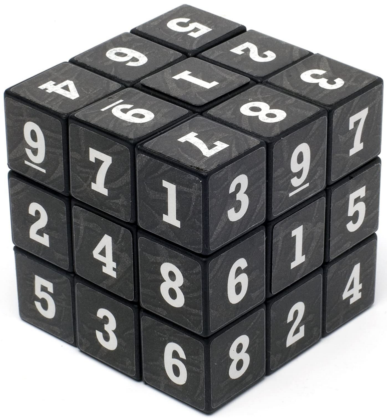
Solving the Sudokube



I've had this puzzle for a few years. Pretty immediately (using a piece of paper and a pencil) I was able to solve it with only a few hours of thinking. Recently, I started to wonder how does this problem space change as I add in constraints and so I turned to a little python. I'm trying to "solve" my own variation on this puzzle in stages to understand the rarity of a solvable cube. I am **not** trying to solve this cube using a computer; my puzzle is inspired.

Each face will have the number 1-9 in no particular order. Unlike sudoku, the "rows" do not matter because orientation is all over the place. It is key to note that on any given face, all the numbers face the same way.

Base line questions:

Q: How many valid sudoku 3x3 faces are there?
A: 9!

Q: How many correct solutions to a given sudoku-rubix-cube (sudokube) ?
A: 1. For the one that I have, and that's what I'm focusing on.

Now the interesting questions.

Q1: How many valid faces are there to the sudokube? That is, using only the numbers on my cube how many different faces can I make that are 1-9 and properly oriented. I only make one face at a time so I don't have to worry about numbers being used on a different face first.
A: 1865.[1]

This is *a lot* smaller than 9! Obviously. That makes sense because it introduces a very limited constraint in the numbers available on the cube. If we additionally constrain by realizing that the 1, 3, and 7 (on the visible corner in the picture above) are all on the same piece, then we

[1] How did I get 1,865? Good question. I recorded the cube. Using the visible faces in the picture at the top as an example: No matter how scrambled the sudokube is you can always tell that there will be two 1s and a 4 at the centers. Based on orientation you know that the top center piece of the solved cube has to be 7, 8, or 9. And so on. When I take all the numbers for every position on the face (Top Left through Bottom Right), I get the 6 options for each position. Most of these options aren't unique, like the centers in this example. It is easy enough to brute force the combinations and get that 1,865 of them don't have two of the same number in a single face. If I treat the duplicated numbers as unique (ex. the center can be 1, 4, or 1') that gives 16,720 possible combinations that don't have X and X or X and X' in the same face.

find that the single solution. 1 << 1865 <<< 9!, but what other numbers can I find to benchmark the way?

Q2: From the 1865 face options, how many sets of 6 *might* solve the cube? That is, if I still ignore that the 1, 3, and 7 have a physical, geometric relationship and just treat them as three separate “pieces” how many sets of 6 faces use all of the pieces available without reusing pieces.
A: I don’t know. I suspect it is a lot smaller than $\binom{1865}{6}$ because some faces require pieces from others but I don’t know how much, and that’s what’s interesting me.

Insights So Far

(1) Although most of the positions on the cube have repeated numbers so a completed solution will have two faces with a 1 at the center, for example, there is one position that is completely unique. Because the Top Middle position can only be a {3, 4, 5, 7, 8, or 9} then I know that if I group possible faces by this position, then I only need to choose one face that has a 3, one with a 4, etc. This means that instead of a full $\binom{1865}{6}$ this simplifies to combining 6 elements, one from each of 6 buckets (each of which is conveniently $\approx \frac{1865}{6}$). $\left(\frac{1865}{6}\right)^6$ is better but still taking my poor computer forever.

(2) After separating everything into buckets I realized that since everything is based on a limited pool of options, after determining 5 of the faces then the 6th one is just the pieces not yet used. Either those pieces are a valid face, which is easy to check, or they aren’t, in which case I can move on from the determined 5. This reduces the number of possible sets to $\left(\frac{1865}{6}\right)^5$.

(3) I try to be smart about which sets I’m checking. If I pick from the 3 bucket and the next 4 bucket item doesn’t work with the 3 (they demand the same options that aren’t repeated), then I skip that 4 bucket item and move on. I don’t know how to show this mathematically, but I’ve found that for my cube about 1/3 of all 3 bucket - 4 bucket combinations are valid and should be considered for checking for 5 bucket and so on. Similarly with choosing from any two buckets; I don’t know why but it seems to be all within 5%.

Strategies

- The (1) bucket insight came from when we were dealing with hashes. It isn’t a true hash but the idea of traversing a smaller bucket for options was so appealing I knew I must find a way to use it. All the other positions have at least one duplicate option so “hashing” on multiple positions is progressively less beneficial. Might be worth it if that will continue to bring down the total
- With our talking about graphs I realized that this has an analogue to graph theory. If each vertex is one of the 1865 faces and two vertices are connected if they are compatible with each other, I’m trying count all the full connected sub graphs size 6. I looked into that algorithm and it’s NP-Hard (worse than choosing sets size k from set of n) BUT!! If the graph is bipartite or planar I think it could be poly-time. Bipartite is right out, but planar could be

possible. A graph is only non-planar if it contains subgraphs K4,5 (4 vertices connected to each of 5 other vertices) or K6 (fully connected 6 vertices). I'm unfortunately trying to count the K6s in the overall set of 1865 faces, if I use the same trick as (2) where I only search for 5 and try and create the 6th based on the options that remain, then I can guarantee that no K6 subgraph is included. I'm pretty sure that K4,5 is present. So *probably* not planar.

Because this puzzle is tangentially connected to combinatorix, graph theory, and a little obscure algorithm analysis, I've really enjoyed turning it around on all sides. It's frustrating that I can find all the good 3bucket-4bucket pairs in under a second and all the good 3bucket-4bucket-5bucket triples in about 10 seconds, but trying to add in the 4th one hasn't finished and takes at least 2 minutes, and I haven't even tried 5 buckets together. Thank you for reading this far.