#5 Continuous line - Formulation

Author: Eric Zettermann
July, 2021

This is the MIP formulation of the puzzle. Statement and solution implementation of all puzzles are available from the main page of the Fun Puzzles project, which is maintained by Mip Master.

Formulation

In this puzzle you are challenged to draw a continuous line inside the grid in a way that the line visits every empty cell exactly once. To make it harder, the grid has some obstacles (filled cells) that your line can't touch. This puzzle can be quite easily solved by hand and it has multiple solutions. We wonder here weather this puzzle can be solved using MIP. What do you think? Of course it can! Let's see how we can solve it.

 Continuous Line

Even before we dive into the formulation, keep in mind that we are going to solve a different problem. Wait, what does it mean? No worries... Everything is gonna be fine! Sometimes, it's useful--or even required--to reframe the problem to be able to solve it. After reframing, the problem might look different but the solution will still be the same. All right, now that you might be more comfortable with this idea, let's get back to work!

Here is the deal. Instead of thinking how to draw a continuous line trough the grid, we are going to think about filling that same grid with integers. It goes like this: we place 1 in the first cell we want to visit, then 2 in the second cell, then 3 in the third cell, and so on. We only need to make sure to place $n+1$ in a cell that is adjacent to the cell where we placed n. Once all cells are filled in with numbers, it will be easy to draw a continuous line, don't you think so? I do! So, let's begin.

Input Data

We begin by defining the characteristics of our grid:

- Number of rows:
  `num_rows=6`
- Number of Columns:
  `num_cols=6`
- The holes (obstacles):
  `H = [(0, 2), (1, 4), (3, 2), (3, 3), (4, 0), (4, 3), (4, 5), (5, 0), (5, 5)]`

With this information, we can determine how many digits we are going to need to fill the grid. Just subtract the number of holes (9 in our case) from the total number of cells in the grid (6x6=36 in this case): `num_digits = num_rows*num_cols-len(H)`

Note that we could just set `num_digits=36-9=27`. However, it's a good practice to avoid hard-coding and keep the data as separated as possible from the model. In fact, if we decide to change the data of this problem, we will only need to update the three parameters: `num_rows`, `num_cols`, and `H`. And we don't need to touch the rest of the code! Much better, right?

Now, we define some lists to help with the implementation of the model. Here is our opportunity to start using the parameters defined before.

- rows indices:
  `I = [1, 2, 3, ..., num_rows]`
- columns indices:
  `J = [1, 2, 3, ..., num_cols]`
- digits:
  `K = [1, 2, 3, ..., num_digits]`
- empty cells:
  `C= [(i,j) for i in I for j in J if (i, j) not in H]`
  Note that for building C, we just take out the holes (H) from de 6x6 grid.

We will assign decision variables to every empty cell, one variable for each digit in K. So let's go ahead and define a list of triples that will help us to define the decision variables later: `keys = [(i, j, k) for i, j in C for k in K]`

Decision Variables

There is one type of decision to make: What digit goes in each empty cell?

Or putting it in a different way, we need to decide (well, not we, the solver) whether a given digit should go in a given cell or not?

So we define one set of binary variables:

- x_{ijk} equals 1 if digit k goes in cell (i, j), 0 otherwise.

Constraints

Three constraints are required to model this puzzle.

- *Exactly one digit gets assigned to every cell*:

$$\sum_k x_{ijk} = 1, \quad \forall (i,j) \in C$$

  As all variables are binary, this constraint is forcing that only one of the variables inside the sum get set to 1. Let's make it clearer with an example. Consider the cell $(2, 3)$. We hope that only one digit gets assigned to it, right?
  So, cross your fingers and let's see how our constraint works in this case:

$$x_{2,3,1} + x_{2,3,2} + x_{2,3,3} + \cdots + x_{2,3,27} = 1.$$

All these variables are assigned to cell $(2, 3)$ and each one represents a different digit (the third index). So, if this sum must be 1, this means that only one of them must be 1 and the others must be zero. In other words, only one digit will be assigned to cell $(2, 3)$, exactly as we wanted. Hi five!

- *Every digit must be used exactly once*:

$$\sum_{(i,j)\in C} x_{ijk} = 1, \quad \forall k.$$

Here we use the same strategy adopted in the first constraint, except that we choose one digit at a time and force it to appear exactly one across all empty cells. If you are struggling to understand this constraint, choose one digit and expand the constraint to see what is happening, like we did before.

- *Every digit must have its consecutive in an adjacent cell*:

$$x_{i,j,k} \leq x_{i+1,j,k+1} + x_{i-1,j,k+1} + x_{i,j+1,k+1} + x_{i,j-1,k+1}, \quad \forall k, \forall (i,j) \in C.$$

All right! This is the soul of our formulation! Remember that we want to draw a continuous line, right? But, instead of drawing a line, we are filling the grid with integers--and then draw the line by tracking the sequence of numbers. For this to work, the numbers must define a contiguous sequence across the grid. This means that we must know where to go next from every single cell, by just looking at the number filled in the neighboring cells. That's what this constraint is about. It guarantees that, if k goes into cell (i, j), then the next number, $k + 1$, must go in the cell above $(i + 1, j)$, below $(i - 1, j)$, to the left $(i, j - 1)$, or to the right $(i, j + 1)$ of cell (i, j). Example pleeeeease! Sure! Here we go... Consider the number $k = 10$ and the cell $(2, 3)$ again. The constraint will then be:

$$x_{2,3,10} \leq x_{3,3,11} + x_{1,3,11} + x_{2,4,11} + x_{2,2,11}.$$

Here, we have two cases: the number 10 may or may not be in the cell $(2, 3)$. If it is not, then $x_{2,3,10}$ is zero. In which case, the left-hand-side can be one (if number 11 is in the neighborhood) or it can be zero (if 11 is not in an adjacent cell), but it doesn't matter for us. On the flip side, if 10 is in fact in cell $(2, 3)$, and this is the case that matters, then $x_{2,3,10}$ must be one. The constraint will then force the right-hand-side to be one as well (note that the right-hand-side can't be more than one because of the first constraint). If the left-hand-side must be one, it means that the number 11 must be in a cell adjacent to $(2, 3)$.
Obs.: Note that the cell $(3, 3)$, that appears in the right-hand-side in our example, is not an empty cell, because it composes one of the obstacles. You need to take care of this fact in your implementation, ok?

Objective Function

There is no objective to maximize or minimize in this problem. We only need to find one feasible solution. But it doesn't hurt if we define an objective function like this (it can actually help the solver!):

$$\min \sum_{i,j,k} x_{ijk}.$$

Final Formulation

$$
\begin{aligned}
\min \quad & \textstyle\sum_{i,j,k} x_{ijk} \\
\text{s.t.} \quad & \sum_{k} x_{ijk} = 1, \quad \forall (i,j) \in C, \\
& \sum_{(i,j)\in C} x_{ijk} = 1, \quad \forall k, \\
& x_{i,j,k} \le x_{i+1,j,k+1} + x_{i-1,j,k+1} + x_{i,j+1,k+1} + x_{i,j-1,k+1}, \quad \forall k, \forall (i,j) \in C. \\
& x_{ijk} \in \{0,1\}, \quad \forall i,j,k.
\end{aligned}
\tag{1}
$$