Generating test cases:
I can choose the number of nodes on each side of the manifold that need to be wired up and the number of (forward WOLOG) edges needed to be wired up. I generate all 3x3 matrices possible with 1-9 edges, then I take those matrices, and I get the row and column sums. I store each unique pairing of row/column sums, and I count how many unique matrices made those column/sum pairs. This guarantees that each row/columns sum has at least 1 real solution, and also tells us how many matrices reduce to that row/column sum pair.

The wiring strategies:
I compared 3 strategies:
1. Random (pick a random empty cell in the matrix - put a 1 there).
2. Greedy (find the neediest row, then neediest column - put a 1 there).
3. Anit-greedy (find the least needy non-zero row, then least needy column - put a 1 there).

After each placement of a 1 in all three algorithms, there is a basic check if the rest of the row or column is fully determined, and it puts in the 1s or 0s in as needed when that happens.

Failure case: It is possible any of them to fail still, though, since a cascading effect of binding constraints could possibly lead to no solution no longer being possible.

The Analysis:
First, for each unique row/column sum pair, I run each algorithm on it 20 times to look for solutions. Since 2 and 3 are deterministic, a failure on the first try means it'll fail every other try - but I make it try anyway just to double check.

Performance Summary

Method	Successes	Failures	Success Rate
FILL_GUIDED_RANDOM	6532 / 6540	8 / 6540	99.9%
FILL_GUIDED_GREEDY	6540 / 6540	0 / 6540	100.0%
FILL_GUIDED_LEAST_GREEDY	2580 / 6540	3960 / 6540	39.4%

Uniqueness & Coverage Metrics

Method	Unique Solutions Found	Known Possible Solutions	Coverage	Problems w/ All Solutions	Problems w/ Multiple Solutions	Avg. Unique Solutions / Problem
FILL_GUIDED_RANDOM	510	511	99.8 %	326	98	1.56
FILL_GUIDED_GREEDY	327	511	64.0 %	229	0	1.00
FILL_GUIDED_LEAST_GREEDY	129	511	25.2 %	49	0	1.00

I also made some plots for just the first two: