

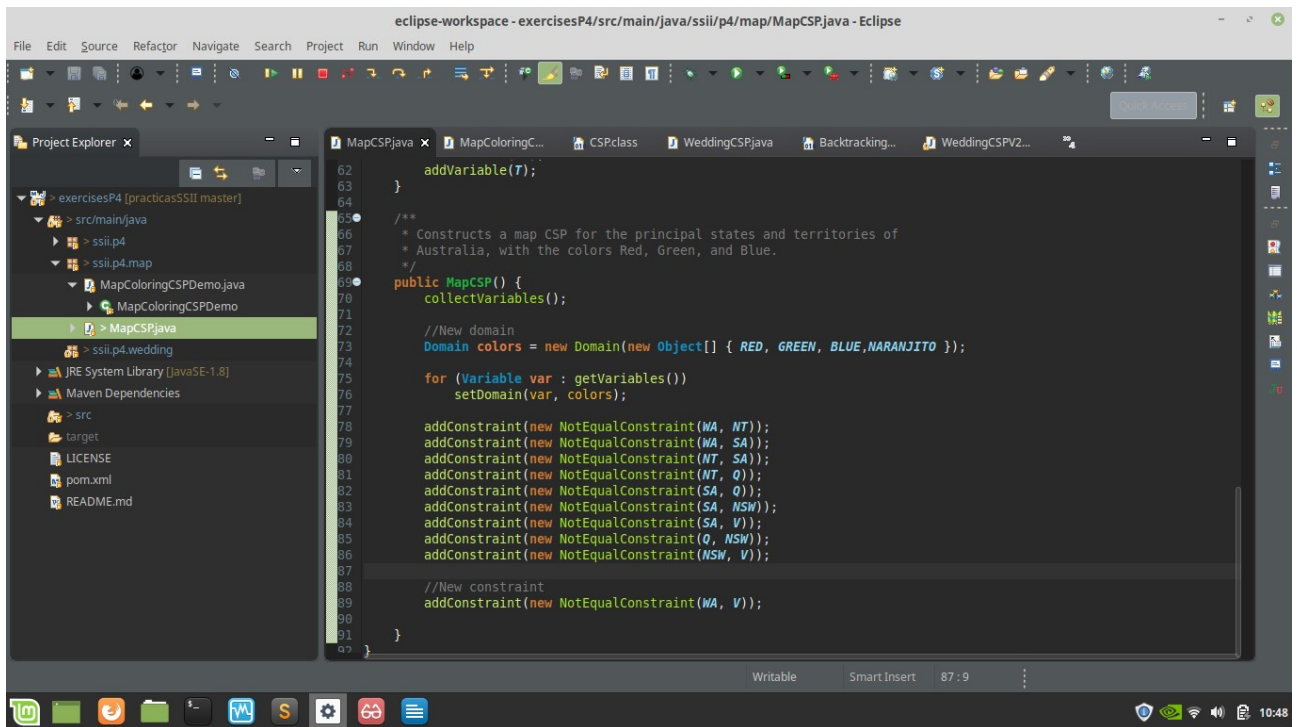
Práctica 4

CSP

**Fernando Candelario Herrero
Gonzalo Sanz Rodríguez**

MODIFICACIONES

- Modifica MapCSP para incluir una nueva restricción. La nueva restricción consiste en que WA no puede tener el mismo color que V. ¿Salen más o menos soluciones? ¿Se parecen?



Versión sin modificar:

#Consola

Map Coloring (Minimum Conflicts)

{NSW=GREEN, WA=RED, NT=GREEN, Q=RED, SA=BLUE, V=RED, T=BLUE}

assignment changes: 5

Map Coloring (Backtracking + MRV + DEG + AC3 + LCV)

{SA=RED, NSW=GREEN, WA=BLUE, NT=GREEN, Q=BLUE, V=BLUE, T=RED}

assignment changes: 7; domain changes: 3

Map Coloring (Backtracking)

{NSW=RED, WA=GREEN, NT=RED, Q=GREEN, SA=BLUE, V=GREEN, T=RED}

assignment changes: 27

Versión modificada:

#Console

Map Coloring (Minimum Conflicts)

null

assignment changes: 1.001

Map Coloring (Backtracking + MRV + DEG + AC3 + LCV)

null

assignment changes: 9; domain changes: 9

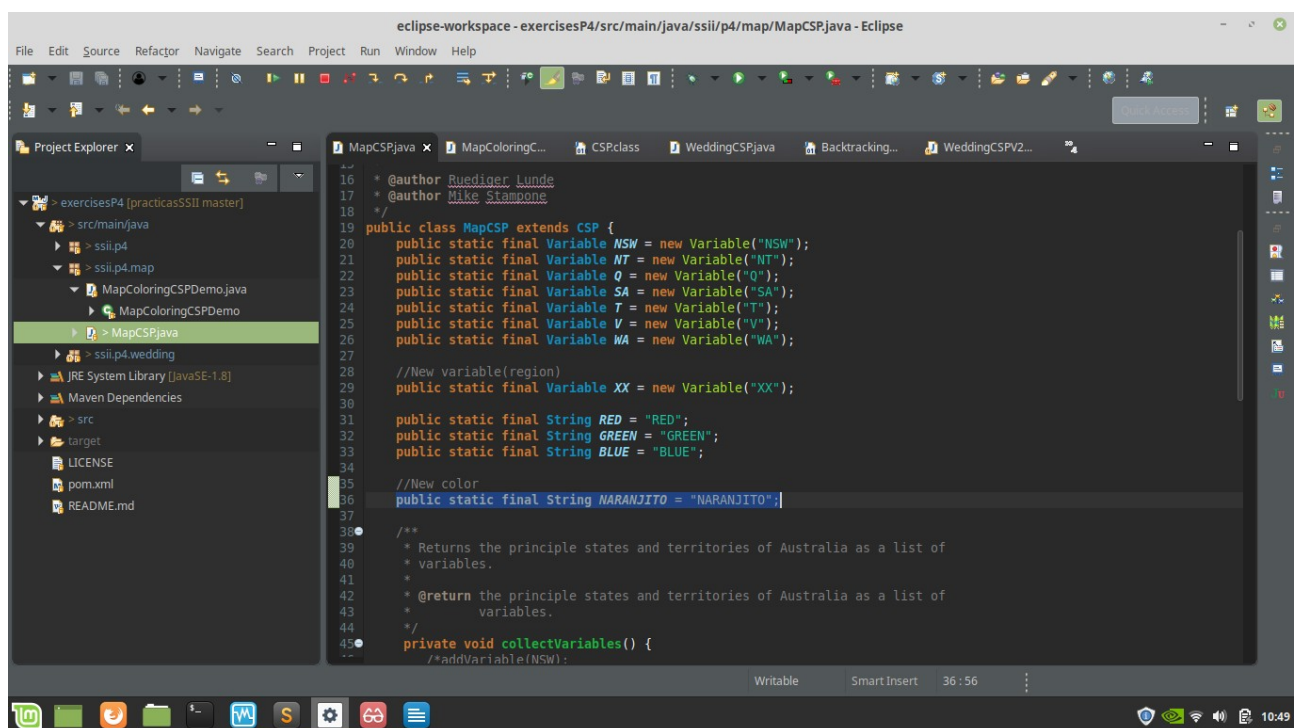
Map Coloring (Backtracking)

null

assignment changes: 183

Al ejecutarlo con la modificación no encuentra ninguna solución. No se parecen en nada las salidas.

- Modifica el mapa MapCSP anterior para incluir un nuevo color llamado NARANJITO. ¿Cómo son las soluciones?



```
eclipse-workspace - exercisesP4/src/main/java/ssii/p4/map/MapCSP.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer:
  exercisesP4 [practicasSSII master]
    > src/main/java
      > ssii.p4
        > ssii.p4.map
          MapColoringCSPDemo.java
          MapColoringCSPDemo
          MapCSP.java
          > ssii.p4.wedding
            JRE System Library [JavaSE-1.8]
            Maven Dependencies
            > src
            target
            LICENSE
            pom.xml
            README.md
MapCSP.java
16  * @author Ruediger Lunde
17  * @author Mike Stampone
18  */
19  public class MapCSP extends CSP {
20      public static final Variable NSW = new Variable("NSW");
21      public static final Variable NT = new Variable("NT");
22      public static final Variable Q = new Variable("Q");
23      public static final Variable SA = new Variable("SA");
24      public static final Variable T = new Variable("T");
25      public static final Variable V = new Variable("V");
26      public static final Variable WA = new Variable("WA");
27
28      //New variable(region)
29      public static final Variable XX = new Variable("XX");
30
31      public static final String RED = "RED";
32      public static final String GREEN = "GREEN";
33      public static final String BLUE = "BLUE";
34
35      //New color
36      public static final String NARANJITO = "NARANJITO";
37
38      /**
39       * Returns the principle states and territories of Australia as a list of
40       * variables.
41       * @return the principle states and territories of Australia as a list of
42       *         variables.
43       */
44      private void collectVariables() {
45          //addVariable(NSW);
46      }
47  }
```

Soluciones:

#Console

Map Coloring (Minimum Conflicts)

{NSW=BLUE, WA=BLUE, NT=NARANJITO, Q=RED, SA=GREEN,
V=NARANJITO, T=RED}
assignment changes: 2

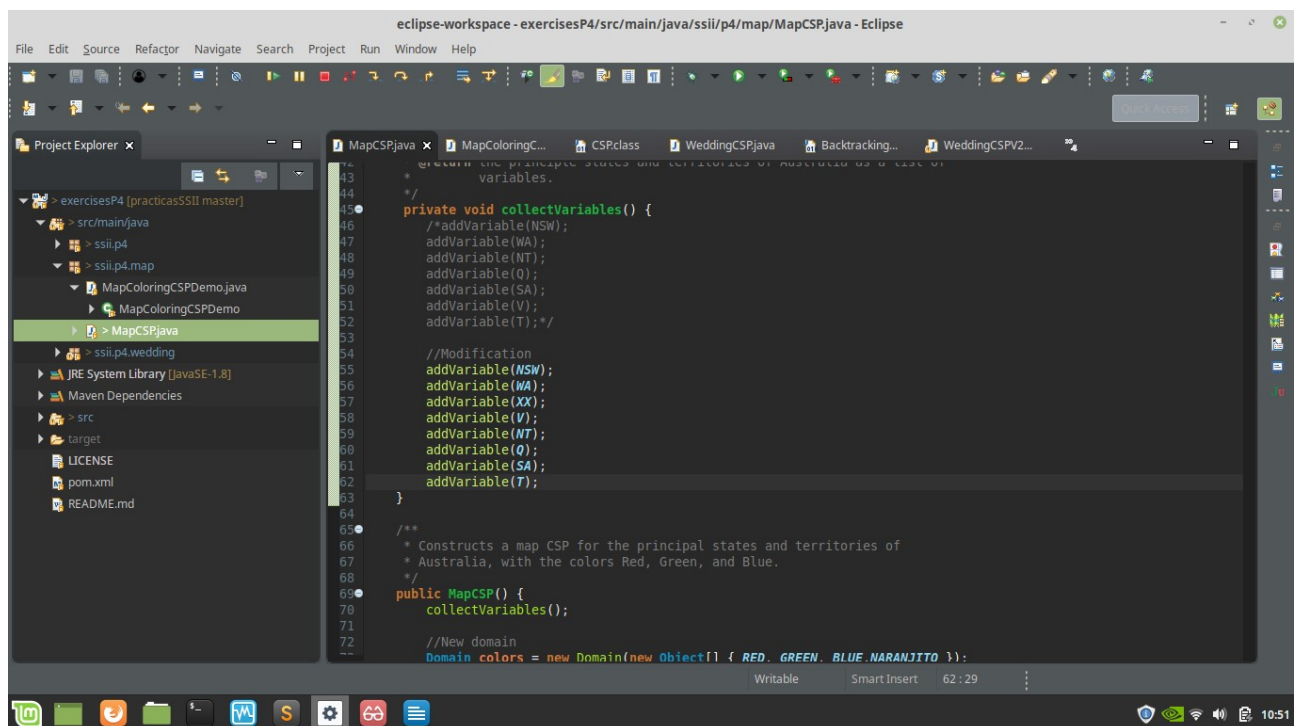
Map Coloring (Backtracking + MRV + DEG + AC3 + LCV)

{SA=RED, NSW=GREEN, Q=BLUE, NT=GREEN, WA=BLUE,
V=NARANJITO, T=RED}
assignment changes: 7; domain changes: 6

Map Coloring (Backtracking)

{NSW=RED, WA=RED, NT=GREEN, Q=BLUE, SA=NARANJITO, V=GREEN,
T=RED}
assignment changes: 14

- Modifica el mapa MapCSP para incluir una nueva región XX que se situa junto a WA y V ¿Qué soluciones salen?



Soluciones:

#Console

Map Coloring (Minimum Conflicts)

{NSW=NARANJITO, WA=BLUE, XX=RED, V=GREEN, NT=GREEN,
Q=BLUE, SA=RED, T=RED}

assignment changes: 2

Map Coloring (Backtracking + MRV + DEG + AC3 + LCV)

{SA=RED, NSW=GREEN, V=BLUE, WA=GREEN, NT=BLUE,
Q=NARANJITO, XX=RED, T=RED}

assignment changes: 8; domain changes: 7

Map Coloring (Backtracking)

{NSW=RED, WA=RED, XX=RED, V=GREEN, NT=GREEN, Q=BLUE,
SA=NARANJITO, T=RED}

assignment changes: 15

- Ejecutar el programa principal que demuestra el caso de la boda.
mvn exec:java -Dexec.mainClass=ssii.p4.wedding.WeddingCSP
¿cuánto tarda en ejecutar el ejemplo? ¿cuántas asignaciones intenta?

#Console

Table table3

chair1-3=m1,chair2-3=m6,chair3-3=h3,chair4-3=h6,chair5-3=m3,

Table table2

chair1-2=m7,chair2-2=h7,chair3-2=h8,chair4-2=h2,chair5-2=m2,

Table table1

chair1-1=m4,chair2-1=m5,chair3-1=h1,chair4-1=h4,chair5-1=h5,

assignment changes: 18.060.904; domain changes: 5.885.427 in
26.44675 minutes

assignment changes: 18.060.904; domain changes: 5.885.427

[INFO] -----

[INFO] BUILD SUCCESS

[INFO] -----

[INFO] Total time: 26:28 min

[INFO] Finished at: 2019-03-07T17:10:07+01:00

[INFO] Final Memory: 10M/37M

[INFO] -----

Tarda 26,28 minutos y realiza 18.060.904 intentos de asignación

- Busca una relajación del problema para que tarde menos modificando WeddingCSP

Una relajación posible sería quitar algunas restricciones de parejas en una misma mesa, que haya menos parejas que tengan que sentarse en la misma mesa.

Resultado de quitar 3 restricciones de parejas

#Console

Table table3

chair1-3=m1,chair2-3=m6,chair3-3=m7,chair4-3=h3,chair5-3=h7,

Table table2

chair1-2=h4,chair2-2=h5,chair3-2=h6,chair4-2=h2,chair5-2=m2,

Table table1

chair1-1=m3,chair2-1=m4,chair3-1=m5,chair4-1=h1,chair5-1=h8,

assignment changes: 2.651.574; domain changes: 1.027.941 in
2.6040332 minutes

assignment changes: 2.651.574; domain changes: 1.027.941

[INFO] -----

[INFO] BUILD SUCCESS

[INFO] -----

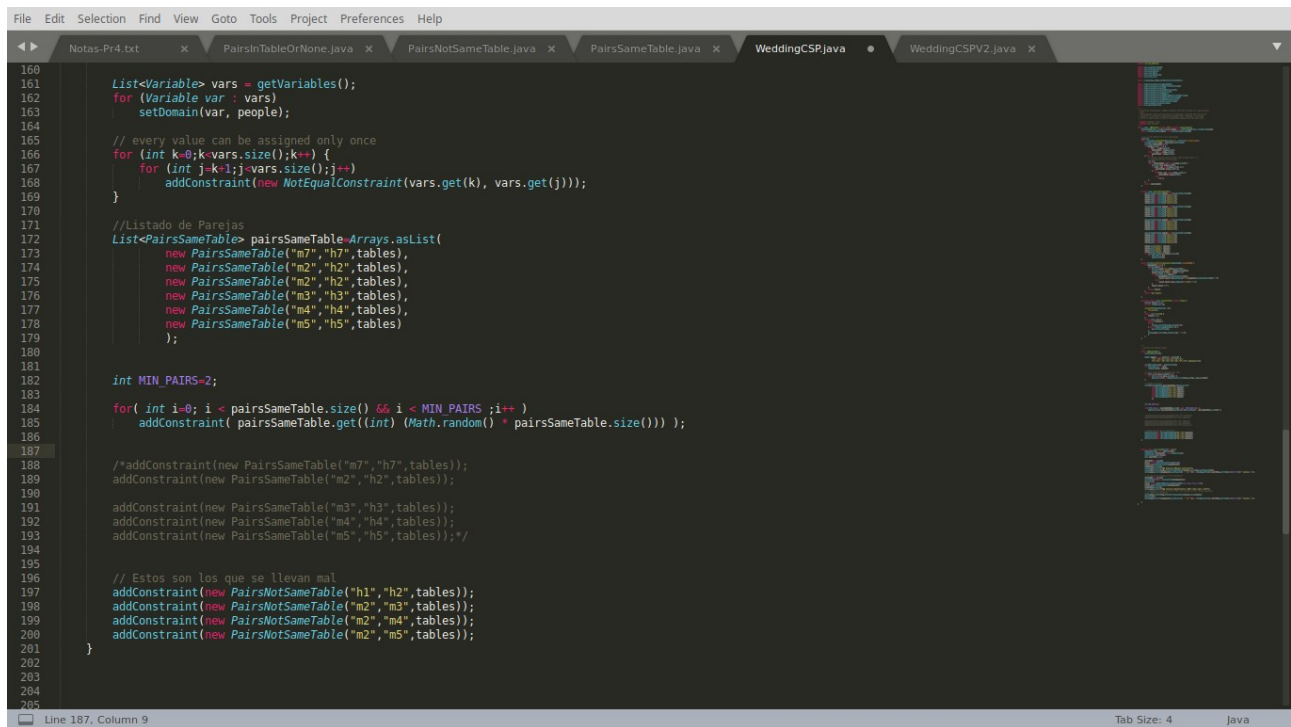
[INFO] Total time: 02:37 min

[INFO] Finished at: 2019-03-07T17:17:58+01:00

[INFO] Final Memory: 10M/37M

Otra posible relajación sería establecer un mínimo de parejas en una misma mesa. Como mínimo tiene que haber "x" parejas en una misma mesa contando todas las mesas en su totalidad.

Ejemplo de implementación de Min parejas:



```
160
161 List<Variable> vars = getVariables();
162 for (Variable var : vars)
163     setDomain(var, people);
164
165 // every value can be assigned only once
166 for (int k=0;k<vars.size();k++) {
167     for (int j=k+1;j<vars.size();j++)
168         addConstraint(new NotEqualConstraint(vars.get(k), vars.get(j)));
169 }
170
171 //Listado de Parejas
172 List<PairsSameTable> pairsSameTable=Arrays.asList(
173     new PairsSameTable("m7","h7",tables),
174     new PairsSameTable("m2","h2",tables),
175     new PairsSameTable("m2","h2",tables),
176     new PairsSameTable("m3","h3",tables),
177     new PairsSameTable("m4","h4",tables),
178     new PairsSameTable("m5","h5",tables)
179 );
180
181
182 int MIN_PAIRS=2;
183
184 for( int i=0; i < pairsSameTable.size() && i < MIN_PAIRS ;i++ )
185     addConstraint( pairsSameTable.get((int) (Math.random() * pairsSameTable.size())) );
186
187
188 //addConstraint(new PairsSameTable("m7","h7",tables));
189 addConstraint(new PairsSameTable("m2","h2",tables));
190
191 addConstraint(new PairsSameTable("m3","h3",tables));
192 addConstraint(new PairsSameTable("m4","h4",tables));
193 addConstraint(new PairsSameTable("m5","h5",tables));
194
195
196 // Estos son los que se llevan mal
197 addConstraint(new PairsNotSameTable("h1","h2",tables));
198 addConstraint(new PairsNotSameTable("m2","m3",tables));
199 addConstraint(new PairsNotSameTable("m2","m4",tables));
200 addConstraint(new PairsNotSameTable("m2","m5",tables));
201
202
203
204
205
```

- Ejecutar el programa principal que demuestra una variante del caso de la boda `mvn exec:java -Dexec.mainClass=ssii.p4.wedding.WeddingCSPV2`
¿Cuánto tarda en ejecutar el ejemplo? ¿cuántas asignaciones intenta?

Tarda 14,26 minutos y realiza 42.402.731 asignaciones.

#Console Backtracking solution

Table table3

chair1-3=m1,chair2-3=m6,chair3-3=h6,chair4-3=h3,chair5-3=m3,

Table table2

chair1-2=h8,chair2-2=m7,chair3-2=h7,chair4-2=h2,chair5-2=m2,

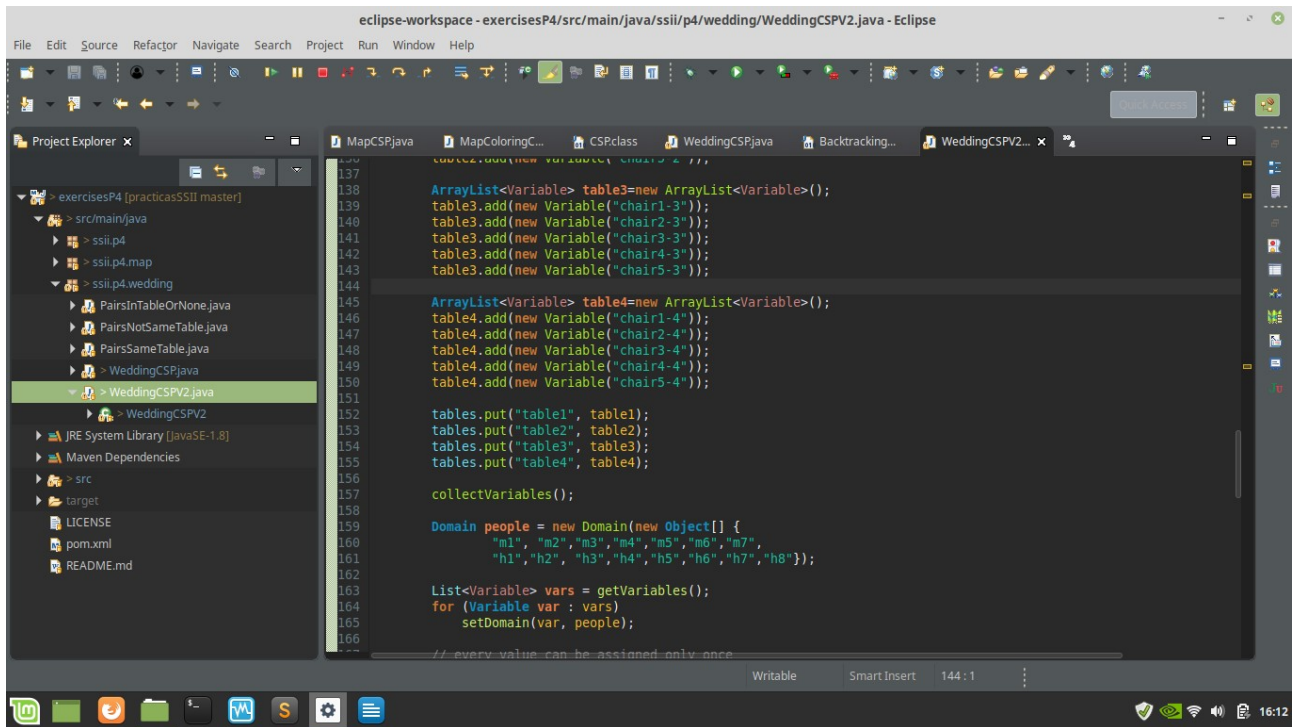
Table table1

chair1-1=m5,chair2-1=h1,chair3-1=h5,chair4-1=h4,chair5-1=m4,

assignment changes: 42.402.731; domain changes: 13.361.194 in
14.257334 minutes

EXPERIMENTANDO

1. En el ejemplo de la boda, prueba a definir una mesa más



2. En el ejemplo de la boda, intenta definir como válido que la silla esté vacía

3. En el ejemplo de la boda, ¿Cómo definirías el que en una mesa sólo hubiera solteros?

Respuesta: 2 y 3

Hemos pensado que una forma de definir los solteros sería en función de las restricciones de parejas que se determinen. Sabiendo las parejas, el resto de los individuos serían considerados solteros, si fuese necesario dejar una mesa entera de solo solteros y no hubiese solteros suficientes para completar la mesa, sería obligatorio definir el comportamiento de una silla vacía. La silla vacía se consideraría como un elemento del dominio que puede ser reasignado a varias variables, esto obliga a definir una restricción adicional al problema que es, que todos los elementos del dominio deben ser asignados a alguna variable excepto el de silla vacía.

PREGUNTAS

1. ¿es posible definir tantas restricciones de tal forma que no haya respuesta?
Razona la respuesta

Si, esto pasa por ejemplo al modificar el MapCSP añadiendo la restricción de que WA no tenga el mismo color que V que el algoritmo no encuentra ninguna solución dadas las restricciones impuestas, esto da a entender que las restricciones son conflictivas entre ellas, si una se cumple otra no.

2. ¿Qué utilidad ves a esta técnica para otro caso diferente que no sea el coloreado de un mapa? ¿Podrías argumentarlo?

Por ejemplo la forma para la planificación de tareas en un equipo de trabajo. El dominio estaría formado por las tareas a realizar y las variables serían los intervalos de tiempo en los que esas tareas se realizan. Algunas restricciones podrían ser: que ciertas tareas tengan un máximo de tiempo que pueden estar realizándose y luego se debe relegar a otra tarea, que ciertas tareas sean prioritarias y se realicen antes que otras.

Planificación de procesos en un SO.

3. ¿Ves dificultades en el uso de CSP para resolver problemas? ¿Cuáles?

El planteamiento del CSP en primera vista parece sencillo, asignar a cada variable un valor de tu dominio y encontrar una combinación de variables/ dominio que cumplan las restricciones. Un primer problema que se presenta es la progresión del coste en tiempo al seguir una estrategia de backtraking, incrementando ligeramente el número de restricciones, el tiempo para resolver el problema se incrementa considerablemente. El segundo problema que nos percatamos consiste en el conflicto que pueda darse entre las restricciones haciendo que el algoritmo no encuentre una solución, darse cuenta de que restricciones se oponen a otras no siempre es trivial sobre todo si hay muchas restricciones que satisfacer.

4. Si en lugar de tener este mapa, se tuviera uno diez veces mayor, ¿qué crees que pasaría? ¿Y qué harías tú en ese caso?

El tiempo para resolver el problema se dispararía convirtiéndose en un problema intratable. Intentar paralelizarlo de alguna forma y buscando alguna relajación al problema.

CONCLUSIONES

Fernando:

He aprendido una forma de enfocar los problemas viendolos como un conjunto de variables que toman un determinado valor y al tomar valor pueden cumplir o no una serie de propiedades (restricciones), si el conjunto de variables cumplen todas las propiedades impuestas entonces es considerado una solución al problema. Esta abstracción puede ser muy interesante para problemas que requieran organizar tareas o individuos.

Gonzalo:

Me ha resultado muy interesante el procedimiento de ordenamiento del algoritmo CSP, organizando el dominio en las variables gracias al backtracking y de este modo comprobar que cumple en cada variable las numerosas restricciones impuestas por nosotros además de los numerosos resultados posibles que podemos obtener, también he sido consciente de todos los posibles errores que puedan surgir a partir de las restricciones.