Corral Puzzles are NP-complete

Erich Friedman
Stetson University, DeLand, FL 32723
efriedma@stetson.edu

Introduction

Corral puzzles are pencil and paper puzzles which originated in Japan [8]. Each puzzle consists of a grid of squares, some of which contain numbers. The goal is to find a closed loop containing some of the grid squares so that all the numbers are inside the loop and each number is equal to the number of grid squares visible from it. A grid square is visible from a number if it is horizontally or vertically aligned with it and all the squares between them are inside the loop. An example of a Corral puzzle and its solution are shown in Figure 1.

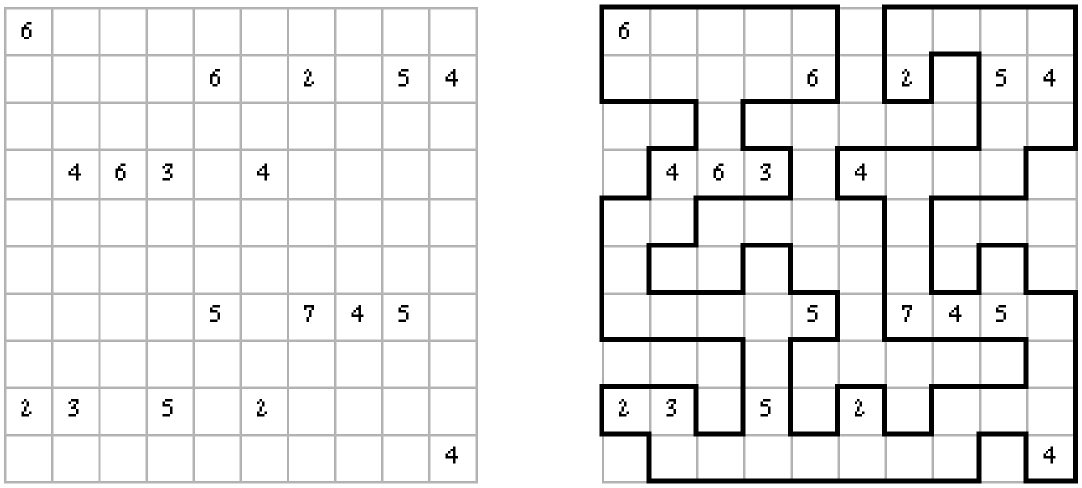

Figure 1. A Corral puzzle (left) and its solution (right)

To build Corral puzzles that correspond to planar graphs we will need several gadgets that we explain in later sections. We need "wires" capable of transmitting 3 different signals (corresponding to the 3 different colors of vertices), a "fork gadget" that splits a signal (so that all edges incident to a vertex carry the same color signal), a "differ gadget" that forces two incoming signals to be different (so that adjacent vertices must be different colors), a "turning gadget" to bend and delay wires (so that our other gadgets can be connected arbitrarily), and "walls" to contain our gadgets.

From these building blocks, any given planar graph can be constructed using Corral components. A fork gadget has 3 inputs, so we need to place d-2 fork gadgets at each vertex of degree d. On each edge, we place a differ gadget. See Figure 2 for an example of a Corral blueprint corresponding to a 3-colorable planar graph.

We will show that the question of whether or not a given Corral puzzle has a solution is NP-complete. To do so, we construct Corral puzzles which correspond to arbitrary planar graphs. The Corral puzzle we construct will have a solution if and only if the vertices of the corresponding planar graph can be 3-colored so that adjacent vertices are colored differently. Since 3-colorability of planar graphs is known to be NP-complete [6], this will show that Corral puzzles are NP-hard. We complete the proof by showing that a solution to a Corral puzzle can be checked in polynomial time. Similar approaches to proving puzzles are NP-complete are taken in [1, 2, 4, 5, 7].

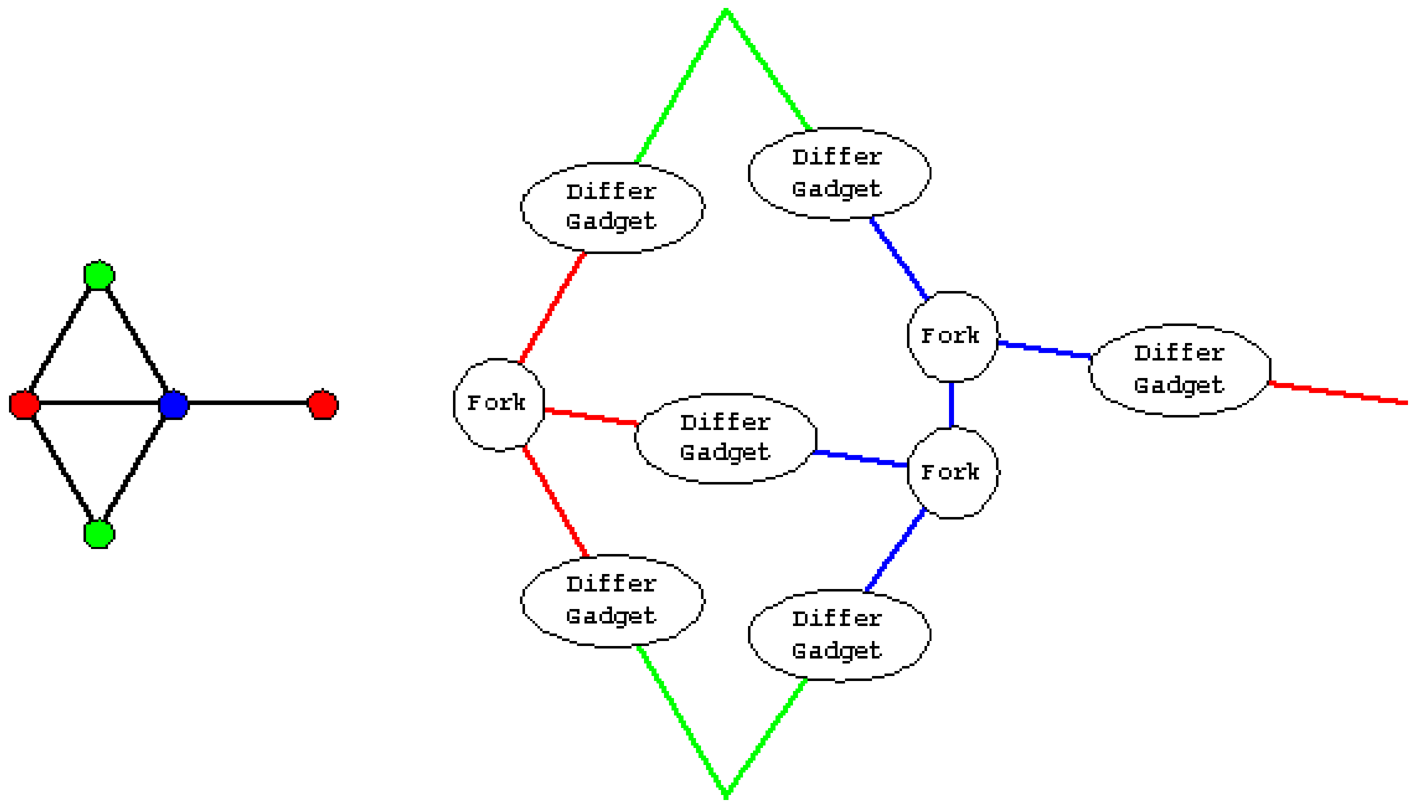

Figure 2. A graph (left) and its corresponding Corral blueprint (right)

Wires

Our wires will be rectangles of width 3 in the Corral puzzle, with every fourth row containing 2 in the left column and 12 in the middle column. A wire can be locally solved in essentially 3 different ways, as shown below in Figure 3. For reference, we alternately color some squares in the middle column with 3 colors: red, green, and blue, always in that order. We think of a wire carrying the signal of the color that is does NOT contain in its loop. Some additional grid squares can also be included in the loop, but the only possible squares in the middle column that are not in the loop are the colored squares.

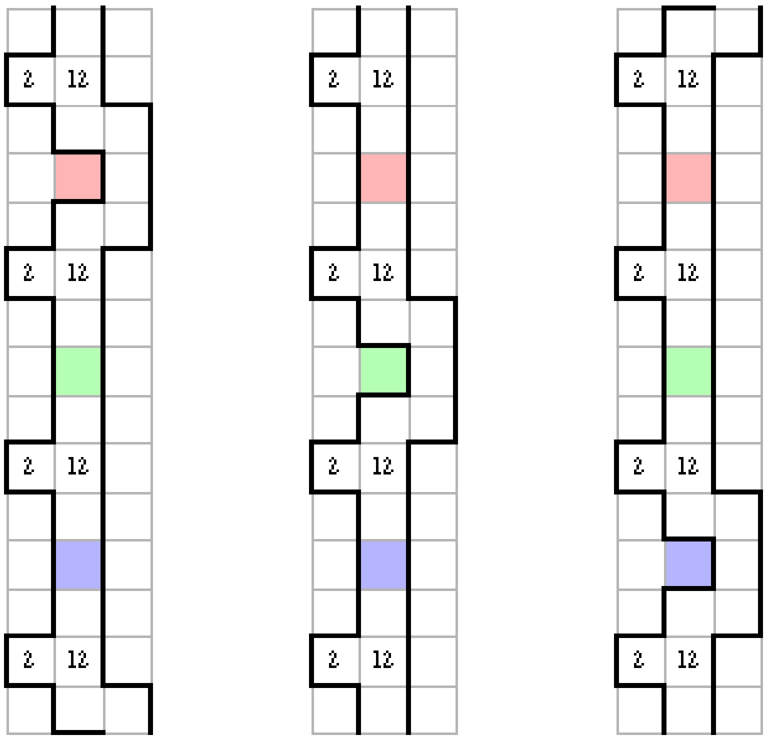

Figure 3. Wires carrying red, green, and blue signals

Notice that we can "cut" a wire by removing from the loop some squares from the third column. This breaks the local solution into 2 partial loops that would need to be connected elsewhere in the puzzle.

Since the squares containing the number 12 must be connected in triples, we can view our wires in a more schematic way, as in Figure 4. Each square in the schematic represents a 4x4 square centered around a 12 in the original wire. Lines of sight must connect triples of squares. The signal carried by a wire is the color not included in the line segments.

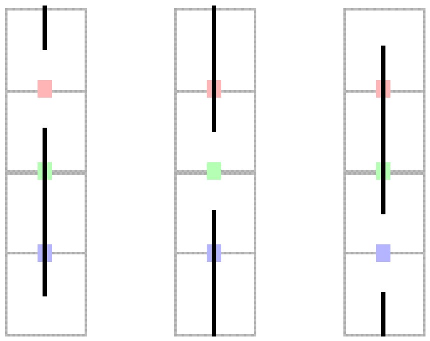

Figure 4. Schematics of wires carrying red, green, and blue signals

Turning, Delaying, and Containing Wires

Our turning gadget is shown in Figure 5. Notice that in each case, the signal coming in vertically must match the signal leaving horizontally. The number 2 in the middle prevents the gadget from turning a red signal into a green signal. Although the example shows a turn between the red and blue squares, we could turn between any pair of colored squares.

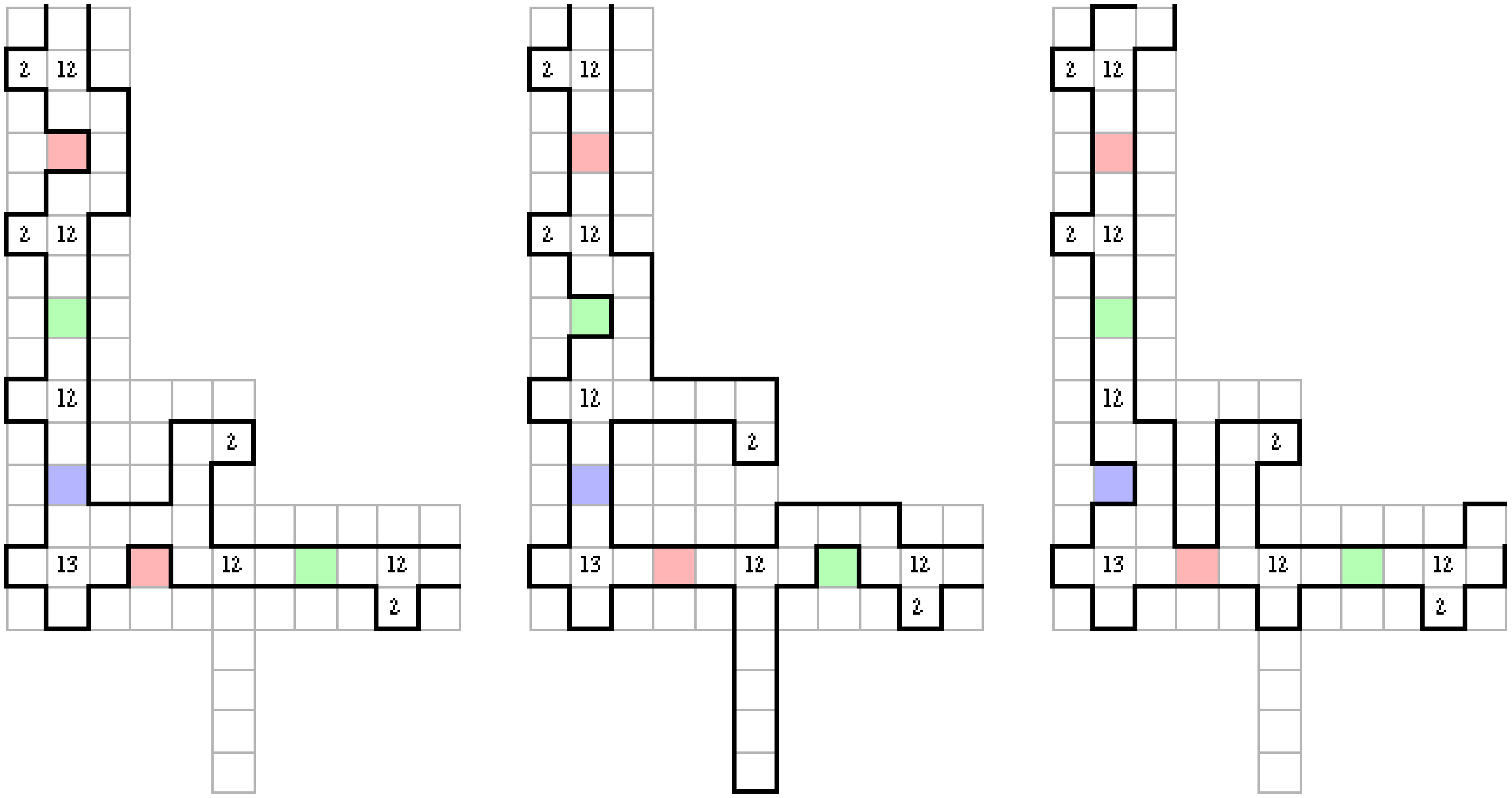

Figure 5. Gadgets turning wires carrying red, green, and blue signals

The schematic for our turning gadget is shown in Figure 6. Each square in the schematic is a 4x4 square centered around a 12 or 13 which must be visible from 2 other squares in the schematic. The rounded squares in the schematic do not contain a 12 or 13, and therefore are optional space. Due to the placement of the centermost 2, these rounded squares can only be seen from the flat side.

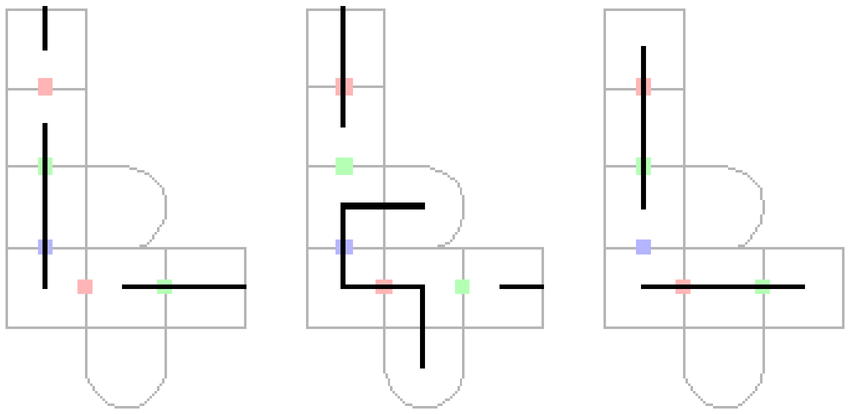

Figure 6. Schematics for gadgets turning wires carrying red, green, and blue signals

By turning a wire repeatedly, we can move a wire any multiple of 4 squares in any direction. We can also delay a wire so that any particular color square has a given color. For example, to delay a wire carrying a red signal to the right by 4 squares, we can turn the wire as shown in the schematic in Figure 7. By turning the wire up and then down , we have changed the phase of the colored squares.

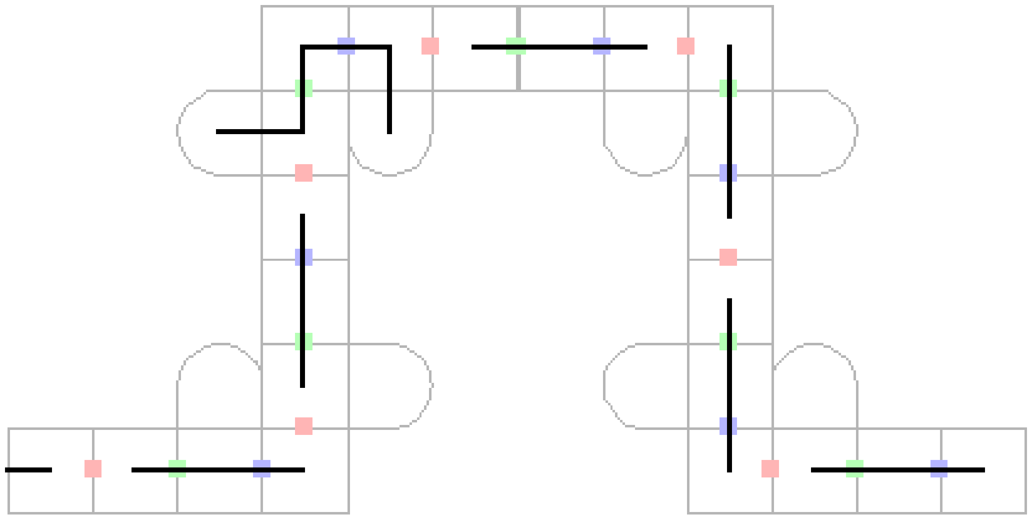

Figure 7. Schematic for delaying a wire carrying a red signal

To contain our wires, we build walls around them. These walls consist of tightly packed numbers which can only be solved in one way, and which prevent any wire or gadget from being connected to them. An example is shown in Figure 8. To connect the walls with the rest of the puzzle to form one continuous loop, we can leave off a few numbers in the wall to allow a connection.

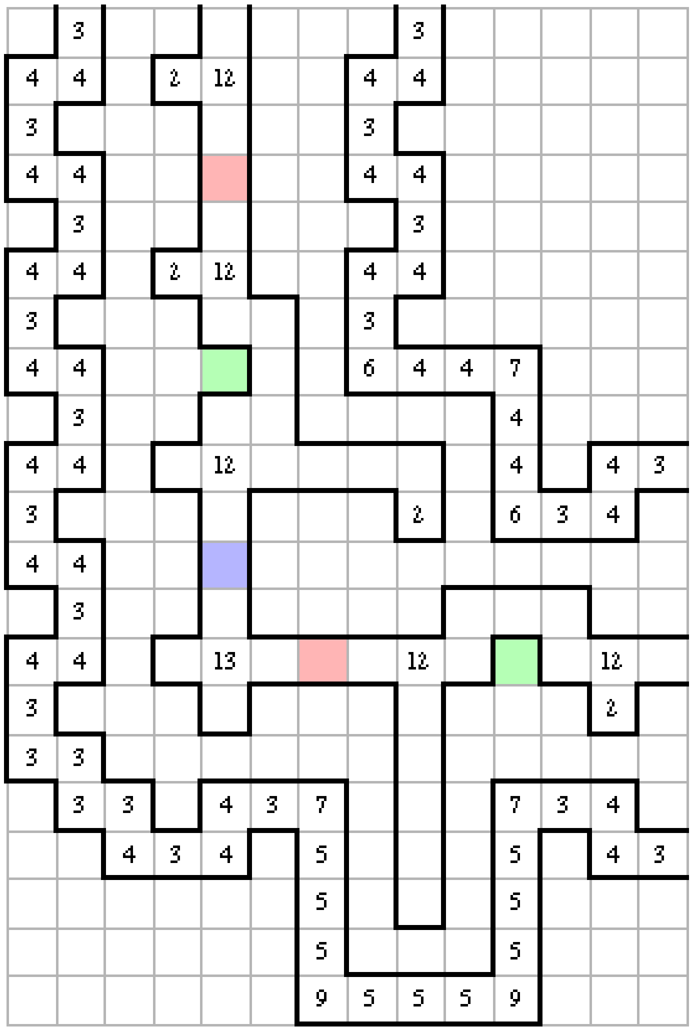

Figure 8. Turning gadget with walls

From these building blocks, it is clear that for any given Boolean circuit, we can construct a Cubic position that is solvable if and only if the circuit is satisfiable. The mapping from circuits to Cubic is bounded, so any polynomial time verification of the Satisfiability problem extends to a polynomial time verification for Cubic. Thus Cubic is NP-complete.

The Fork Gadget

Our fork gadget is shown in Figure 9, 10, and 11. Wires leave the gadget on the left, top, and right. The schematic of a fork gadget is shown in Figure 12.

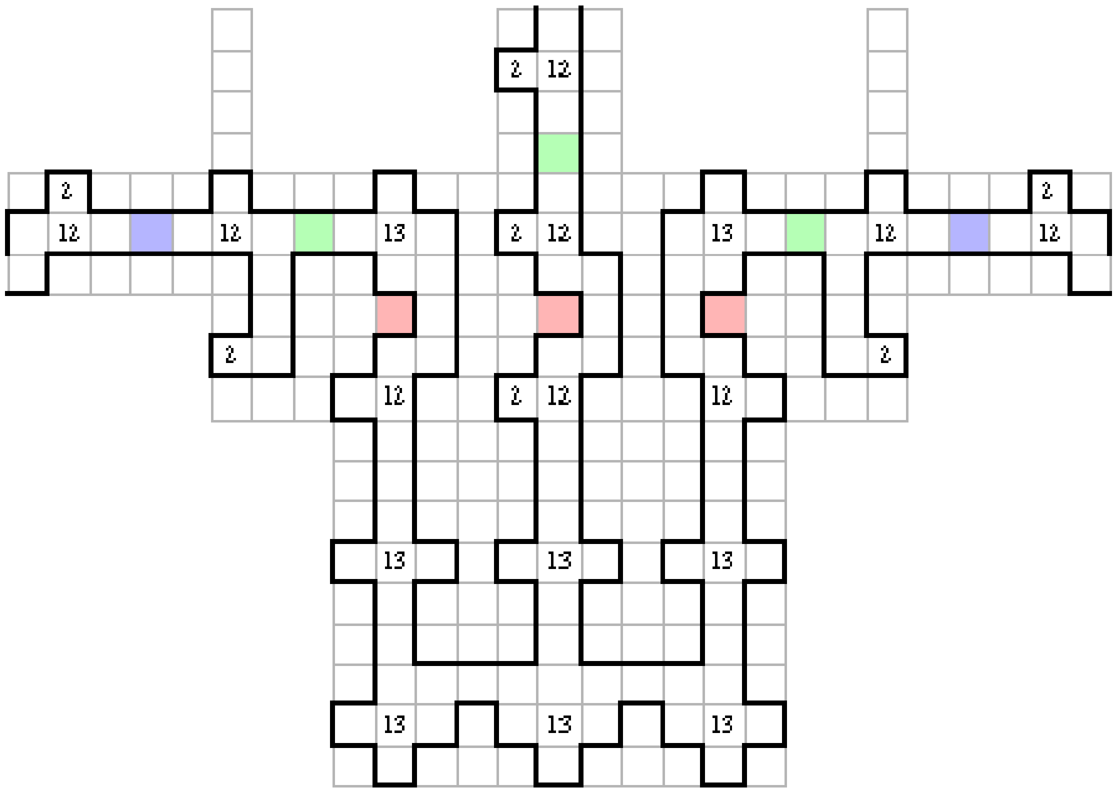

Figure 9. Fork gadget producing three red signals

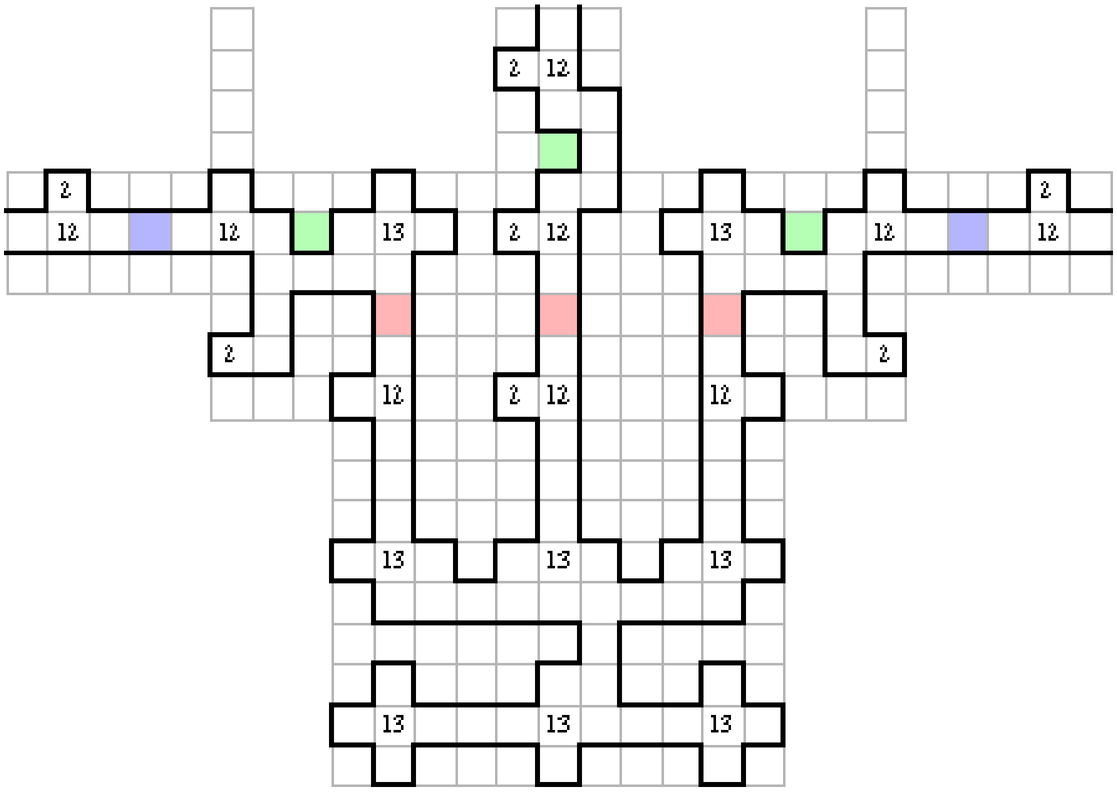

Figure 10. Fork gadget producing three green signals

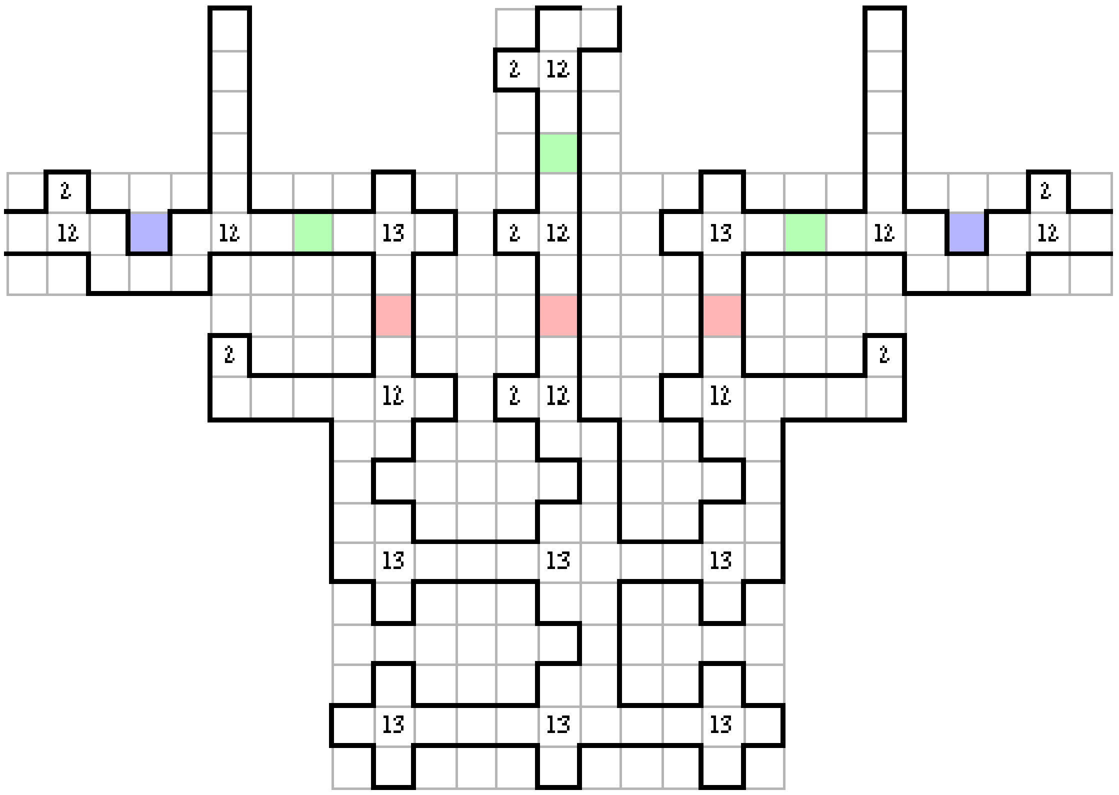

Figure 11. Fork gadget producing three blue signals

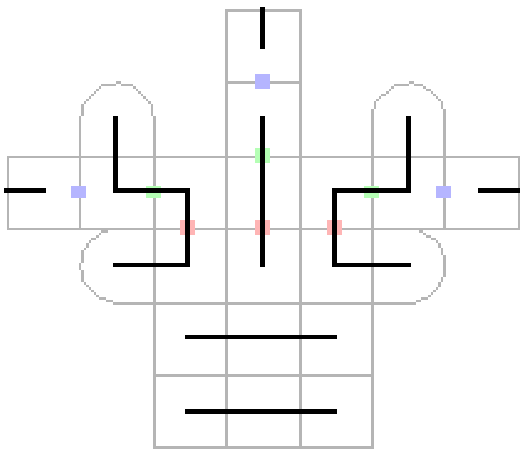

Figure 12. Schematic for a fork gadget producing three blue signals

The Differ Gadget

Our differ gadget is built from 3 differ modules, one associated with each color. A red differ module is a configuration that takes two input signals and has a local solution unless both the wires carry red signals. Wires from the top and right meet, and the 2 in the lower left hand corner prevents two red signals from meeting. Any other pair of signals can meet because the blank space added to the wires allows a green signal to change to a blue signal, and a blue signal can meet a red signal (see Figure 13) or another blue signal (see Figure 14). The schematics of these are shown in Figure 15.

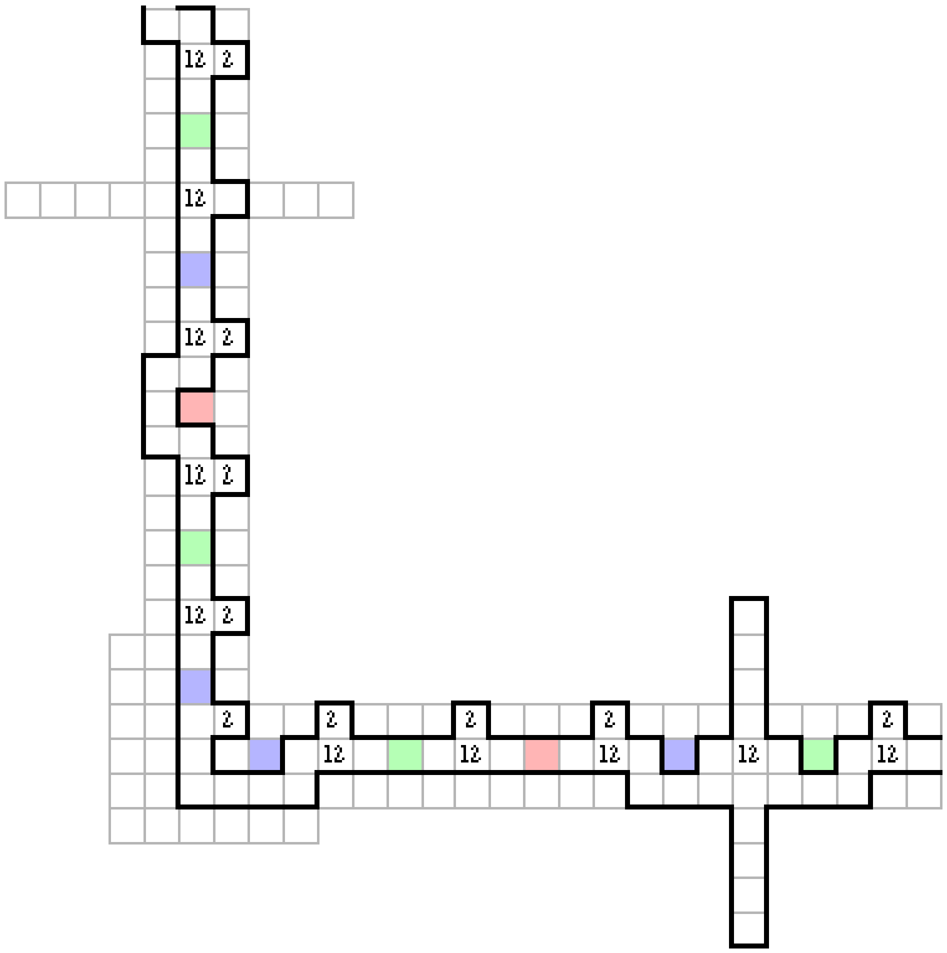

Figure 13. Red differ module with red and green (changed to blue) inputs

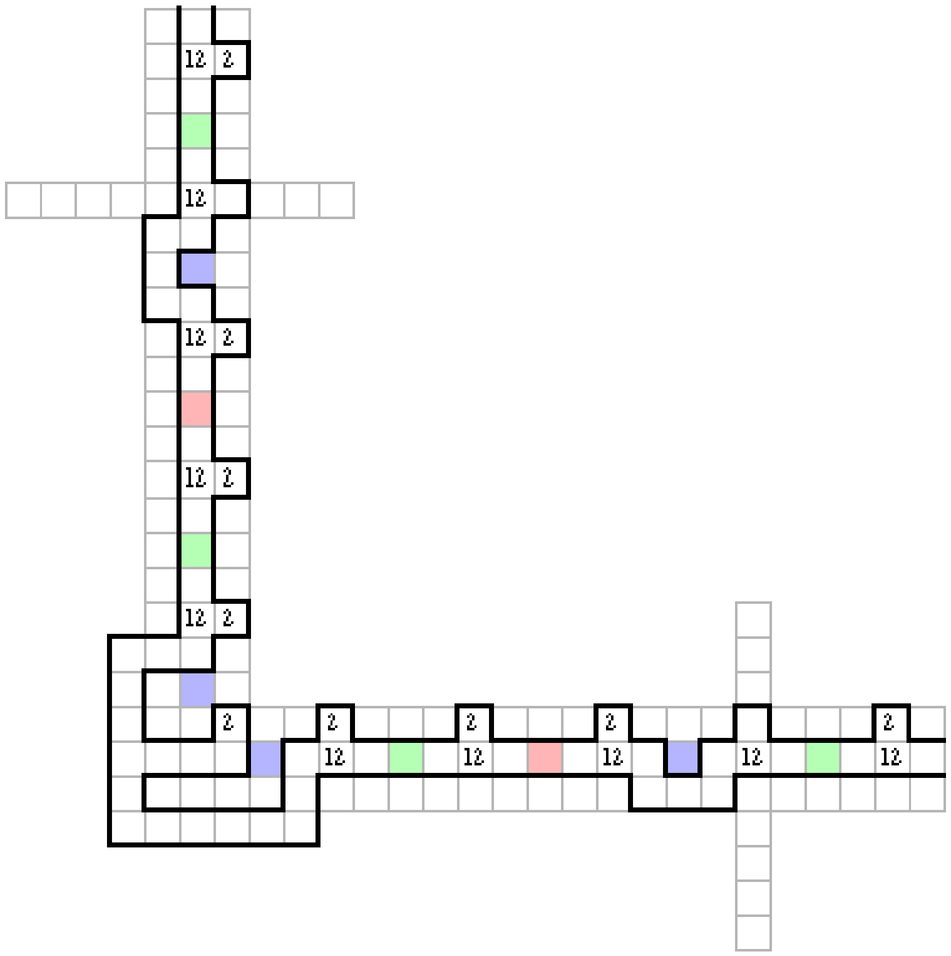

Figure 14. Red differ module with two blue inputs

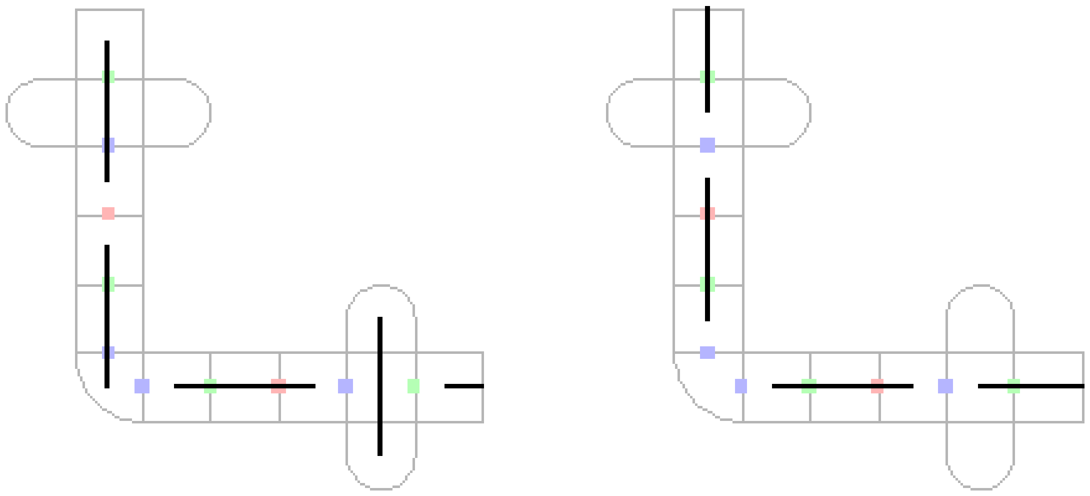

Figure 15. Schematics of Red differ modules

By permuting the colors, we can build green or blue differ modules as well. We then build a differ gadget according to the blueprint in Figure 16. Two wires that we want to carry different colored signals are sent through 2 fork gadgets each to split each signal into 3 identical signals. We can connect these 3 pairs of wires with a red differ module, a green differ module, and a blue differ module. Each module prevents the original pair of wires from both being a specific color. If there is no local solution in a given differ module, there will be no solution to the Corral puzzle, so there can only be a solution if the original pair of signals carry different signals. Recall that we can remove the "holes" in our gadget by breaking the wire where needed.

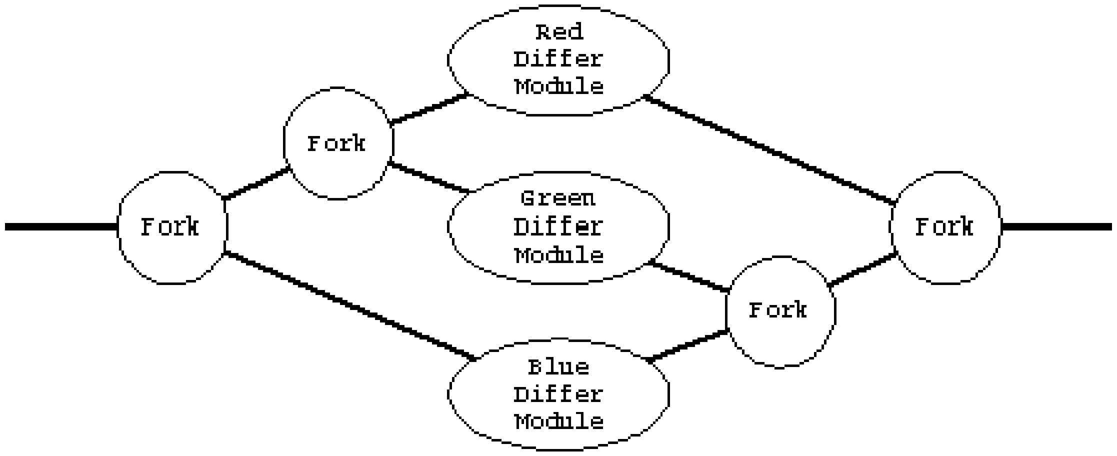

Figure 16. Blueprint for a Differ gadget

We have now completed the mapping of planar graphs to Corral puzzles. Since arbitrary planar graphs can be expressed as Corral puzzles, the problem of determining whether a Corral puzzle has a solution is at least as hard as that of 3-coloring planar graphs, so it is NP-hard.

Polynomial Time Mapping and Checking

We now show that the Corral puzzle we have built is polynomial in the size of the graph we started with. The addition of an edge requires 2 more fork gadgets, 1 more differ gadget, and an amount of wire and wall that grows polynomially in the size of the graph [3]. This means the mapping we have constructed from planar graphs to Corral puzzles is polynomial.

Assume that we are given a potential solution to a Corral puzzle as a polyomino. To verify that it is indeed a solution, for each numbered square we need to check the number of visible squares. This is clearly polynomial in the size of the polyomino. Thus, in addition to being NP-hard, the problem of determining whether a Corral puzzle has a solution is NP-complete.

References

[1] J. Culberson, "Sokoban is PSPACE complete." *Proc. Internet Conf. Fun with Algorithms* (1998), N. S. E. Lodi, L. Pagli, Ed., Carelton Scientific, 65-76.
[2] E. D. Demaine and M. Hoffman, "Pushing blocks is NP-complete for non-crossing solution paths". *Proc. 13th Canad. Conf. Comput. Geom.* (2001), 65-68.
[3] H. de Fraysseix, J. Pach, and R. Pach, and R. Pollack, "How to Draw a Planar Graph on a Grid". *Combinatorica*, **10** (1990), 41-51.
[4] E. Friedman, "Cubic is NP-complete". preprint.
[5] E. Friedman, "Spiral Galaxies Puzzles are NP-complete". preprint.
[6] M.R. Garey and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.
[7] R. Kaye, "Minesweeper is NP-complete." *Mathematical Intelligencer*, to appear.
[8] Nikoli, **91** (2000), 52.