Experimental report for the 2021 COM1005 Assignment: The Rambler's Problem*

Alex Street

May 16, 2021

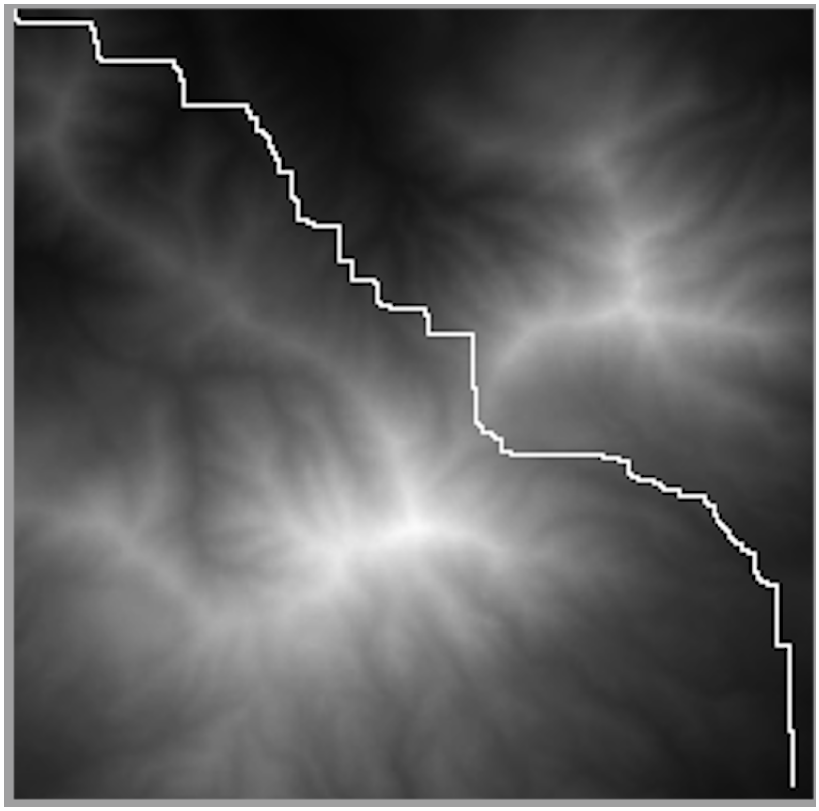

Figure 1: Diablo Solved

*https://github.com/streetalexander13/com1005-project

1 Description of my branch-and-bound implementation

Branch and bound is a search algorithm for finding the shortest possible route. The algorithm expands the node with the lowest local+global cost. This results in an algorithm with is admissible as it will always find the shortest route.

In the ramblers problem the route cost was one + the positive change in height. The data of the terrain used was stored in a pgm file. In this instance the Max Height was 255 and Minimum was zero, however when going down there was no additional cost. The algorithm is essentially blind to where the target is and will expand out from the start point in every direction regardless of whether it is closer to the target.

2 Description of my A* implementation

The A* algorithm is an improvement to branch and bound algorithm, it adds a heuristic. This is an estimation of the distance to the target. So now when choosing the next node the cost is calculated as (1 + height increase + distance to the target). The distance to the target could be calculated using Manhattan or euclidean distance. Manhattan distance is the x distance + the y distance. The Euclidean distance is if imagined as a triangle the hypotenuse $\sqrt{x^2 + y^2}$.

The heuristic could also consist of more than distance. It could incorporate difference in height, this would result in finding a potentially a longer but flatter route.

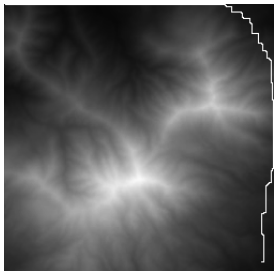

Figure 2: Solved but for flat route

3 Assessing efficiency

In order to test the efficiency of the algorithms i used the calculated efficacy value and tested the time taken to find the solution to a point. In order to increase speed, i commented out the printing of each iteration with the open list.

Algorithm	Time to completion [s]	efficacy
Branch and Bound	682	0.007090568
A* Manhattan Distance	94	0.01359333
A* Euclid's distance	193	0.009946031
A* With Manhattan Distance + Height difference	263	0.015408461

Table 1: Time to point 240 240 in diablo.pgm from 0 0

Algorithm	Time to completion [s]	efficacy
Branch and Bound	441	0.0045361402
A* Manhattan Distance	35	0.021419095
A* Euclid's distance	134	0.0090354225
- A* With Manhattan Distance + Height difference	80	0.015660152

Table 2: Time to point 240 240 in diablo.pgm from 85 57

Algorithm	Time to completion [s]	efficacy
Branch and Bound	460	0.0045484975
A* Manhattan Distance	49	0.012253853
A* Euclid's distance	74	0.009967637
- A* With Manhattan Distance + Height difference	78	0.0126361

Table 3: Time to point 130 240 in diablo.pgm from 85 57

Algorithm	Time to completion [s]	efficacy
Branch and Bound	148	0.0056527774
A* Manhattan Distance	1	0.062852845
A* Euclid's distance	7	0.022046205
- A* With Manhattan Distance + Height difference	16	0.039086375

Table 4: Time to point 87 34 in diablo.pgm from 200 87

As you can see above the most efficient algorithm is the A* using the Manhattan Distance. I did test this with the addition of the height metric. This however is not the quickest as it is finding a longer but flatter route.

3.1 Assessing the efficiency of my branch-and-bound search algorithm

The branch and bound algorithm was very adequate in finding routes of small pgm files such as the tmc file. However it took over 5 minutes in processing to find the shortest route across the diablo file. It also score lowest in efficiency.

3.2 Assessing the efficiency of my A* search algorithm

The A* algorithm had a much shortened processing time to the branch and bound when calculating the best route crossing diablo. It took only 1 minute 30 seconds. Th efficacy was also twice that of branch and bound and time to compute was up to 6 times faster on some runs.

3.3 Comparing the two search strategies

The A* is clearly the superior algorithm when the goal state distance/direction is known. This becomes increasingly important as file size increases A* have the same exponential growth like branch and bound. It was also important the heuristic used in the A* search and i found the best results when using the Manhattan distance.

4 Conclusions

The suitable algorithm becomes relevant as file size increases, for any large file it would be recommended to use the A* algorithm with the Manhattan distance. This will reduce execution time and also has more options such as minimising height change. However this is only possible when the distance from the start to goal node is known.