Lenguajes de Reglas y Propagación de restricciones.

Daniel Benedí García - NIP: 774156 TP6-1 Inteligencia Artificial

8 de Diciembre de 2020

Índice

Primera Tarea - CLIPS

Fichas

Se ha diseñado el nodo de la siguiente manera:

```
(deftemplate MAIN::nodo
      (multislot estado)
      (multislot camino)
      (slot heuristica)
      (slot coste (default 0))
      (slot clase (default abierto))
)
```

El estado serán las fichas y el hueco, por ejemplo el estado inicial se representaría como 'B B B H V V V'.
El camino se representa a través de las operaciones. Estas son las de mover el hueco 1,2 o 3 fichas a la izquierda o a la derecha: I1, I2, I3, D1, D2, D3.
La heurística utilizada ha sido la de fichas descolocadas y el coste es constante para todos los movimientos, pero está adaptado y probado para que ese coste no sea igual en todas las operaciones.
Para aplicar el algoritmo A*, las operaciones sólo se ejecutan en los estados de clase cerrado y un estado pasa a cerrado si no existe un nodo cuya suma del coste y valor de su heurística es menor.
Traza de ejecución:

```
          CLIPS (6.30 3/17/15)
CLIPS> (load fichas.clp)

[CSTRCPSR1] WARNING: Redefining defmodule: MAIN
Defining deftemplate: nodo
Defining defglobal: estado-inicial
Defining defglobal: estado-final
Defining deffunction: heuristica
Defining deffacts: nodoInicial
Defining defrule: pasa-el-mejor-a-cerrado +j+j+j
Defining defmodule: OPERADORES
Defining defrule: MH1I +j+j
Defining defrule: MH2I +j+j
Defining defrule: MH3I +j+j
Defining defrule: MH1D +j+j
Defining defrule: MH2D +j+j
Defining defrule: MH3D +j+j
Defining defmodule: RESTRICCIONES
Defining defrule: repeticiones-de-nodo +j+j+j
Defining defmodule: SOLUCION
Defining defrule: reconoce-solucion +j+j
Defining defrule: escribe-solucion +j+j
```

```
TRUE
CLIPS> (reset)
CLIPS> (run)
Solucion:(I2 D3 I2 D3 D1 I3 I3 D2 D3 I2)
```

Casillas

Se ha diseñado el nodo de la siguiente manera:

```
(deftemplate MAIN::nodo
      (slot estado)
      (multislot camino)
      (slot coste (default 0))
      (slot clase (default abierto))
)
```

Se ha optado por representar el estado como la posición en la que se está, empezando en 1 y siendo el estado objetivo el 8.
El camino se representa a través de las operaciones. Estas operaciones son las de Andar o Saltar, representadas mediante A o S respectivamente.
Para aplicar el algoritmo Coste Uniforme, las operaciones sólo se ejecutan en los estados de clase cerrado y un estado pasa a cerrado si no existe un nodo cuyo coste es menor.
Traza de ejecución:

```
          CLIPS (6.30 3/17/15)
CLIPS> (load casillas.clp)

[CSTRCPSR1] WARNING: Redefining defmodule: MAIN
Defining deftemplate: nodo
Defining deffacts: estadoInicial
Defining defrule: pasa-el-mejor-a-cerrado-CU +j+j+j
Defining defmodule: OPERADORES
Defining deffunction: Contar-Andar
Defining deffunction: Contar-Saltar
Defining defrule: Andar +j+j
Defining defrule: Saltar +j+j
Defining defmodule: RESTRICCIONES
Defining defrule: repeticiones-de-nodo +j+j+j
Defining defmodule: SOLUCION
Defining defrule: reconoce-solucion +j+j
Defining defrule: escribe-solucion +j+j
TRUE
CLIPS> (reset)
CLIPS> (run)
La solucion tiene 3 pasos
(A) (A) (A) (S)
```

Segunda Tarea - Sudoku

Propagación de Restricciones

Para solucionar este problema, el algoritmo utilizado es una versión modificada del algoritmo de Backtracking, ImprovedBacktrackingStrategy (se le aplicacan todas las mejoras que proporciona aima), para la solución de propagación de restricciones.
El algoritmo de Backtracking trata en hacer una de las posibles asignaciones posibles e ir añadiendolas hasta que se encuentre la solución o se encuentre que no se puede seguir avanzando. En este último caso se retrocede y se prueba otra asignación.
Final de la traza de ejecución:

```
++++++++++
....7..2.
8.......6
.1.2.5...
9.54....8
.........
3....85.1
...3.2.8.
4.......9
.7..6....
SUDOKU INCOMPLETO - Resolviendo
{Cell at [0][7]=2, Cell at [8][1]=7, Cell at [0][4]=7, Cell at
[2][1]=1,  Cell  at  [6][7]=8,  Cell  at  [8][4]=6,  Cell  at
[1][0]=8,  Cell  at  [1][8]=6,  Cell  at  [3][2]=5,  Cell  at
[5][6]=5,  Cell  at  [7][0]=4,  Cell  at  [7][8]=9,  Cell  at
[2][3]=2,  Cell  at  [2][5]=5,  Cell  at  [3][3]=4,  Cell  at
[5][5]=8,  Cell  at  [6][3]=3,  Cell  at  [6][5]=2,  Cell  at
[3][0]=9,  Cell  at  [3][8]=8,  Cell  at  [5][0]=3,  Cell  at
[5][8]=1,  Cell  at  [0][0]=5,  Cell  at  [1][3]=9,  Cell  at
[5][4]=9,  Cell  at  [7][5]=7,  Cell  at  [0][8]=3,  Cell  at
[3][1]=6,  ...,  Cell  at  [2][4]=3,  Cell  at  [2][6]=8,  Cell  at
[4][0]=7,  Cell  at  [4][2]=1,  Cell  at  [4][3]=6,  Cell  at
[4][4]=5,  Cell  at  [4][5]=3,  Cell  at  [4][6]=9,  Cell  at
[4][7]=4,  Cell  at  [5][1]=4,  Cell  at  [5][2]=2,  Cell  at
[5][3]=7,  Cell  at  [5][7]=6,  Cell  at  [6][0]=1,  Cell  at
[6][2]=9,  Cell  at  [6][4]=4,  Cell  at  [6][6]=6,  Cell  at
[7][2]=6,  Cell  at  [7][3]=5,  Cell  at  [7][4]=8,  Cell  at
[7][6]=2,  Cell  at  [7][7]=1,  Cell  at  [8][2]=8,  Cell  at
[8][3]=1, Cell at [8][7]=3}
Time to solve = 0.061 segundos
SOLUCION:
594876123
823914756
617235894
965421378
```

```
781653942
342798561
159342687
436587219
278169435
++++++++++
Numero sudokus solucionados: 156
```

Tercera Tarea - N-Reinas

Algoritmo min-conflicts

Para solucionar este problema, el algoritmo utilizado es una versión modificada del algoritmo de MinConflicts, con un máximo de 1000 pasos, para la solución de propagación de restricciones.
El algoritmo consiste en un máximo de iteraciones (pasos que se le dan como parámetro) y un estado inicial aleatorio con todas las asignaciones. En cada iteración primero se comprueba si estamos en un estado solución, si es así se termina, sino se escoge una asignación con variables en conflicto de manera aleatoria. A dicha asignación se le asigna el valor que minimice el número de conflictos. Ahí terminaría la iteración.
Al tener un número máximo de pasos, puede no encontrar una solución, por eso la ejecución está dentro de un bucle while que lo vuelve a repetir hasta que encuentre una solución.

Traza de ejecución:

```
{Cell at [0]=4, Cell at [1]=6, Cell at [2]=1, Cell at [3]=5,
Cell at [4]=2, Cell at [5]=0, Cell at [6]=7, Cell at [7]=3}
Time to solve = 0.008 segundos
SOLUCION:
-----Q--
--Q-----
----Q---
-------Q
Q-------
---Q----
-Q------
------Q-
```