Open Lab 1

Solving Problems By Searching

CSCI 4350 - Introduction to Artificial Intelligence

Due: Sep. 30 @ 11:00pm

Overview

Develop a software agent in Python to solve the 8-puzzle problem using A* search.

Procedure

1. Create a Python program which uses random actions to generate random starting states for the 8-puzzle problem (`random-board.py`). *Note: the purpose of this program is to generate random puzzles that can be solved by your A* agent described below.*
    - The goal configuration for the 8-puzzle is defined as follows (zero is the "blank" square):
      ```
      0 1 2
      3 4 5
      6 7 8
      ```
    - A file containing this (goal) configuration can be downloaded from here: OLA1-input.txt.
    - Your random board generator should read the input configuration (the goal) from **standard input**, and also accept **two command-line arguments (integer: random number generator seed, integer: number of random moves to make)**, and should print a final board configuration to **standard output** in the same format as the input file format (see above).
2. Create a Python program which performs A* search for the 8-puzzle problem (`a-star.py`). *Note: The purpose of this program is to determine a solution (sequential set of board configurations leading back to the goal configuration) to one of the random puzzles generated by the program above.*
    - Your program should read an 8-puzzle board configuration from **standard input**, and take a **single command line argument (integer: heuristic to use)**:
        - `0` - h(n) = 0
        - `1` - h(n) = Number of tiles displaced from the goal
        - `2` - h(n) = Sum of Manhattan (city-block) distances of all tiles from the goal
        - `3` - h(n) = A novel heuristic of your own design
    - Each node should be given a *unique integer ID number*, starting with **zero** for the **root node**.
    - When sorting nodes in the frontier by f(n), **ties** should be broken by using the node ID number so that **newer** nodes will be preferred over **older** nodes.
    - Your program should output:
        - The total number of nodes visited/expanded (V)
        - The maximum number of nodes stored in memory (closed list + open list size) (N)
        - The depth of the optimal solution (d)
        - The -approximate- effective branching factor (b) where $N = b^d$
        - Each state along the optimal path from the starting state to the goal state
3. Utilize your programs to analyze the performance of the heuristics.
    - Use your random-board program to generate 100 unique starting states:
        - Use a unique seed for each board
        - Use the goal state as the starting configuration
        - Use exactly 100 random moves to generate each board
    - Run your a-star code on the 100 unique starting states using each heuristic (`0` , `1` , `2` , `3`).
    - Compile the following statistics for V, N, d, and b:
        - Minimum
        - Median
        - Mean
        - Maximum
        - Standard Deviation
4. Write a report (at least 2 pages, single spaced, 12 point font, 1 inch margins, no more than 4 pages) describing: *Note: The purpose of your report is to demonstrate how the different parameter choices impact the search process and to connect the concepts discussed in class to the lab assignment.*
    - the 8-puzzle problem,
    - the code you develpled to solve the problem,
    - the heuristics you implemented,
    - the experiments you performed and why,
    - the analysis methods used and associated tabulated statistics,
    - the performance of the code using the different algorithms/heuristics (**using the tables/statistics for justification**),
    - any limitations of the overall approach,
    - and any additional implementation details that improved the performance of your code.
5. Read the submission requirements in the syllabus before submitting your work for grading.

Requirements

- Use insightful comments in the code to illustrate what is happening on each step.
- Include a header in the source code and report with the relevant information for assignments as defined in the syllabus.
- Your `random-board.py` code should **only** print a shuffled board in **exactly** the following format:
  ```
  7 8 6
  3 4 5
  2 1 0
  ```
- Your A* code should **only** print the information as listed above, in the **exactly** the following format:
  ```
  V=379
  N=654
  d=14
  b=1.58896

  6 3 4
  1 0 2
  7 5 8

  6 3 4
  0 1 2
  7 5 8

  ...
  ```
- Write your report such that a peer NOT taking this course would understand the problem, your approach to solving it, justification of various choices (heuristic, newest node first, etc.), and your final comments.
- Include a table of **all** of the *statistics* (i.e. not the raw data) compiled for your report.
- Include at least one figure to illustrate the 8-puzzle problem.
- All sources must be properly cited; failure to do so may result in accusations of plagiarism.
- Your report should be submitted in PDF format.

Submission

- A zipped file (.zip) containing (with **exact** filenames):
    - `random-board.py`
    - `a-star.py`
    - `report.pdf`
- Typical command to zip your lab: `zip OLA1.zip a-star.py random-board.py report.pdf`
- Download your zip file and then use your PipelineMT credentials to log in and submit your zip file to the Open_Lab_1 dropbox: https://jupyterhub.cs.mtsu.edu/azuread/services/csci4350-assignments/