Experimental Design

* Identify the algorithms and problems selected for comparison:

**-A\***

**-Bidirectional A\***

**-Weighted Bidirectional A\***

**-Bidirectional Search**

**-Path Planning (maps generated using DFS)**

**- complex maps with very few solutions**

**-Path Planning (maps generated using Prim’s)**

**- less complex maps with many possible solutions**

**-Sliding Puzzle**

* List a minimum of 4 experiments involving each of two problems being solved with each of two algorithms. For example, your 4 experiments might be nQueens solved with GA and Backtracking DFS; and cross-math solved with GA and Backtracking.

**-Path planning comparison of standard A\*, Bidirectional A\*, and weighted Bidirectional A\* on maps/mazes generated from a DFS maze generator**

**-Same as above except on maps/mazes generated from a maze generator that utilizes Prim’s algorithm. The difference between the two being that DFS generates complex mazes with very few solutions while Prim’s algorithm generates less complex mazes with many solutions.**

**-Comparing standard A\* to Bidirectional A\* on sliding puzzle.**

* Describe the metrics collected for each experiment.

**All of my experiments use the same three metrics: Time, Space (nodes generated), Optimality**

* If relevant, describe how test cases are constructed or sourced.

**I use a puzzle/map generator to create a problem, then I have all algorithms that I’m comparing solve the problem. I record the time it takes, number of nodes they generate, and how optimal the solution is for each algorithm. I repeat this process to get a large sample size (10-100 samples) for each size/difficulty of problem, then I average the results for each size and algorithm to get my data.**

Preliminary Data Collection

* For each experiment, identify the scale at which the algorithm "breaks"

**-For path planning it’s extremely difficult to test when the algorithms will break given that it takes at least 10x longer to generate a good map/maze than it takes the algorithm to solve it and usually the algorithm that generates the maze will give out before the search algorithm. For mazes generated with Prim’s Algorithm, Bidirectional A\* can solve mazes that are around 10,000x10,000 in less than an hour. Maps/mazes of that size take hours to generate and often cause my computer to run out of memory or cause an error because of their size. A\* solved a similarly sized maze in three hours, slower than Bidirectional A\*. To “break” the algorithm’s I’m planning to instead have them solve multiple mazes simultaneously (within the same memory space), or create an algorithm that generates mazes that are more difficult but smaller.**

**-For Sliding Puzzles, A\* and Bidirectional A\* run out of memory after about an hour, and what causes them to “break” isn’t so much the size of the puzzle as how many moves were used to scramble it. A\* successfully solved a 4x4 puzzle (scrambled with 50 moves) and a 5x5 puzzle (scrambled with 40 moves) in less than an hour, however it ran out of memory on a 5x5 puzzle (scrambled with 45 moves). Bidirectional A\* does slightly better than standard A\* because it’s better able to solve a puzzle before running out of memory. Bidirectional A\* successfully solved a 5x5 (scrambled with 50 moves) but failed on a 5x5 (scrambled with 60 moves)**

* Produce a table or graph for each experiment with a minimum of 4 data points of

**Tables and graphs are included at the end of this document. They will be organized and explained in the paper**

* Provide an anticipated conclusion

**-A\* appears to be better in general for path planning. Though in certain circumstances Bidirectional A\* does better, such as in large maps with many solutions. Weighted Bidirectional A\*, as expected, performs similar or worse than A\* and Bidirectional A\* in most cases.**

**-Bidirectional A\* appears to be better for Sliding Puzzle; however, this may only be because the system the experiments were run on is memory constrained.**

**Assessment**

NO CREDIT will be given for preliminary data collection if minimum requirements are not met.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sliding Puzzle | Bidirectional A\* |  |  |  |  |
| Sample Size | Size | Scrambling | Avg Time | Avg Space | Optimality |
| 10 | 3 | 10 | 0.045 | 1649 |  |
| 10 | 3 | 20 | 0.415 | 31729 |  |
| 10 | 3 | 30 | 0.529 | 74325 |  |
| 10 | 3 | 40 | 1.08 | 162438 |  |
| 10 | 3 | 50 | 1.066 | 258147 |  |
| 10 | 4 | 10 | 0.134 | 341809 |  |
| 10 | 4 | 20 | 3.5613 | 433794 |  |
| 10 | 4 | 30 | 6.446 | 758729 |  |
| 10 | 4 | 40 | 42.862 | 2760670 |  |
| 10 | 4 | 50 | 38.137 | 5033117 |  |
| 10 | 5 | 10 | 0.234 | 571652 |  |
| 10 | 5 | 20 | 5.37 | 948654 |  |
| 10 | 5 | 30 | 19.45 | 1974563 |  |
| 10 | 5 | 40 |  |  |  |
| 10 | 5 | 50 |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sliding Puzzle | Standard A\* |  |  |  |  |
| Sample Size | Size | Scrambling | Avg Time | Avg Space | Optimality |
| 10 | 3 | 10 | 0.072 | 1879 |  |
| 10 | 3 | 20 | 0.524 | 27456 |  |
| 10 | 3 | 30 | 1.01 | 86475 |  |
| 10 | 3 | 40 | 1.42 | 184658 |  |
| 10 | 3 | 50 | 1.76 | 321486 |  |
| 10 | 4 | 10 | 0.105 | 489701 |  |
| 10 | 4 | 20 | 4.512 | 674852 |  |
| 10 | 4 | 30 | 7.14 | 975482 |  |
| 10 | 4 | 40 | 45.621 | 2941670 |  |
| 10 | 4 | 50 | 139.458 | 6473205 |  |
| 10 | 5 | 10 |  |  |  |
| 10 | 5 | 20 |  |  |  |
| 10 | 5 | 30 |  |  |  |
| 10 | 5 | 40 |  |  |  |
| 10 | 5 | 50 |  |  |  |

