Playing With Mazes

David B. Suits
Department of Philosophy
Rochester Institute of Technology
Rochester NY 14623

david.suits@rit.edu

4. More on Connection Matrices

A disadvantage of all the maze solving algorithms is that none will mark out a path from any cell to any other cell. That is, given the same set of doors and walls, but a different start or goal cell, the entire algorithm will have to be run anew.

In section 2.2, M^c was called the complete connection matrix, a maze's connection to the *i*th power, where $1 \leq i \leq n-1$ (where n is the number of cells in the maze; hence, $n-1$ is the longest possible path in the maze) such that all entries in M^i are non-zero. Not only does M^c tell us whether the maze is connected, but it also gives us a basis for understanding *how* it is connected. Specifically, what we are looking for is a matrix which gives us the minimum distance from any cell to any other cell. Let M^d be such a *minimum distance matrix*. It will display in matrix form the structure of walls and doors of the maze. Such information will, in turn, provide us with a means for calculating the path from any start cell to any goal cell. That is, from M^d we will be able to construct M^{sg}, the ***solution matrix*** for the goal cell g.

4.1 Constructing M^d, the Minimum Distance Matrix

Repeatedly running one of the maze solving algorithms (with suitable enhancements) may provide us with M^d. But given M^1, M^d will be simple to calculate as we are constructing M^c. Once an entry in some M^i becomes non-zero, we know how many steps (name, i) there are between the cell in the given column and the cell in the given row for that entry. Thus, i becomes the value for that entry in M^d. That is, for each of M^i, $i = 2 \ldots c$, all non-zero entries in M^i which were 0 in M^{i-1} are assigned the value of i in M^d (excluding M^d_{jj} entries — that is, the entries in the main diagonal — which are assigned 0 in M^d, since there are minimally 0 steps between any cell and itself). Figure 20 repeats the maze shown earlier in figure 15, and gives both M^c and M^d.

4.2 Constructing M^s, the Solution Matrix

Suppose, for the maze in figure 20a, that cell 1 is the goal cell. We wish to know how best to get to that cell from wherever we are. Suppose we are in cell 5. Consulting M^d we discover that cell 1 is two cells away. But in what direction? It is easy to find out. From cell 5 we can move only to cell 4 or to cell 6. (These are the entries in M^d in figure 20c indicating minimal distances of 1 from cell 5.) If we were to move to cell 4, then cell 1, according to M^d, would be one step away; if we were to move to cell 6, then cell 1 would be 3 steps away. Clearly, then, we get closer to cell 1 from cell 5 by moving to cell 4. So the process for calculating M^{sg} for any g in a maze with n cells is as follows:

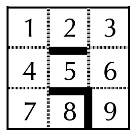
```
1. $M^{sg}_{ij} \leftarrow -1$ for all $i$, $j$ (–1 simply indicates an invalid entry).
2. For $i = 1 \ldots n$, $i \neq g$
       For each $k$, $k = 1 \ldots n$ such that $M^{d}_{ik} = 1$ and $M^{d}_{kg} < M^{d}_{ig}$
           $M^{s}_{ik} \leftarrow M^{d}_{ig}$.
3. $M^{s}_{gg} \leftarrow 0$.
```

Figure 21 shows solution matrices for g = 1, 2 and 3.

Notice that each solution matrix is equivalent to the results of a breadth-first search initiated from the given goal cell outwards to the rest of the maze. Since the structure of a maze is known once M^1 is known, it is then a question of implementation-dependent efficiency whether to engage in matrix multiplication or whether to simulate breadth-first searches in order to create any (or all) desired solution matrices.

Although the solution matrix is perhaps not very useful for the ordinary solve-a-maze puzzle, it might be useful in certain variations. Suppose, for example, that the rat is looking for the cheese placed somewhere in a familiar maze. And suppose the cheese has legs with which to move away from the rat. OK, if that's supposing too much, then we may instead imagine a computerized demon trying to catch a human player in a maze game. As long as the demon knows the identity to the cell its prey is presently in, it can consult the proper M^s and make the most efficient move towards it. In effect, the demon faces the problem of a continuous automated solution, where the start cell and goal cell are changing in the same maze. Either the specific M^s is calculated as needed, or else all of them are calculated, stored and consulted later on as required.

(a)



(b)

	1	2	3	4	5	6	7	8	9
1	56	47	35	63	40	33	35	15	10
2	47	51	47	40	32	40	17	5	16
3	35	47	56	33	40	62	10	2	30
4	63	40	33	91	64	45	61	30	17
5	40	32	40	64	61	63	35	15	30
6	33	40	62	45	63	86	18	5	46
7	35	17	10	61	35	18	51	31	5
8	15	5	2	30	15	5	31	21	1
9	10	16	30	17	30	46	5	1	26

(c)

	1	2	3	4	5	6	7	8	9
1	0	1	2	1	2	3	2	3	4
2	1	0	1	2	3	2	3	4	3
3	2	1	0	3	2	1	4	5	2
4	1	2	3	0	1	2	1	2	3
5	2	3	2	1	0	1	2	3	2
6	3	2	1	2	1	0	3	4	1
7	2	3	4	1	2	3	0	1	4
8	3	4	4	2	3	4	1	0	5
9	4	3	2	3	2	1	4	5	0

Figure 20. (a) A 3×3 maze. (b) M^c = M^5 for the maze. (c) M^d for the maze.

4.3 Cautions

But let's be clear about what has been claimed for these matrices. If you were put at the entrance (i.e., in a start cell) of an unfamiliar maze and told to go find the treasure and return, then there would be nothing for it but to use a suitable maze solving algorithm; after all, you do not yet know the structure of the maze; and not only do you not know how to get to the goal cell, you do not know its identity. However, once given the structure of the maze — after having examined it on your first run — you can construct the connection matrix, and hence M^c, M^d and M^s. At that point you can efficiently retrieve any further treasure in the maze merely by being told the identity of the cell it is in. Suppose, however, that although you have M^d, you know only that there is a treasure somewhere in the maze. Even though in your search for the treasure you can now be more efficient than a traveler who knows nothing about the structure of the maze, you will nevertheless have to employ some search procedure or other. That is, you know how to get efficiently from any cell to any other cell; but since you do not know the identity of the goal cell, you will have to search for it. That is, you face a traveling salesman's problem: how shall you organize your travels along alternative but known routes in order to cover all the territory while incurring the least cost? (Cost, in this case, is distance. The least cost will be incurred for the least backtracking.) It is a restricted traveling salesman's problem in that the start cell is stipulated. I have a gut feeling that some sort of matrix manipulation might straightforwardly solve this problem (would that make me famous?), but I have no suggestions to offer at this time. (Gut feelings are not very reliable anyway.)

(a)

	1	2	3	4	5	6	7	8	9
1	0	1	2	1	2	3	2	3	4
2	1	0	1	2	3	2	3	4	3
3	2	1	0	3	2	1	4	5	2
4	1	2	3	0	1	2	1	2	3
5	2	3	2	1	0	1	2	3	2
6	3	2	1	2	1	0	3	4	1
7	2	3	4	1	2	3	0	1	4
8	3	4	4	2	3	4	1	0	5
9	4	3	2	3	2	1	4	5	0

(b)

	1	2	3	4	5	6	7	8	9
1	0	1	2	1	2	3	2	3	4
2	1	0	1	2	3	2	3	4	3
3	2	1	0	3	2	1	4	5	2
4	1	2	3	0	1	2	1	2	3
5	2	3	2	1	0	1	2	3	2
6	3	2	1	2	1	0	3	4	1
7	2	3	4	1	2	3	0	1	4
8	3	4	4	2	3	4	1	0	5
9	4	3	2	3	2	1	4	5	0

(c)

	1	2	3	4	5	6	7	8	9
1	0	1	2	1	2	3	2	3	4
2	1	0	1	2	3	2	3	4	3
3	2	1	0	3	2	1	4	5	2
4	1	2	3	0	1	2	1	2	3
5	2	3	2	1	0	1	2	3	2
6	3	2	1	2	1	0	3	4	1
7	2	3	4	1	2	3	0	1	4
8	3	4	4	2	3	4	1	0	5
9	4	3	2	3	2	1	4	5	0

Figure 21. (a) M^{s1} for the maze in fig. 20. (b) M^{s2} (c) M^{s3}.