**Artificial Intelligence Assignment Report**

1. **Introduction**

Tetravex is a puzzle game where we have n\*n pieces that are divided in four small triangles that contain a number and a color (k colors in total). The objective of the game is organizing the pieces in such a way that only two equal numbers are together. The game ends when all the pieces have been positioned correctly.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza bajaImagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **Model**

The model was written in the tetravex.mzn file:

This part is the data that comes from the data files (.dzn) such as the dimension of the table and the pieces we must organize.

% Tetravex n\*n

int: n;

array[1..n\*n,1..4] of int: pieces;

set of int: N = 1..n;

This part indicates that the solution is a 2-dimensional array where one dimension is the number of pieces and the other the position (row and column) where each of them should be placed.

%positions of each piece in the table (solution)

array[1..n\*n,1..2] of var N: p;

Here start the constraints. The first one indicates that there cannot be two pieces in the same position (same row and same column at the same time). The rest of constraints indicate how the numbers must coincide to solve the puzzle

%No two pieces in the same position

constraint forall(i in 1..((n\*n)-1)) (forall(j in i+1..(n\*n)) (if p[i,1]==p[j,1] then p[i,2]!=p[j,2] endif));

%The piece at the right (piece j) must have the same number at the left as piece i has at the right

constraint forall(i in 1..(n\*n)) (forall(j in i+1..(n\*n)) (if p[i,1]==p[j,1] /\ p[j,2]==p[i,2]+1 then pieces[i,3]==pieces[j,1] endif));

%The piece at the left (piece j) must have the same number at the right as piece i has at the left

constraint forall(i in 1..(n\*n)) (forall(j in i+1..(n\*n)) (if p[i,1]==p[j,1] /\ p[j,2]==p[i,2]-1 then pieces[i,1]==pieces[j,3] endif));

%The piece at the bottom (piece j) must have the same number at the top as piece i has at the bottom

constraint forall(i in 1..(n\*n)) (forall(j in i+1..(n\*n)) (if p[i,2]==p[j,2] /\ p[j,1]==p[i,1]+1 then pieces[i,4]==pieces[j,2] endif));

%The piece at the top (piece j) must have the same number at the bottom as piece i has at the top

constraint forall(i in 1..(n\*n)) (forall(j in i+1..(n\*n)) (if p[i,2]==p[j,2] /\ p[j,1]==p[i,1]-1 then pieces[i,2]==pieces[j,4] endif));

Here we are indicating that we want a solution satisfying the constraints and how the solution will be displayed, in this case indicating in which position of the table should each piece go.

% Find a solution that satisfies the constraints

solve satisfy;

output [ *"Piece \(i): [\(pieces[i,1]),\(pieces[i,2]),\(pieces[i,3]),\(pieces[i,4])] in position [\(p[i,1]),\(p[i,2])]\n"* | i in 1..(n\*n) ];

1. **Data examples and results**

The data was written in .dzn files:

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamentetetravex.dzn (3\*3):**

n = 3;

%1st number: left, 2nd number: top, 3rd number: right, 4th number: bottom

pieces = array2d(1..n\*n,1..4,

[4,1,9,6, 6,1,4,0, 6,9,9,1,

0,5,1,5, 6,0,6,9, 4,5,6,4,

3,4,6,4, 1,7,6,1, 6,6,2,9

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente**Texto

Descripción generada automáticamente ]);

Initial pieces

Solution in Minizinc

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamentetetravex2.dzn (4\*4):**

n = 4;

%1st number: left, 2nd number: top, 3rd number: right, 4th number: bottom

pieces = array2d(1..n\*n,1..4,

[8,0,7,2, 7,0,3,2, 5,8,2,3, 7,3,5,0,

4,2,8,2, 2,5,5,1, 5,5,2,7, 3,3,9,1,

7,1,6,9, 7,3,6,0, 2,2,5,6, 7,4,7,7,

8,7,7,5, 5,1,7,3, 2,2,7,6, 6,1,5,4,

]);

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteTabla

Descripción generada automáticamente**

Initial pieces

Solution in Minizinc