INTELLIGENCE REPORT

Hohyun Jeon

# Algorithms and Heuristics to be Implemented

The following are the intelligence methods to be implemented for the Numbrix puzzle. We will need to make the following assumptions that the starting table is a square grid, correct, and results in a unique solution. In order to find a solution the Numbrix Artificial Intelligence needs a way to quantify knowledge, a way to measure a solution, and a fast way to find it.

**Node Data Structure:** The intelligence will store and access data through a node data structure. A node is defined as a location on a grid. Each node’s location defined by its coordinate in the map. The coordinates start from 0 and end at length-1. For example, in a 2x2 table, the upper right hand node will have the coordinate of [1, 0]. The size of a map is defined as the lateral length of the table. So a 3x3 table is of size 3. An adjacency relation is defined between the nodes in the map for neighbor nodes in all four directions: North, East, South, and West. Each node will store its definite value, the value that is committed to the table, and a list of probable values, values that are heuristically determined but not yet committed. The intelligence will also keep a list of assigned numbers and remaining numbers. When the list of remaining numbers is empty, the solution is found.

**Numbrix Behavior Heuristics:** There needs to be some way to measure a potential solution from committing a number to a node in order to determine the correct solution. The heuristics can be organized into two categories: potential commit that is more likely and potential commit that is less likely due to current circumstances.

**Adjacent and Increment Rule Heuristic:** If the adjacent node value is known, then the current node is more likely to have an increment value. This heuristic will be extrapolated, for a broader vision, into a tree if a single depth does not result in a most probable solution. Thus the 3x3 table, given [0,0] is 1, below will have the following tree of depth 4.

**Dead Space Heuristic:** This heuristic is an extension to the puzzle rules. If the table is in the state below, 4 cannot be assigned to [1,2] because it will created a dead space on both sides on the next move. If such nodes exist it will be pruned and its ancestor’s probability will drop.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| **0** | 1 | 2 |  |
| **1** |  | 3 |  |
| **2** |  |  |  |

**Dead End Heuristic:** This heuristic is also an extension of the puzzle rules. The heuristic will define nodes that result in no children and is not the solution as dead ends. So, in the table above, if 4 is placed in [2,1] then 5 in [2,0] will result in a dead end.

**Echo Pruning Search Algorithm:** The heuristics above will help in generating the correct solution but can be extremely costly if the proper search algorithm is not used. The Adjacent and Increment rule tree has a branching factor of 1 to 3. To reduce time and memory, the algorithm will need to choose which possibilities to explore and what not to explore. Pruning will help increase the speed by not spending time on paths that will result in an incorrect or lesser solution. This algorithm will perform with the following instructions:

1. Given two numerically closest and committed nodes, two Adjacent/Increment trees will branch (one branching up numerically, and the other down like a bi-directional search) to each other.
2. Any choices that definitely result in either Dead Space or Dead End will be pruned. Probabilities will be increased or decreased depending on heuristics.
3. Eventually island (midpoint) nodes will be determined.
4. If only one island is possible, the path is “echoed” and committed. If there are more than one, then the process is repeated for other committed node pairs until the most probable solution is determined.
5. The algorithm ends when all numbers are committed.

# Algorithms and Heuristics to be Considered for Implementation

**Clumping Behavior Heuristic:** Because of the nature of this puzzle, the numbers behave in a way that the solution is one in which the path is clumped into an optimally compressed path. I say optimally because given the area to fill and the number to be incremented to, there is a mathematically and logically definable optimal solution that will clump increments to the destination node value. So given 1 to 9 in distance of 4, the solution path will be compressed from 9-1=8 to 4. Meaning the solution path needs to clump into a thickness of 2 or more before approaching the destination node. This has to be considered not only because it will make the AI more efficient but also because it will help in generating the correct solution. The downside is that the heuristic is as accurate as the defined equation.

**Hill Climbing Algorithm:** Node based algorithms will generate multiple solutions with local maxima’s. In order to determine the absolute maximum and prevent improper iteration exits, the proper hill climbing algorithm needs to be implemented. Because this is a solution optimization algorithm, this will be implemented last.