

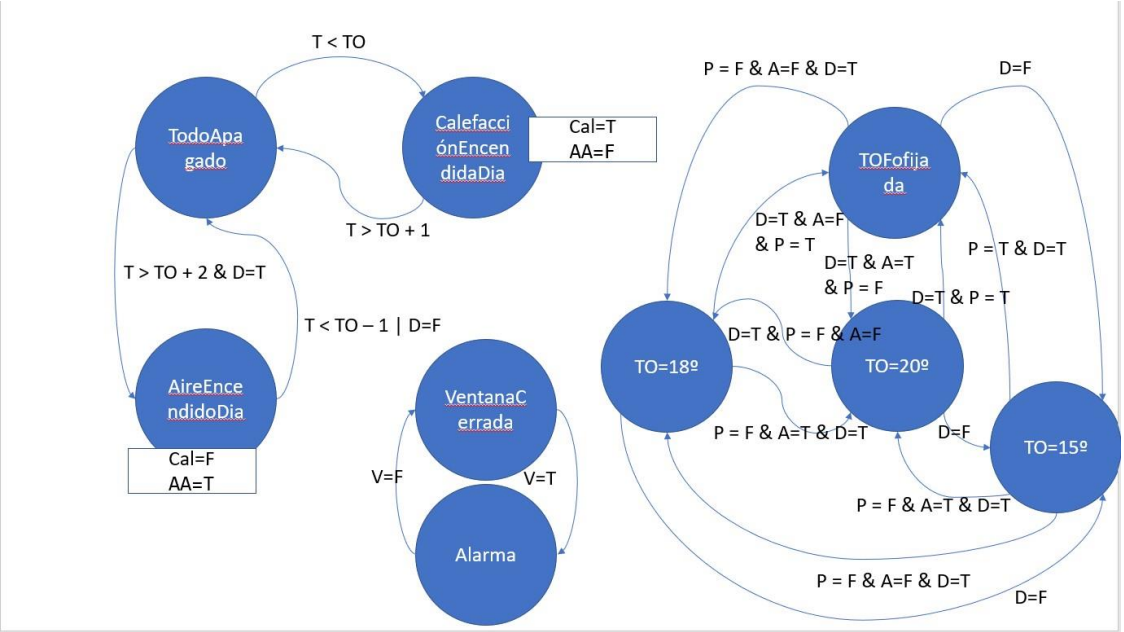
Rendimiento: Temperatura media de la habitación, temperatura mínima y temperatura máxima.

Modelamos el Agente como un agente basado en modelo. Para modelarlo usamos el formalismo de máquinas de estado. Tenemos claramente 3 sub-sistemas que se ejecutan en paralelo.

- El subsistema que detecta si hay o no habitantes en la habitación
- El subsistema que controla la temperatura
- El subsistema que controla la alarma.

Podemos definirlo como una máquina de estados jerárquica con 3 estados. Pero como entre los diferentes subsistemas no parece que haya relación (cambiar la temperatura objetivo no implica un cambio de estado directamente en el sub-sistema de control de temperatura) es más sencillo plantearlo como tres máquinas de estados que se ejecutan de forma paralela en el agente.

Control del agente



Problema 02

Problema 03

Para resolver un problema de búsqueda necesitamos definir cómo vamos a modelar el problema, el estado inicial, el final, las acciones (operadores) y el coste del camino.

Vamos a modelar el problema asumiendo un vector de cajas y la posición del coche

Las cajas guardan:

- La posición en la que están (que puede cambiar) y que puede ser A,B,C o F (furgoneta)
- El destino (que no puede cambiar)
- El peso (que no puede cambiar)
- El id que sólo nos sirve a nivel informativo para seguir el algoritmo.

La furgoneta que guarda la posición en la que está y que puede ser A,B o C

Recordemos la configuración de partida y los datos de cada caja:

Cajas

id	Peso	Posicion	Destino
1	20	A	C
2	50	A	B
3	60	A	C
4	80	A	C

id	Peso	Posicion	Destino
5	10	A	B
6	10	A	C
7	30	B	A
8	60	C	A

Así el estado inicial será:

Cajas

id	Posicion
1	A
2	A
3	A
4	A
5	A
6	A
7	B
8	C

Furgoneta: A

El estado final será

Cajas

id	Posicion
1	C
2	B
3	C
4	C
5	B
6	C
7	A
8	A

Furgoneta: *

Acciones

- Cargar(Caja): Cambia la posición de la caja a F
- Descargar(Caja): Asigna la posición de la caja la posición de la furgoneta. La caja debe estar en posición F

- MoverA: Asigna a la posición de la furgoneta el valor de A. Solo si furgoneta está en B
- MoverB: Asigna a la posición de la furgoneta el valor de B. Solo si furgoneta está en A o C
- MoverC: Asigna a la posición de la furgoneta el valor de C. Solo si furgoneta está en B
- Función de coste: MoverA,MoverB,MoverC tienen un coste de 1 Cargar y descargar tienen un coste de 0. Por lo que primero expandirá los nodos de cargar si es posible antes de mover la furgoneta.

Problema 04

La función heurística para el problema anterior. Debe ser admisible es decir que el coste que estime nunca supere al real.

- Primera aproximación podría ser comprobar simplemente cuantas cajas están bien colocadas. Por cada caja mal colocada estimamos un coste de 1 que será el coste mínimo que tendrá que hacer la furgoneta para moverla a su almacén destino. Pero esta heurística puede no ser admisible en ciertas circunstancias, por ejemplo, si tenemos que entregar dos paquetes de A a B y ambos caben en la furgoneta. Una aproximación mejor es contar el peso de los paquetes entre 80 y redondear hacia arriba. Esto nos da una mejor aproximación de los viajes que tenemos que dar y hace a la función heurística admisible.
- Para tener en cuenta los saltas, podemos modificar la función anterior y tener en cuenta la distancia (número de saltos) a la que tengo que enviar los paquetes en cada ubicación. Pero hay que tener cuidado ya que podemos meter dos cajas que se mueven de A a C en un viaje y serían dos saltos en vez de 4. Así que en principio y para curarnos en salud (y garantizar la admisibilidad de la función heurística) solo añadiremos 1 salto adicional si en el conjunto que queremos enviar de un almacén hay alguna caja que necesitamos enviar a distancia 2. No vamos a tener en cuenta los movimientos de las furgonetas para alcanzar otras cajas en otros almacenes porque podemos utilizar el viaje de una para llegar a esas cajas y volver con ellas.

$$h(x) = ceil\left(\frac{\sum_{c\in C} P\,eso(c)}{80}\right) + 1 \Leftrightarrow \exists c \in C | dist\big(pos(c), dest(c)\big) > 1$$

Generalizando para un problema general donde haya n almacenes a m saltos…

$$h(x) = ceil\left(\frac{\sum_{c\in C} P\,eso(c)}{80}\right) + max(dist\big(posicion(c), destino(c)\big) - 1 \forall c \in C$$

Donde C es el conjunto de Cajas. Hay que tener en cuenta que puede haber cajas en la furgoneta asi que la posición de la caja para efectos del cálculo de la heurística sería la posición de la furgoneta.

Problema 07

Webmaster que tiene 3 posiciones en la web y necesita elegir los tres que mas beneficios le dé entre los 5 que dispone. Cada anuncio tendrá el identificador que indica su id de la tabla, pero en número.

Los números no deben repetirse, así que podemos plantear diferentes alternativas:

- La codificación simple sería asignar a cada posición del gen un banner y colocar los anuncios numerados en los 3 banners. El problema de esto es que tenemos que tener cuidado de no repetir los anuncios. Asi que muchos cruces o mutaciones pueden acabar produciendo individuos inválidos. Se plantean dos alternativas a esto:
- penalizar los individuos inválidos con un fitness negativo
- no permitir que se produzcan individuos inválidos

la función de fitness será recorrer el individuo y calcular el beneficio que se espera obtener con la configuración del individuo, mirando le valor de la tabla.

Otra aproximación es codificar el problema de otra forma. Tenemos números del 1 al 5. Cada número representa la posición que tenía cada elemento en la tabla inicial. Así que en cada gen tenemos una codificación concreta

Gen 0 = de 0 a 5

Gen 1: de 0 a 4

Gen 2 de 0 a 3

Gen 3 de 0 a 2

Gen 4 de 0 a 1

El gen 5 siempre será el que sobre.

Si garantizamos que las mutaciones nunca puedan mutar genes fuera de estas codificaciones y que los cruces van a respetar las posiciones de los genes (es decir que un gen 3 siempre acabara como gen 3 en el hijo después del cruce) entonces podemos realizar esta codificación en base al ordinal de la tabla. Un 0 en el gen 0 significa que hemos puesto el primer elemento en el gen 0. Se sacaría ese primer elemento de la lista y un nuevo gen 0 implicaría que es el segundo elemento de la lista.

Ventaja, no generamos individuos inválidos y reducimos el espacio de búsqueda

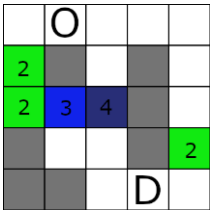
Desventaja: necesitamos un proceso de traducción del gen para saber cual es la solución correcta ya que los valores dependen del orden que fijemos en la tabla inicial y la solución no es aplicable si le orden cambia.

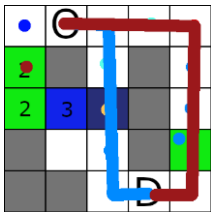
problema 8

Tenemos 10 sensores. 1 de tipo A, 4 de tipo B y 5 de tipo C Cada sensor tendrá una posición X,y en el espacio. Lo que vamos a optimizar es la posición, por tanto tenemos un vector de 20 valores, cada pareja de valores representa la posición X,y del sensor correspondiente.

la función de fitness será calcular el número de casillas que están protegidas por los sensores. Para ello usamos una matriz del tamaño de la habitación y vamos marcando las casillas que cada sensor va detectando. Las marcamos con 1 si la casilla está cubierta o 0.5 si está parcialmente cubierta. Después contamos todas las casillas no nulas que estén ocupadas con valor de 1 o más y aquellas que estén solo parcialmente cubiertas con 0.5. La suma de todas ellas da el fitness. Hay que tener ciertas consideraciones: - Que un sensor no puede asignarse a posiciones no válidas o si está asignado a posiciones no válidas sea equivalente a no poner el sensor. Es decir, ese sensor no contribuirá al cálculo de la matriz de la habitación

problema 9



- ¿Cuál sería el camino más corto? hay dos  tamaño 9

- Si usamos la distancia euclídea, ¿lo resolvería en menos pasos? la distancia euclídea es menos informada.
- ¿Qué pasa si el agente puede moverse en diagonal con coste 1? ¿Sería admisible la distancia de manhattan? no porque podríamos resolver el problema en menos pasos que la heurística.