

The Hidato (Hidoku) Puzzle (hidato)

— Run LPL Code , HTML Document —

Problem: Hidato, also known as Hidoku (or “Number Snake”, “Snapepit”) is a logic puzzle game invented by Gyora M. Benedek. The goal of Hidoku is to fill the cells of a grid with consecutive numbers that connect horizontally, vertically, or diagonally.

Figure 1 gives a small example. In the left part, the puzzle with 4 predefined entries is shown. Complete the empty cells in such a way that a path can be drawn as shown at the right part.

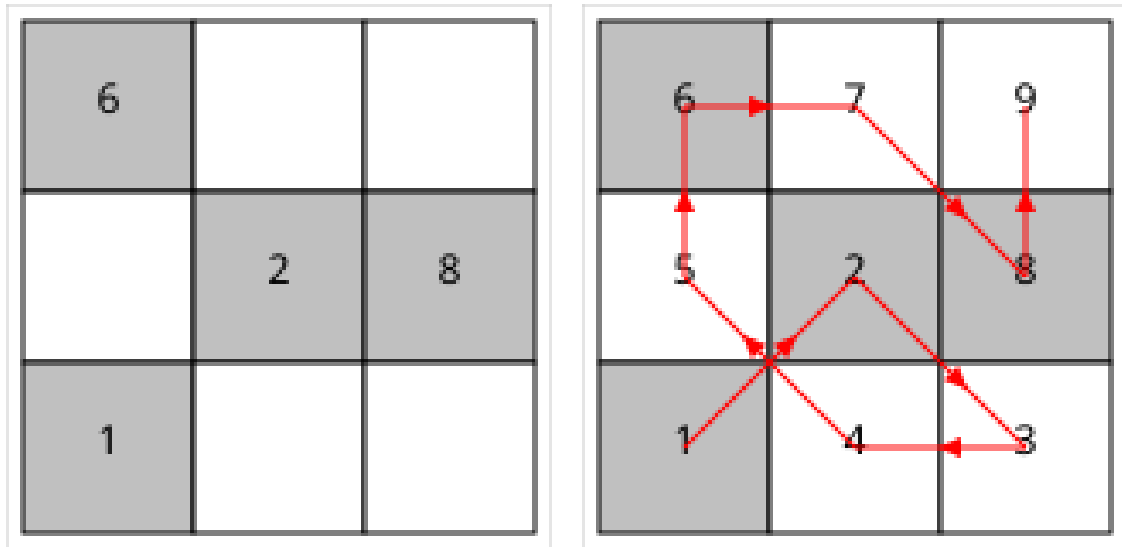


Figure 1: A Hidato Puzzle and its Solution

Modeling Steps: The problem can be formulated as a variant of the TSP problem (see [tsp-1](#)). As starting point we use the Miller-Tucker-Zemlin formulation (see [\[2\]](#)).

Let $i, j \in I = \{1, \dots, n\}$ be a set of cells in a grid (with $n > 0$). The cells are numbered from left to right and from top to bottom. Hence in the small 3×3 grid in Figure 1, the top-left cell is number 1 and the bottom-right cell is number 9. Two cells i and j are “connected” to each other if they are horizontal, vertical, or diagonal neighbours. A relation $e_{i,j}$ is introduced that is *true* (1) if and only if i and j are neighbours.

For drawing the puzzle, the coordinates (X_i, Y_i) of each cell i are needed. The coordinates can also be used to define $e_{i,j}$: Supposing the size of a cell is 1×1 then the relation $e_{i,j}$ can be given as :

$$e_{i,j} = (0.01 \leq \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \leq \sqrt{2} + 0.01) \quad \text{for all } i, j \in I$$

An further connection must be added to satisfy a round trip: The ending cell must be linked to the starting cell. We suppose that the starting cell is always given, if the ending cell is not defined (as in the small example above), then a connection must be added to $e_{i,j}$ for *all* cells with $P_i = 0$ to the starting node.

Furthermore, the numbers inside the predefined cells is defined by the vector P_i . In the small example above we have (where 0 means “not occupied”) :

$$P_i = (6, 0, 0, 0, 2, 8, 1, 0, 0)$$

Two other parameters are introduced: mi and ma which are the index of the starting and the index of the ending cell of the path (in the example we have: $mi = 7$).

A binary variable $x_{i,j}$ is introduced, which has value 1, if connection (i, j) is in the path. Otherwise – if the connection (i, j) is not in the path – then $x_{i,j} = 0$. Let's also introduce a variable $0 \leq u_i \leq n$ for every cell i (we call this variable the potential).

Now the problem can be formulated as follows :

$$\begin{aligned}
 \min \quad & \sum_{i,j \in I} x_{i,j} \\
 \text{subject to} \quad & \sum_{j \in I} x_{i,j} = 1 && \text{forall } i \in I \\
 & \sum_{i \in I} x_{i,j} = 1 && \text{forall } j \in I \\
 & u_i - u_j \leq (1 - x_{i,j})n - 1 && \text{forall } i, j \in I, j \neq mi \\
 & u_i = P_i - 1 && \text{forall } i \in I, P_i \neq 0 \\
 & i, j \in I = \{1 \dots n\}, n > 0
 \end{aligned}$$

The first and second constraints assert that the path can leave or enter each cell exactly once.

The third constraint is the famous Miller-Tucker-Zemlin condition (it has been explained in the model [tsp-1](#)). The constraint basically ensures the connectivity of the path. and it assigns an increasing number to the cells in the path. But we need two notable modifications: (1) The starting cell is the cell with the index mi . Hence, the constraint is not valid for the final path link (i, mi) – the value of u_{mi} is 0 but the value u_i of the ending cell is at its maximum, so the value cannot increase. (2) We limit the upper bound of u_i to $n - 1$, So the number 0 to $n - 1$ are assigned to the u_i increasingly from the starting cell. This is an important trick because in this way u_i is exactly the path length of cell i from the starting cell.

The final constraint predefines the position of some cells – for which $P_i \neq 0$ – and this is exactly defined by u_i .

Listing 1: The Main Model implemented in LPL [3]

```

model hidato "The Hidato (Hidoku) Puzzle";
  set i,j "size of the grid (number of cells!);
    e{i,j} "possible edges in grid (connecting cells)";
  parameter P{i}; mi; ma;
  binary variable x{i,j|e};
  variable u{i} [0..#i-1];
  constraint
    A{i}: sum{j} x = 1;
    B{j}: sum{i} x = 1;
    C{i,j|j<>mi}: u[i] - u[j] <= (1-x)*#i - 1;
    D{i|P}: u = P-1;
  minimize obj: sum{e[i,j]} x;
end

```

The data are read from the file [hidato.txt](#). 8 instances are defined in the file. They can be ran by changing the value of the ID parameter to the values $\{1, \dots, 8\}$. The data are read by the data model :

Listing 2: The Data Model

```

model data;
  parameter n; m; ID:=1; // puzzle ID
    X{i}; Y{i}; //center of a cell
  string typ;

```

```

Read('hidato.txt,%&ID&':Table', n,typ);
i:=1..n;
if typ='square' then
  m:=Sqrt(n);
  {i} (X:=(i-1)%m+1, Y:=Trunc((i-1)/m)+1);
  Read('%&ID&';1', {i} P);
else
  Read('%&ID&';1', {i} X);
  Read('%&ID&';2', {i} Y);
  Read('%&ID&';3', {i} P);
end
mi:=argmin{i|P}P; ma:=argmax{i}P;
e{i,j} := 0.01 <= Sqrt((X[i]-X[j])^2 + (Y[i]-Y[j])^2)
          <= Sqrt(2)+.01;
if P[ma]=P[mi]+#i-1 then e[ma,mi]:=1
else {i|~P} (e[i,mi]:=1); end;
end

```

Solution: As expected from the Miller-Tucker-Zemlin TSP-formulation, we can solve problems up to about $n = 45$. Hence problems 3 and 4 have not been solved with Gurobi 10 after 15mins.

The output model draws the puzzle and its solution:

Listing 3: The Output Model

```

model output friend data;
integer parameter Z{i}; W{i}; PP{i}; p;
{i} (p:=argmax{j} x[p,j], Z[i]:=p); //create the permutation
{i} if(Z=mi,p:=i); //find the starting cell
{i} if(i<p,W[i+#i-p+1]:=Z,W[i-p+1]:=Z); //shuffle
p:=1;
{i} (PP[W]:=p, p:=p+1);
Draw.Scale(50,50);
--{i} Draw.Rect(if(P,P&' ','')&'X,Y,1,1,if(P,2,1),0);
{i} Draw.Rect(PP&' ',X-.5,Y-.5,1,1,if(P,2,1),0);
{i|i>1} Draw.Arrow(X[W[i-1]],Y[W[i-1]],X[W],Y[W],.5,3);
end

```

Further Comments: Figures 2 and 3 display four Hidato puzzles. They are considered as very hard (See here). They correspond to the instances 5, 6, 7, and 8 in the data file. Figure 4 is a Hexagonal puzzle (see model hidato1).

Questions

1. The puzzles need not to be defined in a square grid. IT can take any form, see data instance 2.
2. The cells need not be squares. Many puzzles exist with hexagonal cels. How does this change the model formulation?
3. The puzzle also exists with hexagonal cells. Implement it.

Answers

1. Run the model with ID:=2

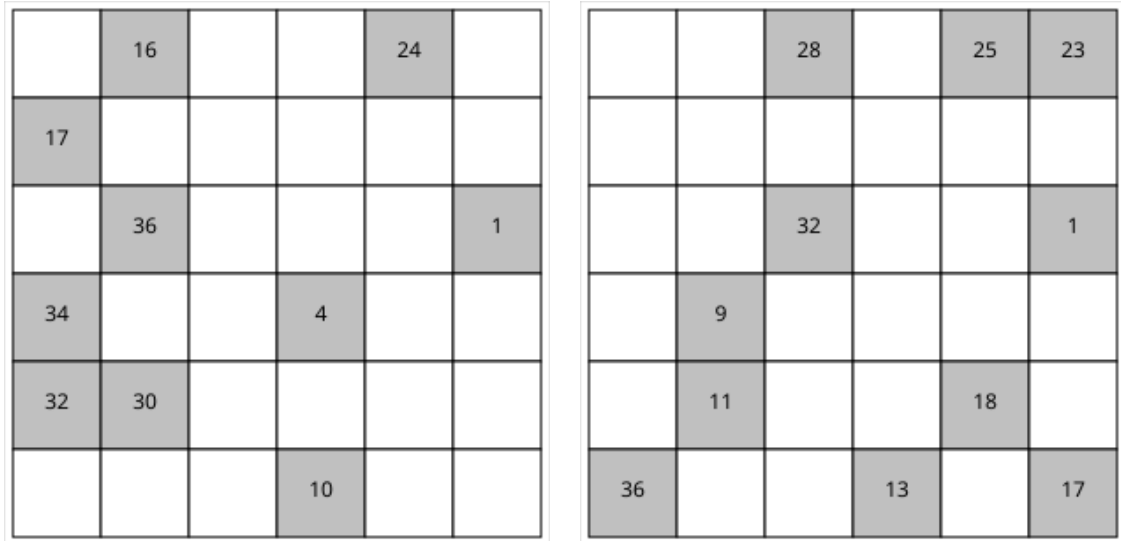


Figure 2: Two Hidato Puzzles

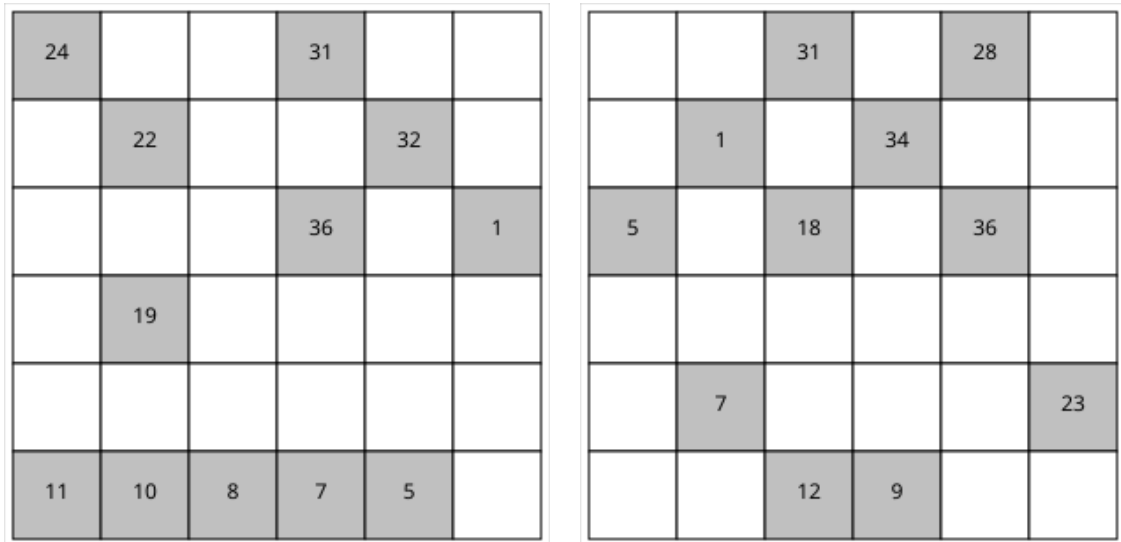


Figure 3: Two Hidato Puzzles

2. The model is exactly the same. The only change is in the calculation of the relation of the connection $e_{i,j}$.
3. An implementation in LPL is model [hidato1](#)

References

- [1] MatMod. Homepage for Learning Mathematical Modeling : <https://matmod.ch>.
- [2] Zemli R.A. Miller C.E., Tucker A.W. Integer Programming Formulation of Traveling Salesman Problems. *Journal of the ACM*, 7:326–329, 1960.

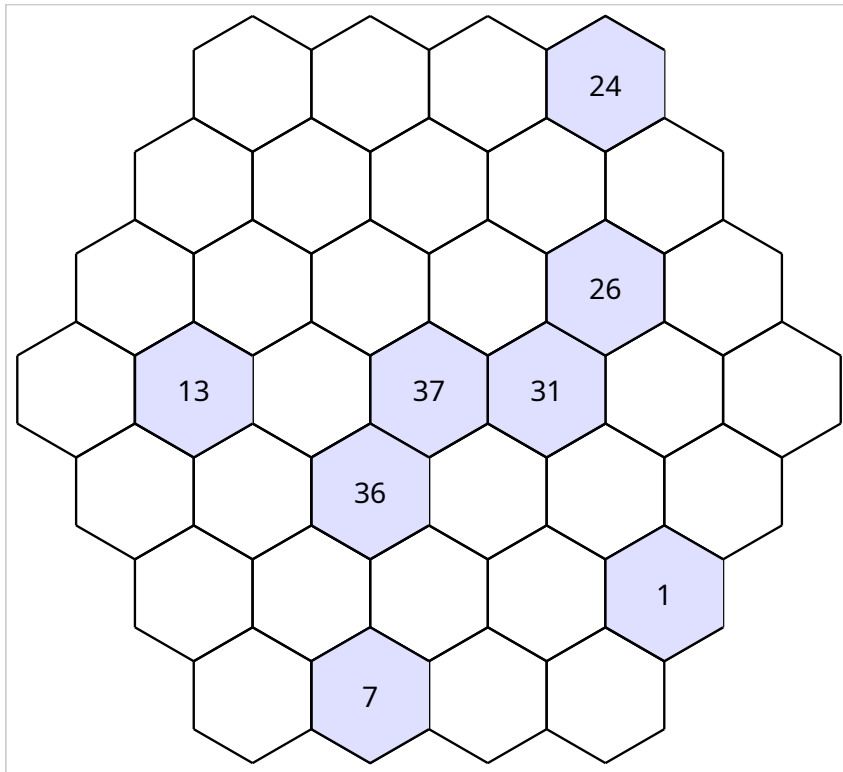


Figure 4: A Hexagonal Puzzle

- [3] Hürlimann T. Reference Manual for the LPL Modeling Language, most recent version.
<https://matmod.ch/lpl/doc/manual.pdf>.