UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. Sistemas Informáticos y Computación
Escuela Técnica Superior de Ingeniería Informática
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

CSPs

Constraint Satisfaction Problems

Indice

Parte 1: Evaluación del problema de las n Reinas

Parte 2: Elegid un problema de los siguientes. Modelarlo como un CSP e implementarlo en Conflex.

2.1) Sodoku
2.2) Cuadrado Mágico
2.3) Mapas coloreados (ver ejemplo en transparencias)
2.4) Lectores y Periódicos
2.5) El Heptágono
2.6) Ajedrez extendido
2.7) Ajedrez mixto
2.8) Caballos del ajedrez
2.9) Mapas coloreados -2
2.10) Futbol
2.11) Policias

Parte 3: Elegid un problema de los siguientes (dependiendo de la elección anterior). Modelarlo como un CSP e implementarlo en Conflex.

3.1) Las reuniones (original de Lewis Caroll en su obra Lógica Simbólica). (si 1,2,3, 5, 9 ,10,11).
3.2) Sal y mostaza (original de Lewis Caroll en su obra Lógica Simbólica). (si 1, 2, 3, 5, 9, 10,11):
3.3) Las casas y sus dueños (original de Lewis Caroll en su obra Lógica Simbólica). (si 4, 5, 6, 7, 10):
3.4) Nacen y Viven (si 2, 4, 6, 7, 10)
3.5) Asignación de locales (si 4, 6, 7, 8, 10, 11)
3.6) Invitados a una mesa (si 1, 2, 3, 5, 9, 10, 11)
3.7) Invitados a una fiesta (si 4, 6, 7, 8, 10)
3.9) Criptogramas (si 6, 7, 8, 9, 11)
3.10) Planificación de tareas (con cualquiera de los anteriores)

Objetivo:
El objetivo de esta práctica es realizar una memoria que incluya la modelización, resultados y análisis crítico de los ejercicios propuestos. Adicionalmente, el alumno deberá aportar todos los programas realizados en formato Conflex que justifiquen la información aportada en la memoria.

En la parte 1, el alumno deberá completar las tablas propuestas, contrastar e interpretar los resultados obtenidos para cada alternativa (independientemente, no combinándolos) tanto para 1 solución como para todas las soluciones.
En la parte 2 y 3, el alumno deberá modelar el problema seleccionado, explicando en la memoria las variables generadas, su dominio y las restricciones resultantes. Finalmente deberá mostrar y realizar un análisis de los resultados obtenidos con el modelo realizado.

Parte-1) Ejercicio de Evaluación de las n Reinas:

Realizar una evaluación del problema de las n-reinas, para valores de n (4..9), en función de:

- Algoritmos de búsqueda: Backtracking (bt), Forward checking (fc), Looking Ahead (rfla)
- Heurísticas variables:
    a. estáticas (orden del dominio más pequeño, orden de variables más ligadas, relación dominio-grado más pequeño)
    b. dinámicas (orden del dominio más pequeño, relación dominio-grado más pequeño)
- Heurísticas Valores (más pequeño, más grande, en medio)

Especificación para 9x9 reinas

```
##### CSP para 9x9 reinas
########################################
###   PARAMETROS  ###
########################################

\alpha = 0.1;
\filtering : f ;
\search :
        bt,
#       fc,
#       rlfa,
        all_solutions
#       first_solution
;
\static_labeling_order  :
        smallest_domain
#       greatest_degree
#       smallest_domain_by_degree
;
#\dynamic_labeling_order :
#       smallest_domain
#       smallest_domain_by_degree
;
\value_order:
        bottom_first;
#       top_first;
#       mid_first;
#
\verbose :
        display_solutions
#       display_csp
#       display_filtering
#       display_search
#     display_intervals
;

##############################
###     VARIABLES       ###
##############################

\vi : Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9 1..9 ;

##############################
###    RESTRICCIONES     ###
##############################
```

```
\ci : rd1 , abs (Z1 - Z2) != 1 ;
\ci : rd2 , abs (Z1 - Z3) != 2 ;
\ci : rd3 , abs (Z1 - Z4) != 3 ;
\ci : rd4 , abs (Z1 - Z5) != 4 ;
\ci : rd5 , abs (Z1 - Z6) != 5 ;
\ci : rd6 , abs (Z1 - Z7) != 6 ;
\ci : rd7 , abs (Z1 - Z8) != 7 ;
\ci : rd8 , abs (Z1 - Z9) != 8 ;

\ci : rd10 , abs (Z2 - Z3) != 1 ;
\ci : rd11 , abs (Z2 - Z4) != 2 ;
\ci : rd12 , abs (Z2 - Z5) != 3 ;
\ci : rd13 , abs (Z2 - Z6) != 4 ;
\ci : rd14 , abs (Z2 - Z7) != 5 ;
\ci : rd15 , abs (Z2 - Z8) != 6 ;
\ci : rd16 , abs (Z2 - Z9) != 7 ;

\ci : rd18 , abs (Z3 - Z4) != 1 ;
\ci : rd19 , abs (Z3 - Z5) != 2 ;
\ci : rd20 , abs (Z3 - Z6) != 3 ;
\ci : rd21 , abs (Z3 - Z7) != 4 ;
\ci : rd22 , abs (Z3 - Z8) != 5 ;
\ci : rd23 , abs (Z3 - Z9) != 6 ;

\ci : rd25 , abs (Z4 - Z5) != 1 ;
\ci : rd26 , abs (Z4 - Z6) != 2 ;
\ci : rd27 , abs (Z4 - Z7) != 3 ;
\ci : rd28 , abs (Z4 - Z8) != 4 ;
\ci : rd29 , abs (Z4 - Z9) != 5 ;

\ci : rd31 , abs (Z5 - Z6) != 1 ;
\ci : rd32 , abs (Z5 - Z7) != 2 ;
\ci : rd33 , abs (Z5 - Z8) != 3 ;
\ci : rd34 , abs (Z5 - Z9) != 4 ;

\ci : rd36 , abs (Z6 - Z7) != 1 ;
\ci : rd37 , abs (Z6 - Z8) != 2 ;
\ci : rd38 , abs (Z6 - Z9) != 3 ;

\ci : rd40 , abs (Z7 - Z8) != 1 ;
\ci : rd41 , abs (Z7 - Z9) != 2 ;

\ci : rd43 , abs (Z8 - Z9) != 1 ;

\cim : ct1 , <>(Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9);
```

Ejemplo de resultados:

El objetivo es alcanzar un tamaño de n razonable con el tiempo de cómputo.

(1 solución)	n=4	n=6	n=8	n=9	n=10	n=12?
BT	**26**	**171**	**876**	**333**	975	**3066**
FC	**7**	**18**	**57**	**20**	47	**104**
RFLA	**5**	**18**	**40**	**-**	42	**79**

(todas las soluciones)	n=4	n=6	n=8	n=9	n=10	n=12?
BT	**60**	**894**	**15720**	**72378**	348150	**-**
FC	**14**	**86**	**1192**	**4741**	17048	**-**
RFLA	**10**	**74**	**950**	**-**	12840	**-**

Heuristicas variables (1 solución) Con BT	n=4	n=6	n=8	n=9	n=10	n=12?
Static: smallest_domain	**26**	**171**	876	**333**	975	**3066**
Static: smallest_domain_by_degree	**26**	**171**	876	**333**	975	**3066**
Dynamic: smallest_domain	**26**	**171**	876	**333**	975	**3066**
Dynamic: smallest_domain_by_deg	**26**	**171**	876	**333**	975	**3066**

Heuristicas valores (1 solucion) Con BT	n=4	n=6	n=8	n=9	n=10	n=12?
bottom_first	**26**	**171**	876	**333**	975	**3066**
top_first	**26**	**171**	876	**333**	975	**3066**
mid_first	**10**	**135**	84	**90**	195	**138**

El ***valor a medir es el número de instanciaciones*** que devuelve CONFLEX (no el tiempo de cálculo, que es dependiente de cada procesador).

ENTREGA-1:

- **Resultados obtenidos. Llegar a un tamaño de n razonable con el tiempo de cómputo y suficientemente discriminatorio.**
- **Contrastar y comentar los resultados obtenidos.**
- **El objetivo de esta entrega NO es obtener simplemente los datos de las tablas, sino que estos resultados son un medio para poder obtener conclusiones sobre cómo afecta una u otra elección, en cada uno de los casos anteriores y para cada tamaño del problema.**

Conclusiones apartado 1

A la vista de los resultados obtenidos, llegamos a las siguientes conclusiones. Primero de todo , como es de esperar el número de instancias crece a medida que crece la talla del problema . Cabe destacar que el número de instancias generadas utilizando el algoritmo backtracking es significativamente mayor que utilizando los demás algoritmos. Esto se debe a que los demás algoritmos están más informados.

Respecto a las distintas heurísiticas no se observan cambios debido a que se ha usado el algoritmo de backtracking, que no tiene en cuenta ninguna heurística.

Por norma general parece ser que la mejor opción es no utilizar backtracking y hacer uso del algoritmo RFLA. Sin embargo debemos tener en cuenta que aunque RFLA pueda generar menos instancias que FC , también requiere mayor tiempo en las comporbaciones.

Apartado 2: Modelar un problema en CSP

En esta parte se pide implementar uno de los diversos problemas propuestos. He elegido el problema del **Sudoku** ya que es un problema con el que alguna vez me he encontrado en la vida cotidiana. Dicho esto vamos a empezar con la implementación.

Según las reglas del sudoku para completar el juego, es necesario rellenar una serie de casillas con números del 1 al 9. Existen tres restricciones, no puede haber dos números iguales en la misma fila, tampoco en la misma columna y tampoco dentro de las regiones de 3x3 casillas en las que se divide el espacio.

En este problema **vamos a modelar un sudoku de 9x9 casillas**. En total tendremos un total de 81 casillas y 9 regiones distintas. Para modelar el problema vamos a necesitar tantas variables cómo casillas, es decir 81 variables. La definición de las variables es la siguiente: (A la derecha tenemos un modelo de la representación del problema)

```
##############################
### VARIABLES ###
##############################
#Cada fila representa los valores de una cuadricula
\vi :X11,X12,X13,X14,X15,X16,X17,X18,X19,
     X21,X22,X23,X24,X25,X26,X27,X28,X29,
     X31,X32,X33,X34,X35,X36,X37,X38,X39,
     X41,X42,X43,X44,X45,X46,X47,X48,X49,
     X51,X52,X53,X54,X55,X56,X57,X58,X59,
     X61,X62,X63,X64,X65,X66,X67,X68,X69,
     X71,X72,X73,X74,X75,X76,X77,X78,X79,
     X81,X82,X83,X84,X85,X86,X87,X88,X89,
     X91,X92,X93,X94,X95,X96,X97,X98,X99 1..9;
```

X11	X12	X13	X21	X22	X23	X31	X32	X33
X14	X15	X16	X24	X25	X26	X34	X35	X36
X17	X18	X19	X27	X28	X29	X37	X38	X39
X41	X42	X43	X51	X52	X53	X61	X62	X63
X44	X45	X46	X54	X55	X56	X64	X65	X66
X47	X48	X49	X57	X58	X59	X67	X68	X69
X71	X72	X73	X81	X82	X83	X91	X92	X93
X74	X75	X76	X84	X85	X86	X94	X95	X96
X77	X78	X79	X87	X88	X89	X97	X98	X99

Dónde cada variable de la forma XYZ representa una casilla. Y hace referencia a la región a la que pertenece una variable. Z
hace referencia a la posición relativa dentro de esa misma región. Finalmente se indica que los valores que pueden tomar las variables están comprendidos entre 1 y 9 (mediante 1..9) .

Una vez ya tenemos representadas las variables sólo queda implementar las restricciones. A parte de las restricciones mencionadas anteriormente, también deberemos añadir otras para introducir la información del problema, es decir, aquellas casillas con números ya asignados.

De este modo las restricciones son:

```
##Restricciones para las filas ##

\cim : fila1 , <>(X11,X12,X13,X21,X22,X23,X31,X32,X33);
\cim : fila2 , <>(X14,X15,X16,X24,X25,X26,X34,X35,X36);
\cim : fila3 , <>(X17,X18,X19,X27,X28,X29,X37,X38,X39);

\cim : fila4 , <>(X41,X42,X43,X51,X52,X53,X61,X62,X63);
\cim : fila5 , <>(X44,X45,X46,X54,X55,X56,X64,X65,X66);
\cim : fila6 , <>(X47,X48,X49,X57,X58,X59,X67,X68,X69);

\cim : fila7 , <>(X71,X72,X73,X81,X82,X83,X91,X92,X93);
\cim : fila8 , <>(X74,X75,X76,X84,X85,X86,X94,X95,X96);
\cim : fila9 , <>(X77,X78,X79,X87,X88,X89,X97,X98,X99);
```

Mediante estas restricciones controlamos que no haya valores repetidos en cada una de las filas.

```
#Restricciones para las columnas ##

\cim : columna1 , <>(X11,X14,X17,X41,X44,X47,X71,X74,X77);
\cim : columna2 , <>(X12,X15,X18,X42,X45,X48,X72,X75,X78);
\cim : columna3 , <>(X13,X16,X19,X43,X46,X49,X73,X76,X79);

\cim : columna4 , <>(X21,X24,X27,X51,X54,X57,X81,X84,X87);
\cim : columna5 , <>(X22,X25,X28,X52,X55,X58,X82,X85,X88);
\cim : columna6 , <>(X23,X26,X29,X53,X56,X59,X83,X86,X89);

\cim : columna7 , <>(X31,X34,X37,X61,X64,X67,X91,X94,X97);
\cim : columna8 , <>(X32,X35,X38,X62,X65,X68,X92,X95,X98);
\cim : columna9 , <>(X33,X36,X39,X63,X66,X69,X93,X96,X99);
```

Siguiendo la filosofía de las restricciones anteriores, mediante estas restricciones se controla que no haya valores repetidos en una misma columna. Además nos queda controlar que no haya variables repetidas dentro de una misma región. Del modo en que hemos definido las variables esto se puede hacer simplemente exigiendo que todas las variables XYZ con la misma Z (es decir en la misma región) sean diferentes. La implementación es la siguiente:

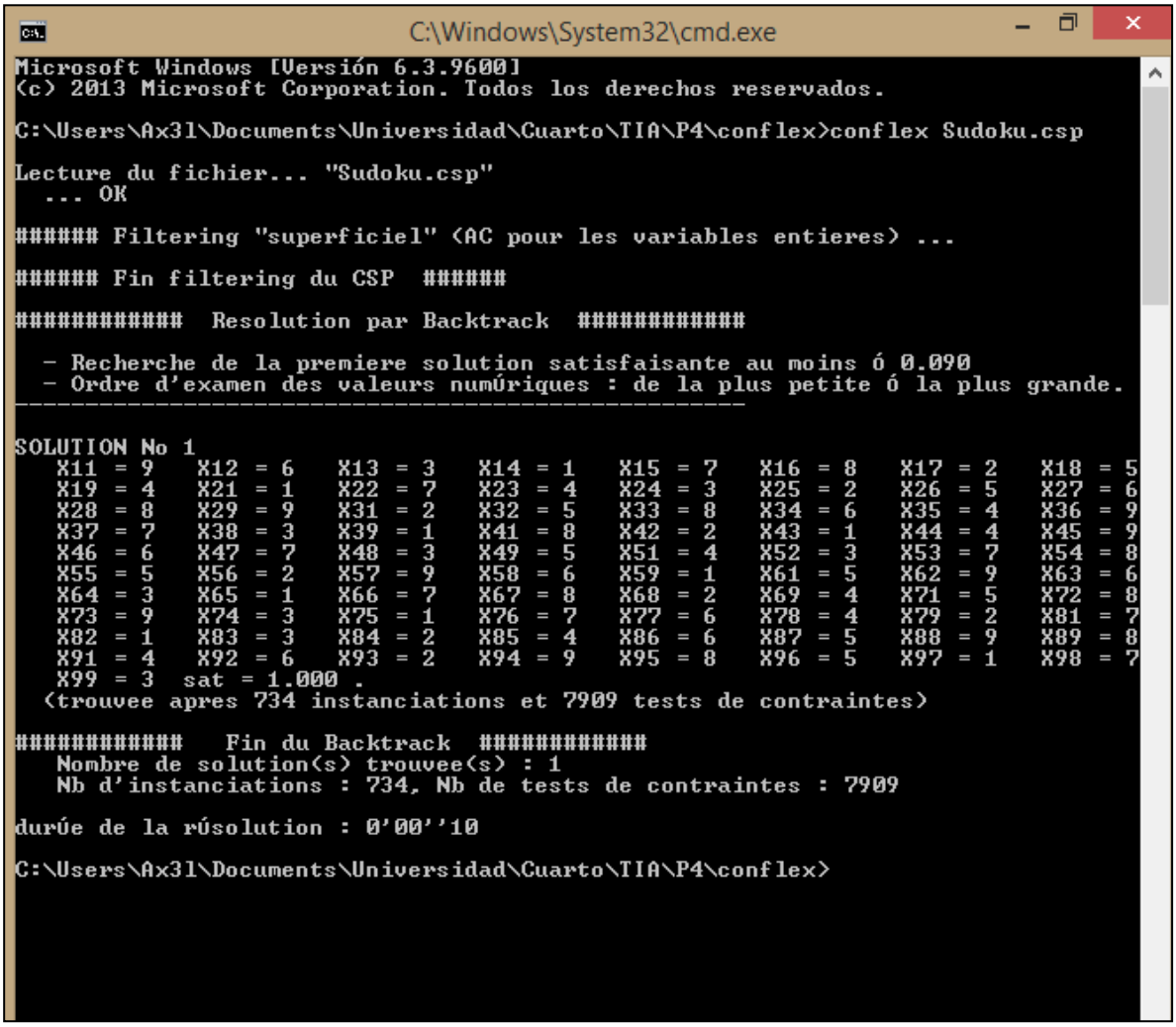
```
##Restricciones para las regiones ##

\cim : cuadricula1 , <>(X11,X12,X13,X14,X15,X16,X17,X18,X19);
\cim : cuadricula2 , <>(X21,X22,X23,X24,X25,X26,X27,X28,X29);
\cim : cuadricula3 , <>(X31,X32,X33,X34,X35,X36,X37,X38,X39);
\cim : cuadricula4 , <>(X41,X42,X43,X44,X45,X46,X47,X48,X49);
\cim : cuadricula5 , <>(X51,X52,X53,X54,X55,X56,X57,X58,X59);
\cim : cuadricula6 , <>(X61,X62,X63,X64,X65,X66,X67,X68,X69);
\cim : cuadricula7 , <>(X71,X72,X73,X74,X75,X76,X77,X78,X79);
\cim : cuadricula8 , <>(X81,X82,X83,X84,X85,X86,X87,X88,X89);
\cim : cuadricula9 , <>(X91,X92,X93,X94,X95,X96,X97,X98,X99);
```

A partir de éste punto ya tenemos las restricciones básicas que implementan las reglas del juego en una instancia de 9x9 casillas. Solamente nos queda añadir la información del problema en particular.

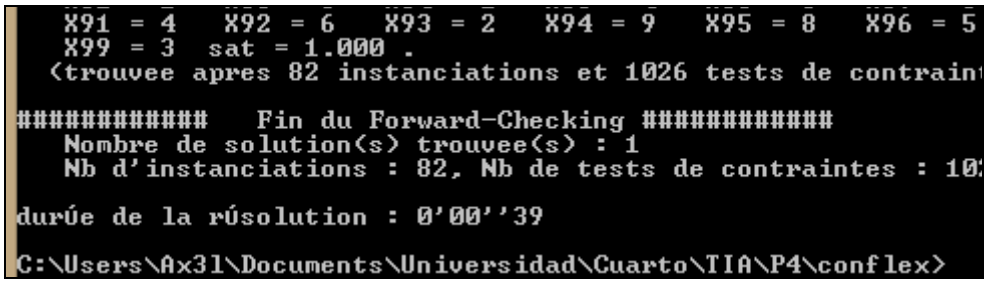
Vamos a modelar el Sudoku que se proporciona en el enunciado del boletín de prácticas. Para ello será suficiente con plantear las siguientes restricciones:

```
##Inicialización de valores conocidos
\ci : r1 , X12=6;
\ci : r2 , X16=8;
\ci : r3 , X17=2;

\ci : r4 , X21=1;
\ci : r5 , X23=4;
\ci : r3 , X24=3;
\ci : r3 , X26=5;

\ci : r3 , X32=5;
\ci : r3 , X34=6;
\ci : r3 , X39=1;

\ci : r3 , X41=8;
\ci : r3 , X46=6;
\ci : r3 , X47=7;

\ci : r3 , X51=4;
\ci : r3 , X53=7;
\ci : r3 , X57=9;
\ci : r3 , X59=1;

\ci : r3 , X63=6;
\ci : r3 , X64=3;
\ci : r3 , X69=4;

\ci : r3 , X71=5;
\ci : r3 , X76=7;
\ci : r3 , X78=4;

\ci : r3 , X84=2;
\ci : r3 , X86=6;
\ci : r3 , X87=5;
\ci : r3 , X89=8;

\ci : r3 , X93=2;
\ci : r3 , X94=9;
\ci : r3 , X98=7;
```

A partir de este punto ya tenemos modelado el problema. Vamos a realizar una pequeña prueba para comprobar los resultados (usando CONFLEX). Para esta prueba hemos utilizado el algoritmo de backtracking . El resultado es el siguiente:



Si comprobamos los valores podemos determinar que el resultado es correcto. El número de instancias no es excesivamente grande y el tiempo de ejecución es bastante reducido. En cambio si hacemos uso de algún algoritmo más informado como por ejemplo fc, los resultados cambian drásticamente, con un número de instancias mucho menor:



Cómo curiosidad también cabe destacar que sólo existe una solución al problema.

Cómo conlcusiones sobre el desarrollo de este problema podemos decir que resulta sencillo implementar este problema. Sin embargo es bastante laborioso sobre todo para instancias mayores que 9x9 por lo que es bastante recomendable automatizar la generación de variables (esto puede hacerse con scripts sencillos en python.). De hehco para este problema se han usado scripts sencillos en python para generar el código de las variables. También cabe destacar que es importante definir las varibales de manera que permitan comprobaciones sencillas.

Apartado 3 : Modelar un problema en CSP

Una vez más, en este apartado se pide modelar un problema en CSP de una lista de problemas propuestos, condicionados a la elección del problema en el apartado anterior. Para realizar este apartado se ha optado por desarrollar el problema de **"Invitados a una mesa" (3.16)** . El objetivo de dicho problema es distribuir a una serie de invitados en una mesa, satisfaciendo una serie de restricciones.

Para modelar este problema vamos a hacer uso de 18 variables , ya que hay 18 invitados que posicionar. Cada una de estas variables tomará un valor entre 1 y 18 , correspondiente a los asientos de la mesa. En la figura siguiente aparece la distribución de la mesa y su relación con los números del uno al dieciocho:

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18

Además de la posición que ocupa cada asistente , hay que tener en cuenta que los asistentes acuden en parejas hombre-mujer por lo que es interesante definir las variables de manera consecuente. Así se pueden definir variables de la forma HY MY. La "H" indica que se trata de un hombre y la "M" que se trata de una mujer. La "Y" puede tomar valor del 1 al 9 i indica el número de matrimonio. Por tanto si un hombre y una mujer tienen "Y" igual es porque forman parte del mismo matrimonio. Esta codificación de las variables ayuda a entender mejor el problema. Dicho esto sólo queda definir las variables.

La definición de las variables en Conflex será de la siguiente forma:

```
##############################
### VARIABLES ###
##############################

\vi : H1,H2,H3,H4,H5,H6,H7,H8,H9 1..18;
\vi : M1,M2,M3,M4,M5,M6,M7,M8,M9 1..18;
```

Ahora que ya tenemos definidas nuestras variables, podemos empezar a modelar las restricciones. El número de restricciones a este problema es bastante elevado y requiere una cantidad considerable de código.

Primero que nada vamos a empezar con una restricción elemental, que aunque no se indique en el problema se sobreentiende. Esta restricción es que dos comensales no pueden sentarse en el mismo sitio. Tal y cómo hemos modelado las variables significa que todas las variables deben tener un valor diferente. La definición de éste tipo de restricciones es muy sencilla y bastará con lo siguiente:

```
#Restricción básica: Dos comensales no pueden sentarse en el mismo sitio
\cim : restriccionSitio, <>(H1,H2,H3,H4,H5,H6,H7,H8,H9,M1,M2,M3,M4,M5,M6,M7,M8,M9);
```

Restricción 1: H2 no debe sentarse al lado de H1, M4, H5 ni H9.

Para controlar que un invitado no se siente al lado de otro bastará con comprobar que la diferencia entre el asiento de un invitado y el otro invitado en valor absoluto es distinta de uno . Si es distinto de uno y dado que dos invitados no pueden tener el mismo asiento (Por la restricción anterior), significará que no se sientan juntos. Esta restricción se implementa del siguiente modo:

```
#Restricción 1 : H2 no debe sentarse al lado de H1,
M4, H5 ni H9
\ci : restriccion1,  abs(H2-H1)<>1;
\ci : restriccion1,  abs(H2-M4)<>1;
\ci : restriccion1,  abs(H2-H5)<>1;
\ci : restriccion1,  abs(H2-H9)<>1;
```

Restricción 2: M3 no puede sentarse al lado de ninguna otra mujer.

De forma similar al razonamiento anterior , para saber si un invitado se sienta junto a otro basta con comprobar que el valor absoluto de la diferencia entre los invitados es uno . Por ejemplo si un invitado se siente en el asiento 6, los asientos a su lado son 5 y 7. |6-5|=1 y |7-6|=1. Aclarado este aspecto la implementación es la siguiente:

```
#Restricción 2: M3 no puede sentarse al lado de ninguna otra mujer.

\ci : restriccion2,  abs(M3-M1)<>1;
\ci : restriccion2,  abs(M3-M2)<>1;
\ci : restriccion2,  abs(M3-M4)<>1;
\ci : restriccion2,  abs(M3-M5)<>1;
\ci : restriccion2,  abs(M3-M6)<>1;
\ci : restriccion2,  abs(M3-M7)<>1;
\ci : restriccion2,  abs(M3-M8)<>1;
\ci : restriccion2,  abs(M3-M9)<>1;
```

Restricción 3 : H7 no puede sentarse al lado de ningún hombre, ni de M3.

En esta restricción tenemos que controlar que H7 no se siente al lado de ningún hombre ni de M3 , la definición es parecida a la anterior:

```
#Restricción 3: H7 no puede sentarse al lado de ningún
hombre, ni de M3

\ci : restriccion3,  abs(H7-H1)<>1;
\ci : restriccion3,  abs(H7-H2)<>1;
\ci : restriccion3,  abs(H7-H3)<>1;
\ci : restriccion3,  abs(H7-H4)<>1;
\ci : restriccion3,  abs(H7-H5)<>1;
\ci : restriccion3,  abs(H7-H6)<>1;
\ci : restriccion3,  abs(H7-H8)<>1;
\ci : restriccion3,  abs(H7-H9)<>1;
\ci : restriccion3,  abs(H7-M3)<>1;
```

Restricción 4 : H5 debe sentarse al lado de M4, M8, H2, H3 o H7. Pero no debe sentarse al lado de M5.

En esta restricción tenemos la aparición de un “o” , lo cuál significa que tenemos que modelar el problema de modo que sólo sea necesario que se cumpla una de las condiciones. Por tanto H5 podría sentarse al lado de cualquiera de los asistentes indicados. Para modelar esto haremos uso de restricciones condicionales. En cambio la condición de no sentarse al lado de M5 debe cumplirse en cualquier caso. El código propuesto es el siguiente:

```
#Restricción 4: H5 debe sentarse al lado de M4, M8, H2, H3 o H7. Pero no
debe sentarse al lado de M5.

\ci : restriccion3,  abs(H5-M5)<>1;

\doc: Doc1
      \ci : restriccion3 ,abs(H5-M4)=1;
      \or
      \ci : restriccion3 ,abs(H5-M8)=1;
      \or
      \ci : restriccion3 ,abs(H5-H2)=1;
      \or
      \ci : restriccion3 ,abs(H5-H3)=1;
      \or
      \ci : restriccion3 ,abs(H5-H7)=1;;
```

Restricción 5 : El matrimonio 1 no puede sentarse junto a ningún miembro del matrimonio 5, 6, 8, 9.

De nuevo una restricción en la que dos asistentes no pueden sentarse al lado. En este caso debe cumplirse para H1 y M1 con respecto a los miembros de cada uno de los matrimonios indicados:

```
#Restriccion 5 : El matrimonio 1 no puede sentarse junto
a ningún miembro del matrimonio 5, 6, 8, 9

##Restricciones de H1 frente a hombres

\ci: restriccion5, abs(H1-H5)<>1;
\ci: restriccion5, abs(H1-H6)<>1;
\ci: restriccion5, abs(H1-H8)<>1;
\ci: restriccion5, abs(H1-H9)<>1;

##Restricciones de H1 frente a mujeres

\ci: restriccion5, abs(H1-M5)<>1;
\ci: restriccion5, abs(H1-M6)<>1;
\ci: restriccion5, abs(H1-M8)<>1;
\ci: restriccion5, abs(H1-M9)<>1;
```

**Continuación de la restricción en la siguiente hoja

```
##Restricciones de M1 frente a hombres

\ci: restriccion5, abs(M1-H5)<>1;
\ci: restriccion5, abs(M1-H6)<>1;
\ci: restriccion5, abs(M1-H8)<>1;
\ci: restriccion5, abs(M1-H9)<>1;

##Restricciones de M1 frente a mujeres

\ci: restriccion5, abs(M1-M5)<>1;
\ci: restriccion5, abs(M1-M6)<>1;
\ci: restriccion5, abs(M1-M8)<>1;
\ci: restriccion5, abs(M1-M9)<>1;
```

Restricción 6: Al menos un miembro del matrimonio 4 debe sentarse junto a un miembro del matrimonio 1 o del matrimonio 9.

Para modelar esta restricción hay que tener en cuenta algunos aspectos. Cuando dice “al menos” significa que debe cumpliré que o bien H4 se sienta junto a algún miembro de los matrimonios especificados o bien M4. Esta parte se modela con una disyunción. La parte de un miembro del matrimonio 1 o del 9 es de nuevo una disyunción. El código para la restricción es :

```
#Restriccion 6 : Al menos un miembro del
matrimonio 4 debe sentarse junto a un miembro
del matrimonio 1 o del
#matrimonio 9

\doc Doc2
      \ci: restriccion6, abs(H4-H1)=1;
      \or
      \ci: restriccion6, abs(H4-M1)=1;
      \or
      \ci: restriccion6, abs(M4-H1)=1;
      \or
      \ci: restriccion6, abs(M4-M1)=1;
      \or
      \ci: restriccion6, abs(H4-H9)=1;
      \or
      \ci: restriccion6, abs(H4-M9)=1;
      \or
      \ci: restriccion6, abs(M4-H9)=1;
      \or
      \ci: restriccion6, abs(M4-M9)=1;;
```

Restricción 7 : Ningún miembro del matrimonio 7 u 8 puede sentarse en un extremo de la mesa.

Para modelar esta restricción vamos a tener en cuenta la forma en la que hemos modelado la posición de los asiento de la mesa. Por tanto para saber is un asistente se sienta en un extremo será suficiente con comprobar que tiene un valor distinto a 1,9,10 o 18. De acuerdo con este razonamiento se implementa el código de la siguiente forma:

```
#Restriccion 7 : Ningún miembro del matrimonio 7 u 8 puede sentarse en un extremo de la
mesa.

\ci: restriccion7, H7<>18;
\ci: restriccion7, H7<>1;
\ci: restriccion7, H7<>9;
\ci: restriccion7, H7<>10;

\ci: restriccion7, M7<>18;
\ci: restriccion7, M7<>1;
\ci: restriccion7, M7<>9;
\ci: restriccion7, M7<>10;

\ci: restriccion7, H8<>18;
\ci: restriccion7, H8<>1;
\ci: restriccion7, H8<>9;
\ci: restriccion7, H8<>10;

\ci: restriccion7, H8<>18;
\ci: restriccion7, H8<>1;
\ci: restriccion7, H8<>9;
\ci: restriccion7, H8<>10;
```

Restricción 8: Entre los miembros del matrimonio 4 o 9 deben haber al menos 3 personas.

Esta restricción se entiende cómo que tanto para el matrimonio 4 y el 9 la distancia entre sus miembros debe ser al menos tres personas:

```
#Restriccion 8 :Entre los miembros del matrimonio 4 o
9 deben haber al menos 3 personas

\ci: restriccion8, abs(H4-M4)>3;
\ci: restriccion8, abs(H9-M9)>3;
```

Restricción 9 : Entre los miembros del matrimonio 8, no puede ser mayor que 1 persona, pero tampoco deben estar juntos(Debe haber una persona entre los miembros del matrimonio 8).

Cómo ya hemos comentado anteriormente comprobar que dos asistentes se sientan al lado es comprobar el valor absoluto de la diferencia de sus valores. En este caso queremos que haya una persona entre el matrimonio por lo que el valor absoluto deberá ser igual a dos:

```
#Restriccion 9 : Entre los miembros del matrimonio 8, no puede ser mayor que 1 persona,
#pero tampoco deben estar juntos(Debe haber una persona entre los miembros del
matrimonio 8).

\ci: restriccion9, abs(H8-M8)=2;
```

Restriccion 10 :El matrimonio 1 deben sentarse juntos.

Para modelar esta restricción de nuevo haremos uso del valor absoluto que deberá ser igual a uno:

```
#Restriccion 10 :El matrimonio 1 deben sentarse juntos.

\ci: restriccion10, abs(H1-M1)=1;
```

Restriccion 11 : El matrimonio 2 debe estar cerca, pero no en los extremos. No debe haber más de dos personas hasta el extremo.

Esta restricción resulta un poco compleja de modelar. Se ha optado por especificar simplemente el rango de valores que puede tomar el matrimonio 2 en la mesa. Hay que tener en cuenta que se debe cumplir la condición tanto para el hombre como para la mujer. Para ello es suficiente con especificar el rango de valores permitido para el hombre y después el rango de valores permitido para la muejr. Se hace del siguiente modo:

```
#Restriccion 11 : El matrimonio 2 debe estar cerca, pero no en los
extremos. No debe haber más de dos personas hasta el
#extremo.

\doc: Doc5
\coc: C1 \ci: restriccion11, H2>1;
              \and
              \ci: restriccion11, H2<4;;
\or
\coc: C2 \ci: restriccion11, H2>6;
              \and
              \ci: restriccion11, H2<9;;
\or
\coc: C2 \ci: restriccion11, H2>10;
              \and
              \ci: restriccion11, H2<13;;
\or
\coc: C2 \ci: restriccion11, H2>15;
              \and
              \ci: restriccion11, H2<18;;
;
```

**Continuación de la restricción en la siguiente página

```
\doc: Doc4
\coc: C1 \ci: restriccion11, M2>1;
              \and
              \ci: restriccion11, M2<4;;
\or
\coc: C2 \ci: restriccion11, M2>6;
              \and
              \ci: restriccion11, M2<9;;
\or
\coc: C2 \ci: restriccion11, M2>10;
              \and
              \ci: restriccion11, M2<13;;
\or
\coc: C2 \ci: restriccion11, M2>15;
              \and
              \ci: restriccion11, M2<18;;
;
```

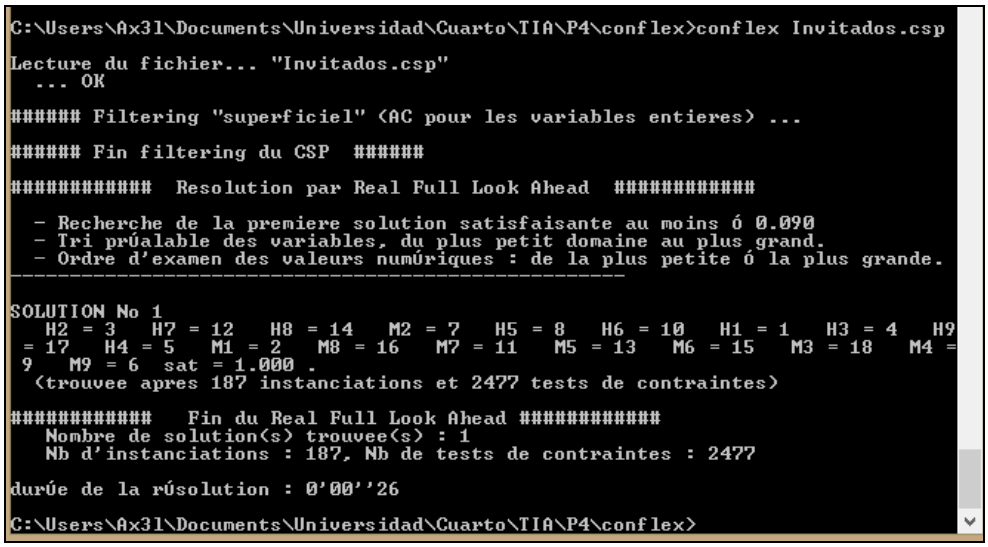
Restricción 12 : Cada hombre-i debe sentarse en un silla k, que debe ser posterior a la silla p donde se sienta cualquier hombre-j, si i>j. Es decir, si i>j, entonces k>p.

El significado de esta restricción es simplemente que los hombres deben sentarse de forma creciente en la mesa. Es decir , H1 debe ocupar un sitio anterior a H2 , este a su vez anterior a H3 y así hasta nueve. Afortunadamente definir esta restricción es muy rápido y se puede hacer en una línea. Mediante la siguiente restricción se indica a conflex que cada variable debe tener un valor menor que la siguiente:

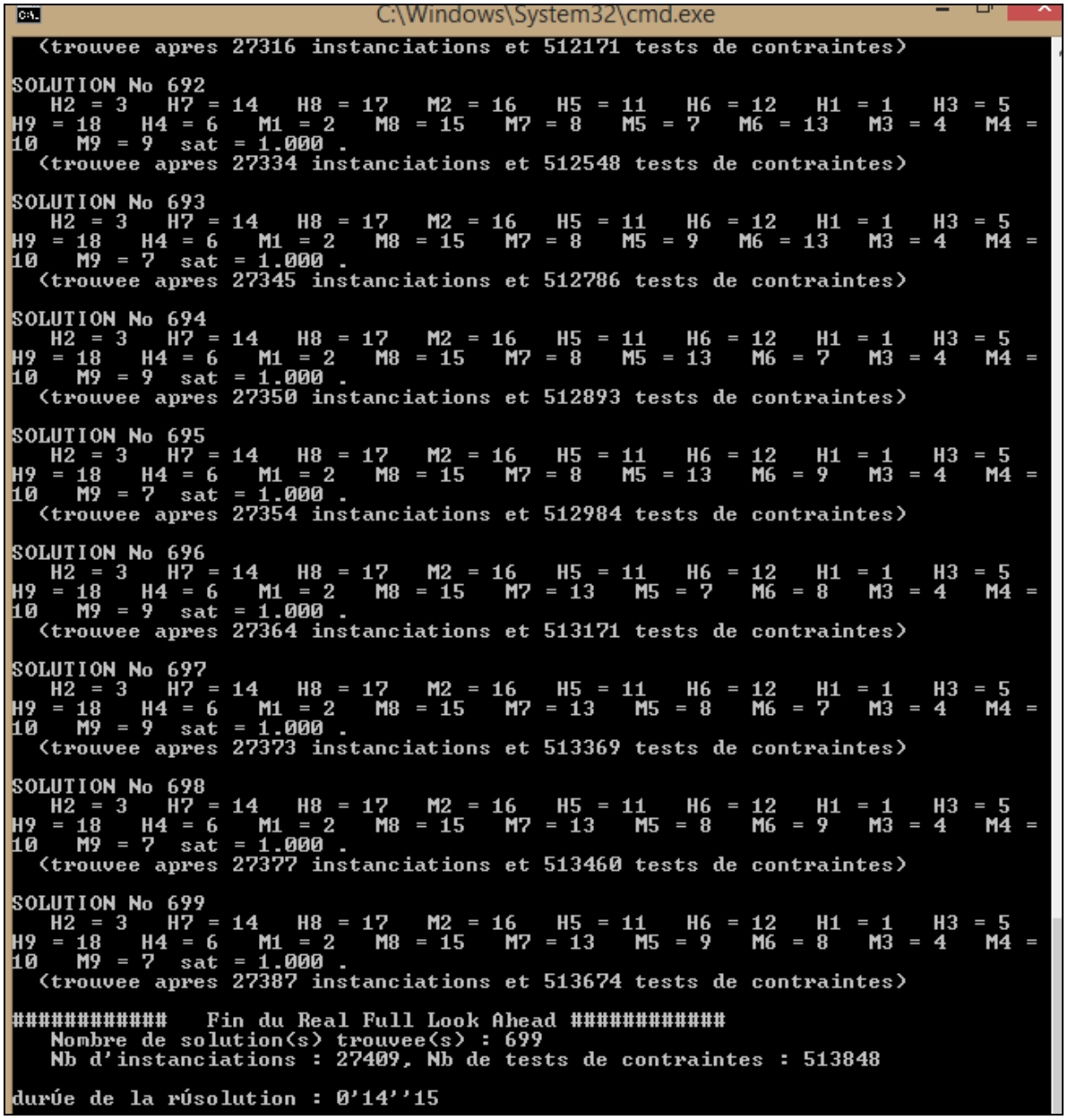
```
#Restriccion 12 : Cada hombre-i debe sentarse en un silla k, que debe ser
#posterior a la silla p donde se sienta cualquier hombre-j, si i>j. Es decir, si
#i>j, entonces k>p.

\cim: restriccion12 , < (H1,H2,H3,H4,H5,H6,H7,H8,H9);
```

Pruebas y evaluación de los resultados

Ahora que ya tenemos el sistema diseñado sólo queda ejecutarlo en conflex y observar los resultados obtenidos. La solución a este problema mediante backtracking resulta bastante costosa (no he conseguido llegar a ninguna solución por este método) por lo que se va a utilizar el algoritmo rfla. Los resultados son los siguientes:



Si se comprueba que se cumplen todos los requisitos puede observarse que la solución es correcta. Cabe destacar que con el algoritmo fc el tiempo de ejecución es casi el doble. Si ejecutamos una prueba para obtener todas las soluciones obtenemos lo siguiente:



Contrariamente al caso del Sudoku , en este problema existe una cantidad notable de soluciones.

Tras completar este apartado las conclusiones son diversas. Primero que nada el modelado del problema ha resultado laborioso pero no ha sido demasiado complicado conceptualmente. Ha resultado muy importante la manera de definir las variables y de modelar el problema tal y cómo se ha propuesto ya que de otra forma puede resultar mucho más difícil resolver el problema . También parece que el coste de los problemas aumenta con el número de restricciones. Sin embargo esto no siempre tiene por que ser así ya que algunos algoritmos pueden sacar ventaja de que haya mayor número de restricciones y podar ramas de soluciones prematuramente.

En cuánto a las conclusiones generales de la práctica, resulta una práctica que exige esfuerzo de comprensión y un tiempo considerable . La realización de esta práctica ha supuesto un total aproximado de 20 horas , incluida la memoria. Los mayores problemas han sido el uso del lenguaje en el que se

modelan los problemas. No obstante la práctica consigue su objetivo, pues para completarla hay que familiarizarse con los conceptos relacionados con CSP. Como opinión personal destacar que el apartado que me ha parecido más aburrido ha sido el primero, ya que requería un trabajo bastante monótono.