Thomas Fiorilla, Anthony Tiongson

Professor McMahon

CS440 Intro to AI

February 22, 2020

Assignment 1 Search and Genetic Algorithms

Environment

An n-by-n grid puzzle, where a chosen value n can be greater than or equal to 5 or a randomly generated n value of 5, 7, 9, or 11. n represents the number of row and column positions on the puzzle. The row and column indices are $(r_{min}, \ldots, r_{max})$ and $(c_{min}, \ldots, c_{max})$, so for a puzzle where $n = 5$, $r_{min} = c_{min} = 1$ and $r_{max} = c_{max} = 5$. The starting position of the puzzle is the upper left most position $r_{min} \times c_{min}$, or 1×1, and the goal position is the lower right most position $r_{max} \times c_{max}$, or $n \times n$. There exists an integer value representing a legal move for every non-goal position, with the goal position designated with the letter G.

There is one agent tasked to evaluate the puzzle by attempting solving it to derive a solvability score for it. The agent may then be tasked to manipulate the puzzle to increase its solvability score. The puzzle is static unless the agent is manipulating it, so the next state is determined by either the agent's position on the puzzle during an evaluation or in the resulting manipulation of it. Every randomly generated puzzle for the agent to evaluate and possibly manipulate is independent from the last.

Thus, this is a fully observable deterministic episodic static environment with discrete states for a single agent.

State Space

A generated puzzle is represented by the two-dimensional matrix $boardBuilt[i][j]$ where i represents a position in the row range $r_{min} - 1 = 0$ to $r_{max} - 1 = n - 1$ and j

represents a position in the column range $c_{min} - 1 = 0$ to $c_{max} - 1 = n - 1$. The values stored in the array $boardBuilt[i][j]$ each represent a legal move m at that particular i-th row and j-th position of the puzzle. In general, a legal move for a non-goal position ranges from 1 to the $max\{r_{max} - r, r - r_{min}, c_{max} - c, c - c_{min}\}$. The puzzle starting position is always at $boardBuilt[0][0]$ and its goal position is always at $boardBuilt[n - 1][n - 1]$; the goal position will always have a legal move value of 0 and a displayed value of G to represent the final desired destination.

An agent can then evaluate the randomly generated $n \times n$ puzzle $boardBuilt[i][j]$ starting at position $(0, 0)$. Generally, when arriving at position (i, j), the agent stores the minimum number of moves required to reach it $depth$ in $steps[i][j]$, takes note that its visited it in $visited[i][j]$, and finds out its move value m. The agent can then evaluate the puzzle using two different techniques.

The agent can make an exhaustive analysis using a breadth-first-search of the possible moves from its current position and put every subsequent possible position and their $depth + 1$ level as $(i_2, j_2, steps[i][j] + 1)$ into a queue q and (i_2, j_2) into the set v to remember not to evaluate that position again. The agent will evaluate every position in q in FIFO order. Once the q is empty, the agent will derive a solvability score and assign it to the puzzle's $value$.

The agent can alternatively take a targeted, more efficient analysis using an A^* search with an $estimate$, a $heuristic$, and a priority queue pQ. The $estimate$ can be considered the function $f(x) = g(x) + heuristic$, where $g(x)$ is the number of moves the agent has made $depth$ and the $heuristic$ is an estimated guess of the remaining minimum amount of moves the agent has left to reach goal. The $heuristic$ says that for the agent's current position (i, j) and its possible next position (i_2, j_2), if $i_2 = size - 1$ and $j_2 = size - 1$, the estimated amount of moves left to the goal is only 1. If $i_2 = size - 1$ or $j_2 = size - 1$, the estimated amount of moves left to the goal is at least 2. Otherwise, the estimated amount of moves to goal is at least 3. Using this $heuristic$, the agent

calculates an $estimate$ to be used with the next possible position and its $depth + 1$ level to put it in pQ as $(estimate, i_2, j_2, self.steps[i][j] + 1)$. A lower $estimate$ will give a position (i_2, j_2) a higher priority in the queue pQ, so the agent will evaluate positions estimated to be closer to goal ahead of those with higher estimates. Unlike the breadth-first-search, the A* search will end the agent's traversal once it arrives to the goal position, and the agent will derive a solvability score at this moment. Otherwise, the agent will derive a score when all possible moves are exhausted.

During either techniques, the agent will use the dictionary $prevPos[(i_2, j_2)] = (i\ ,\ j)$ to keep track of a possible path to the goal position. If the puzzle is solvable, the agent will populate a list $shortestPath$ to store and display the puzzle's solution.

HILL CLIMBING MUTATION

GENETIC ALGORITHM MUTATION

If $\mathbf{S}$ is the set of all states, then any state $\mathbf{s}$ where $\mathbf{s} \in \mathbf{S}$ is the combination of the array $boardBuilt[i][j]$, all the levels of the agent's evaluation of $boardBuilt[i][j]$, and all the levels of the agent's possible mutation of $boardBuilt[i][j]$.

Actions

If $\mathbf{A}$ is the set of all possible actions, then $\mathbf{a}$ where $\mathbf{a} \in \mathbf{A}$ is any action an agent can make to evaluate the puzzle, like traversing it, or to mutate the puzzle, like applying a hill-climbing algorithm to it.

Perception/Observations

The input the agent receives is the array $boardBuilt[i][j]$. If $\mathbf{O}$ is the set of all possible observations for a given puzzle, then $\mathbf{o}$ where $\mathbf{o} \in \mathbf{O}$ is any observable position of the agent when evaluating it or any observable change an agent can make to the puzzle's solvability score $value$ when manipulating it.

Transition Function

Any agent action $\mathbf{a}$ during any state $\mathbf{s}$ for a transition function τ gives a new state $\mathbf{s}'$, or $\mathbf{s}' = \tau(\mathbf{s}, \mathbf{a})$. The next state of depends on which legal move the agent takes. Once chosen, the current position of the agent and its current available legal moves are updated to reflect the new state.

Evaluation Metric

After an agent evaluates a puzzle, it will assign a solvability score to it. Upon completion of either search techniques, the agent uses $steps[size - 1][size - 1]$ or $visited[i][j]$ to derive a solvability score m or $-k$ for the puzzle's $value$, respectively. If the puzzle is solvable, the agent will set the $value$ of the puzzle to the minimum m number of moves it takes to solve the puzzle. A higher value m is defined as a more difficult puzzle. If the puzzle is unsolvable, then the agent will set the value of the puzzle as $-k$, where k is the number of positions not reachable from the start.