uc3m | Universidad Carlos III de Madrid

Práctica 2:

Satisfacibilidad Lógica y Búsqueda Heurística

Heurística y Optimización

Fecha	19/12/2018
Autores	Ángel Luis Alonso Blázquez 100363923@alumnos.uc3m.es
	Marcos Arroyo Ruiz 100363919@alumnos.uc3m.es

1. Breve introducción de los contenidos del documento

Este documento estará estructurado de tal manera que la parte 1 muestra una descripción de nuestro modelo SAT y los diferentes casos de prueba (con sus análisis). En la parte 2 se explica el modelo de búsqueda heurística creado, los casos de prueba y el análisis de los mismos. En la parte 3 se describe lo mismo que la anterior pero teniendo en cuenta los movimientos diagonales. Para acabar, se muestran las conclusiones sacada acerca de la práctica.

2. Parte 1: SAT

a. Descripción del modelo.

Restricciones

- Al y las serpientes sólo se pueden colocar en celdas vacías.

$V_{xy} \rightarrow S_{xy} \vee Al_{xy}$ $\quad$ $(\overline{V_{xy}} \vee S_{xy} \vee Al_{xy})$

- Una serpiente no puede estar en la misma fila que otra serpiente.

$Sxy \rightarrow \bigcap_{z=0}^{n} \overline{Sxz}\,;\ y \neq z$ $\quad$ $\bigcap_{z=0}^{n} (\overline{S_{xy}} \vee \overline{Sxz})\,;\ y \neq z$ x: número de fila

y, z: número de columna

- No puede haber ninguna serpiente ni en la misma fila ni en la misma columna que Al.

$Al_{xy} \rightarrow \bigcap_{u=0}^{m} \overline{S_{uy}} \wedge \bigcap_{v=0}^{n} \overline{S_{xv}}\,;\ x \neq u,\ y \neq v$ $\quad$ $\bigcap_{u=0}^{m} (\overline{Al_{xy}} \vee \overline{S_{uy}}) \wedge \bigcap_{v=0}^{n} (\overline{Al_{xy}} \vee \overline{S_{xv}})$

$x \neq u,\ y \neq v$
x, u: número de fila
y, v: número de columna

- Si Al está en una posición no está en resto.

$Al_{xy} \rightarrow \bigcap_{u=0}^{m} \bigcap_{v=0}^{n} \overline{Al_{uv}}\,;\quad x \neq u,\ y \neq v$ $\quad$ $\bigcap_{u=0}^{m} \bigcap_{v=0}^{n} (\overline{Al_{xy}} \vee \overline{Al_{uv}})$

$x \neq u,\ y \neq v$
x, u: número de fila
y, v: número de columna

- Si S está en una posición no está en resto.

$$S_{xy} \rightarrow \bigcap_{u=0}^{m} \bigcap_{v=0}^{n} \overline{S_{uv}}\,; \qquad x \neq u,\ y \neq v \qquad \bigcap_{u=0}^{m} \bigcap_{v=0}^{n} (\overline{S_{xy}} \vee \overline{S_{uv}})$$

$x \neq u,\ y \neq v$
x, u: número de fila
y, v: número de columna

- Al tiene que aparecer una vez.

$$\bigcap_{x=0}^{m} \bigcap_{y=0}^{n} \overline{Al_{xy}} \rightarrow Al_{uv}\,; \qquad \bigcap_{x=0}^{m} \bigcap_{y=0}^{n} (\overline{Al_{xy}} \vee Al_{uv})$$

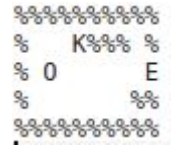
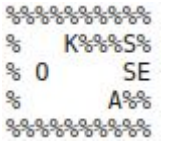
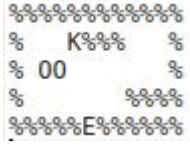
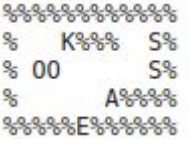
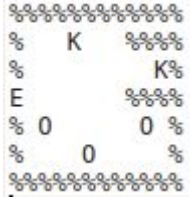
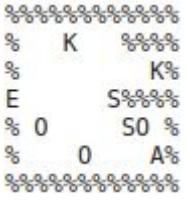
x, u: número de fila
y, v: número de columna

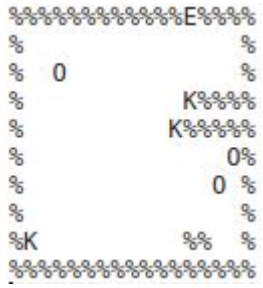
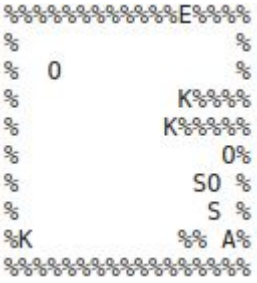
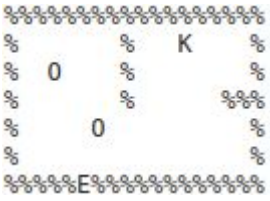
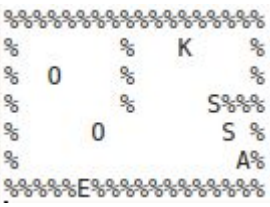
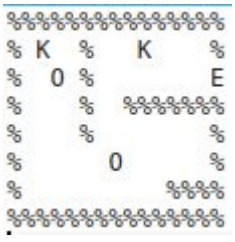
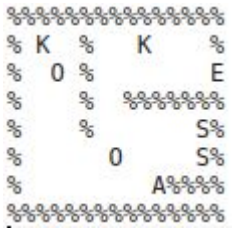
- Una serpiente tiene que aparecer una vez si esta existe.

$$\bigcap_{x=0}^{m} \bigcap_{y=0}^{n} \overline{S_{xy}} \rightarrow S_{uv}\,; \qquad \bigcap_{x=0}^{m} \bigcap_{y=0}^{n} (\overline{S_{xy}} \vee S_{uv})$$

x, u: número de fila
y, v: número de columna

b. Análisis de los resultados.

Inicio	Resultado
	
	
	

Como podemos observar los laberintos se crean cumpliendo con las restricciones establecidas. En todo momento, Al y las serpientes son colocados por el programa lo más a la derecha y abajo posible.
Los dos últimos mapas han sido creados por nosotros.

3. Parte 2: Búsqueda Heurística

a. Descripción del modelo.

Representación de estados

- Tablero

Casilla(x,y)

- x e y valores de posición.
- Estados posibles del valor una casilla:
  - '%' = Pared
  - 'A' = Personaje Inicial
  - 'K' = Llave
  - 'S' = Serpiente
  - 'O' = Roca
  - 'E' = Puerta de salida
  - ' ' = Casilla vacía

Ejemplo:

Tablero: { Casilla(0,0), Casilla(1,0), ..., Casilla(12,5)}

```
%%%%%%%%%%%%
%           %
A   O   S   %
%   K  %%%%%
E   %%%%%%%%
%%%%%%%%%%%%
```

- Estados

Estado: (Keys, End, Tablero)

Keys = nº de llaves restante.
End = 1 si 'A' está encima de 'E', 0 si no.

Ejemplo tablero anterior:

Estado inicial: (1, 0, Tablero)
Estado final: (0, 1, Tablero)

Operadores

En la implementación la 'x' es el eje vertical y la 'y' el horizontal.

Operador	Precondiciones	Efectos
MoverAbajo(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x+1,y) = ' ' ∨ Casilla(x+1,y) = 'K')	Casilla(x,y) = ' ', (si Casilla(x+1,y) = 'K' → Keys = Keys - 1), Casilla(x+1,y) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y) = 'O' ∧ Casilla(x+2,y) = ' '	Casilla(x,y) = ' ', Casilla(x+1,y) = 'A', Casilla(x+2,y) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x+1,y) = 'A', End = 1
MoverArriba(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x-1,y) = ' ' ∨ Casilla(x-1,y) = 'K')	Casilla(x,y) = ' ', (si Casilla(x-1,y) = 'K' → Keys = Keys - 1), Casilla(x-1,y) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y) = 'O' ∧ Casilla(x-2,y) = ' '	Casilla(x,y) = ' ', Casilla(x-1,y) = 'A', Casilla(x-2,y) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x-1,y) = 'A', End = 1
MoverIzq(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x,y-1) = ' ' ∨ Casilla(x,y-1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x,y-1) = 'K' → Keys = Keys - 1), Casilla(x,y-1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x,y-1) = 'O' ∧ Casilla(x,y-2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x,y-1) = 'A', Casilla(x,y-2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x,y-1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x,y-1) = 'A', End = 1
MoverDer(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x,y+1) = ' ' ∨ Casilla(x,y+1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x,y+1) = 'K' → Keys = Keys - 1), Casilla(x,y+1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x,y+1) = 'O' ∧ Casilla(x,y+2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x,y+1) = 'A', Casilla(x,y+2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x,y+1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x,y+1) = 'A', End = 1

Heurísticas informadas y admisibles

- Heurística 1

Mediante relajación de restricciones; Las serpientes no hacen nada y no se tienen en cuenta las paredes:

h2 = { (int) (Distancia a cada una de las llaves desde Al +
Menor distancia a la puerta desde la llave más cercana a E) x 1.5 }

- Heurística 2

Mediante relajación de restricciones; Las serpientes no hacen nada y no se tienen en cuenta las paredes:

h2 = { Distancia a cada una de las llaves desde Al +
Menor distancia a la puerta desde la llave más cercana a E }

Estas heurísticas son admisibles porque nunca sobreestiman el coste de la solución óptima.

b. Análisis de los resultados.

Implementación

El algoritmo A* está implementado de la siguiente manera:

ABIERTA=I, CERRADA=Vaco, EXITO=Falso
Hasta que ABIERTA está vacío O EXITO
Quitar el primer nodo de ABIERTA, N
SI N es Estado-nal ENTONCES EXITO=Verdadero
SI NO Expandir N y meterlo en CERRADA, generando el conjunto S de sucesores de N
Para cada sucesor s en S

1. Si s no está ni en ABIERTA y CERRADA se inserta en orden en ABIERTA
2. Si está en ABIERTA y la función de evaluación f() de s es mejor, se elimina el que ya estaba en ABIERTA y se introduce s en orden
3. Si esta en CERRADA se ignora

Si EXITO Entonces Solucion=camino desde N a I a través de los punteros
Si no Solución=Fracaso

Resultados

- Laberintos predefinidos

H1 = Heurística 1 | H2 = Heurística 2

Laberinto	Tiempo (segundos)	Coste	Longitud	Nodos Expandidos
lab1.map (H1)	0.005822802	18	10	24
lab1.map (H2)	0.008691981	18	10	46
lab2.map (H1)	0.010520325	20	10	76
lab2.map (H2)	0.014228331	20	10	93
lab3.map (H1)	0.079103971	48	24	592
lab3.map (H2)	0.128082036	48	24	883
lab4.map (H1)	0.363854404	60	28	1024
lab4.map (H2)	2.751229082	60	28	2850

- Laberintos personalizados

Laberinto	Tiempo (segundos)	Coste	Longitud	Nodos Expandidos
lab5.map (H1)	0.62962626	100	47	1483
lab5.map (H2)	37.553704974	100	47	15188
lab6.map (H1)	1.713941879	96	46	3200
lab6.map (H2)	4.15176093	96	46	5305
lab5.lab.output (H1)	65.572038347	150	61	14466
lab5.lab.output (H2)	97.805661438	150	61	17169
lab6.lab.output (H1)	6.325304563	102	43	5883
lab6.lab.output (H2)	17.692214761	102	43	10514

* Los laberintos .lab.output son laberintos generados por nuestra implementación SAT.

* Todos los laberintos adicionales se pueden encontrar en las diferentes carpetas.

Análisis de heurísticas

La heurística 1, al tener un valor más grande o igual que la heurística 2 (ambas admisibles), hace que se expandan menos nodos. Esto ocurre porque la lista 'open' está ordenada en función a qué nodos tienen menor función coste (coste del nodo + heurística del mismo). Esto premia las heurísticas admisibles más grandes, lo que hace que la heurística 1 prevalezca sobre la 2.

4. Parte 3 (extra): Búsqueda Heurística

a. Descripción del modelo.

Representación de estados

- Tablero y Estados

Igual que en la parte 2.

Operadores

En la implementación la 'x' es el eje vertical y la 'y' el horizontal.

Operador	Precondiciones	Efectos
MoverAbajo(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x+1,y) = ' ' ∨ Casilla(x+1,y) = 'K')	Casilla(x,y) = ' ', (si Casilla(x+1,y) = 'K' → Keys = Keys - 1), Casilla(x+1,y) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y) = 'O' ∧ Casilla(x+2,y) = ' '	Casilla(x,y) = ' ', Casilla(x+1,y) = 'A', Casilla(x+2,y) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x+1,y) = 'A', End = 1
MoverArriba(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x-1,y) = ' ' ∨ Casilla(x-1,y) = 'K')	Casilla(x,y) = ' ', (si Casilla(x-1,y) = 'K' → Keys = Keys - 1), Casilla(x-1,y) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y) = 'O' ∧ Casilla(x-2,y) = ' '	Casilla(x,y) = ' ', Casilla(x-1,y) = 'A', Casilla(x-2,y) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x-1,y) = 'A', End = 1
	Casilla(x,y) = 'A' ∧ (Casilla(x,y-1) = ' ' ∨ Casilla(x,y-1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x,y-1) = 'K' → Keys = Keys - 1), Casilla(x,y-1) = 'A'

MoverIzq(x,y)	Casilla(x,y) = 'A' ∧ Casilla(x,y-1) = 'O' ∧ Casilla(x,y-2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x,y-1) = 'A', Casilla(x,y-2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x,y-1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x,y-1) = 'A', End = 1
MoverDer(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x,y+1) = ' ' ∨ Casilla(x,y+1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x,y+1) = 'K' → Keys = Keys - 1), Casilla(x,y+1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x,y+1) = 'O' ∧ Casilla(x,y+2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x,y+1) = 'A', Casilla(x,y+2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x,y+1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x,y+1) = 'A', End = 1
MoverAbajoDer(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x+1,y+1) = ' ' ∨ Casilla(x+1,y+1) = 'K')	Casilla(x,y) = ' ', (si Casilla(x+1,y+1) = 'K' → Keys = Keys - 1), Casilla(x+1,y+1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y+1) = 'O' ∧ Casilla(x+2,y+2) = ' '	Casilla(x,y) = ' ', Casilla(x+1,y+1) = 'A', Casilla(x+2,y+2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y+1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x+1,y+1) = 'A', End = 1
MoverArribaDer(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x-1,y+1) = ' ' ∨ Casilla(x-1,y+1) = 'K')	Casilla(x,y) = ' ', (si Casilla(x-1,y+1) = 'K' → Keys = Keys - 1), Casilla(x-1,y+1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y+1) = 'O' ∧ Casilla(x-2,y+2) = ' '	Casilla(x,y) = ' ', Casilla(x-1,y+1) = 'A', Casilla(x-2,y+2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y+1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x-1,y+1) = 'A', End = 1
MoverAbajoIzq(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x-1,y-1) = ' ' ∨ Casilla(x-1,y-1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x-1,y-1) = 'K' → Keys = Keys - 1), Casilla(x-1,y-1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y-1) = 'O' ∧ Casilla(x-2,y-2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x-1,y-1) = 'A', Casilla(x-2,y-2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x-1,y-1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x-1,y-1) = 'A', End = 1

MoverArribaIzq(x,y)	Casilla(x,y) = 'A' ∧ (Casilla(x+1,y-1) = ' ' ∨ Casilla(x+1,y-1) = 'K') ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', (si Casilla(x+1,y-1) = 'K' → Keys = Keys - 1), Casilla(x+1,y-1) = 'A'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y-1) = 'O' ∧ Casilla(x+2,y-2) = ' ' ∧ (Comprobación de que no hay una serpiente en esa fila que pueda evitar el movimiento)	Casilla(x,y) = ' ', Casilla(x+1,y-1) = 'A', Casilla(x+2,y-2) = 'O'
	Casilla(x,y) = 'A' ∧ Casilla(x+1,y-1) = 'E' ∧ Keys = 0	Casilla(x,y) = ' ', Casilla(x+1,y-1) = 'A', End = 1

Heurísticas informadas y admisibles

- Heurística 3

Mediante relajación de restricciones; Las serpientes no hacen nada y no se tienen en cuenta las paredes:

h2 = { (Distancia a cada una de las llaves desde Al +
Menor distancia a la puerta desde la llave más cercana a E) }

b. Análisis de los resultados.

Resultados

- Laberintos predefinidos

H3 = Heurística 3

Laberinto	Tiempo (segundos)	Coste	Longitud	Nodos Expandidos
lab1.map (H3)	0.018853542	18	10	84
lab2.map (H3)	0.014156018	14	17	59
lab3.map (H3)	0.137643634	34	17	627
lab4.map (H3)	11.222570878	50	23	3490

- Laberintos personalizados

Laberinto	Tiempo (segundos)	Coste	Longitud	Nodos Expandidos
lab5.map (H3)	6.916288075	74	35	3624
lab6.map (H3)	60.072001007	78	37	13679
lab5.lab.output (H3)	563.116007123	120	48	29176
lab6.lab.output (H3)	12.25310974	76	33	6608

Análisis de heurísticas

La heurística 3, al poder hacer movimientos en diagonal, mejora el coste y la longitud del camino (como era de esperar). Pero al poder hacer 4 movimientos más, expande bastantes más nodos que las heurísticas 1 y 2, y por lo tanto empeora el muchísimo el tiempo de ejecución.

5. Conclusiones acerca de la práctica

Con respecto a la parte de SAT. Tuvimos complicaciones al comienzo para entender cómo emplear la librería JaCoP. Además hemos tenido que cambiar varias veces el código porque no se cumplían las restricciones que establecíamos. No sabemos por qué los serpientes y Al siempre se colocan lo más abajo y a la derecha posible, pero se cumplen las restricciones por lo que no hay mucho problema con eso.

Respecto a la búsqueda heurística, para hacerla, usamos una lista doblemente enlazada que dimos cuando cursamos la asignatura 'Estructuras de datos y algoritmos'. Programamos toda la lista a mano, pero tuvimos mucha dificultad para implementar el método .sort, por lo que tuvimos que cambiar todo, implementándolo con una LinkedList. Esta última también nos dio problemas y acabamos rehaciendo todo con un ArrayList. En general hemos perdido mucho tiempo por implementar la lista doblemente enlazada a mano y por rehacer todo dos veces.

Hemos invertido bastante tiempo en la práctica en general, más del óptimo para su elaboración por errores nuestros a la hora de implementar las cosas.