August 13, 2022

Frontier Navigation

Skamania II Hiker's GPS

Group #1

Tobin Joseph, Nathan Diaz, Nava Babaei

Off-trail Pathfinding

Requirements

The Skamania II GPS's off-trail pathfinding feature needs to find the most hiker-friendly trail for an arbitrary terrain. The terrain is represented as a 3D surface discretized into uniformly spaced tiles. The hiker can move with chessboard motion across the XY plane of this surface with the "cost" per step being defined by the change in height for each step:

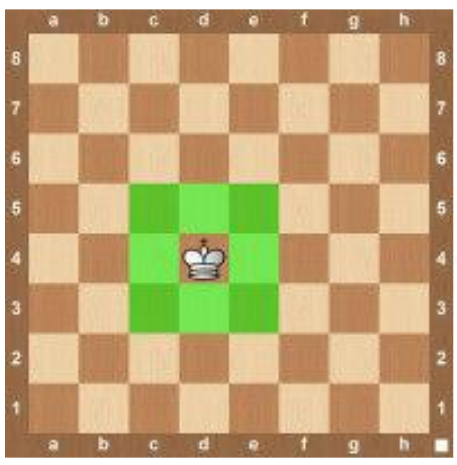
$$cost(h_0, h_1) = e^{h1 - h0} = e^{\Delta h}$$

Where h_0 is the height of the current title and h_1 is the height of the tile to be moved to. The total path cost is defined as the sum of all the individual step costs. We are hoping to implement the A* algorithm for this task, but our team lacks the expertise to code this and design admissible heuristics to ensure that the optimal path is found.

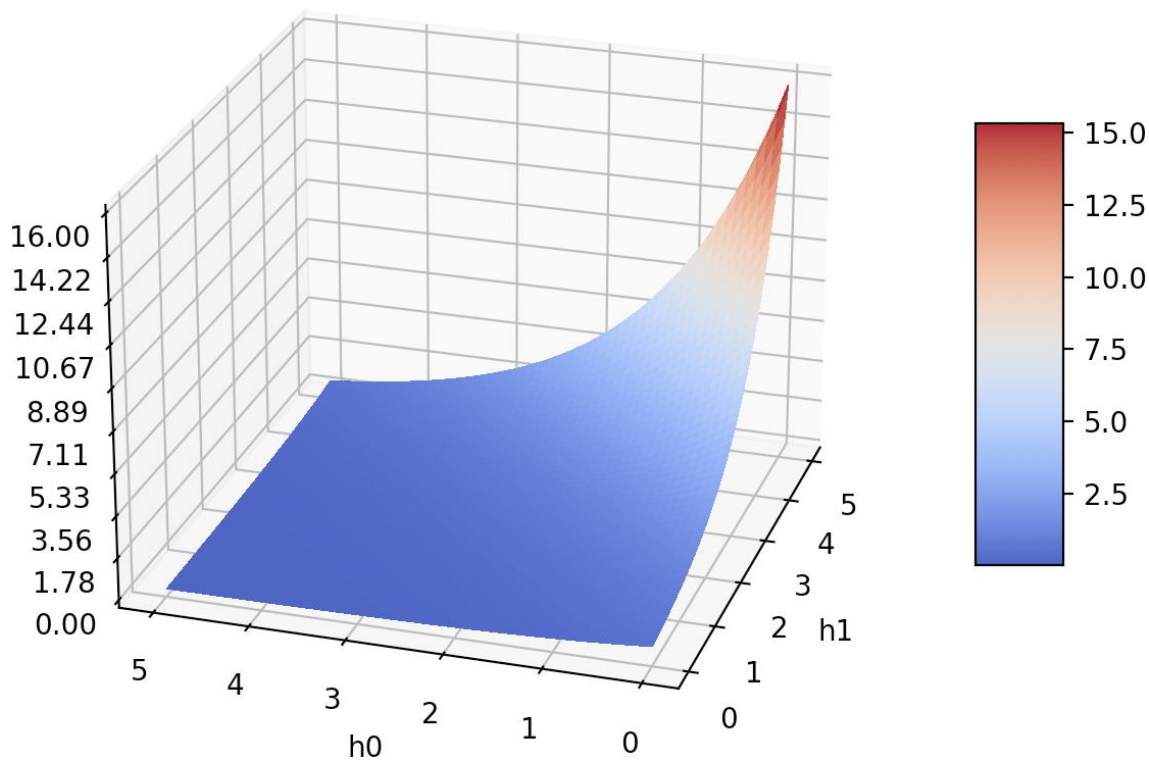
Designing Admissible Heuristics

Assuming the cost function **for chess movement** (all eight neighbors), create an admissible heuristic, define it as a mathematical equation, and **prove/show** it is admissible.

- Tiles with heights [0-255]
- Navigate from start to goal



Chebyshev (chessboard) motion.



The cost surface can be visualized with this image.

Design Process

Heuristic for A* Algorithm

Best guess about remaining cost

In the A* Algorithm using the Frontier Methodology, we use, as in Dijkstra's Algorithm, a priority queue as the frontier. But in the A* algorithm the priority queue is sorted by:

- f(n) = g(n) + h(n) where
  - g(n): objective function (same as Dijkstra's algorithm)
  - h(n): heuristic

This heuristic must be the best guess about the remaining path cost, and it cannot overestimate the remaining path cost. This last requirement will make a heuristic **admissible**, and in so being, A* is complete and optimal. We will go through a process of designing a heuristic and prove/show that it is admissible.

Cost Function Analysis

First, we need to understand the cost function **g(n)** and how it behaves to be able to create an admissible heuristic, as we need to make an educated guess on the remaining cost. The map we are to traverse is divided into discreet tiles that have been assigned a height in the discrete range of [0-255]. We have to move from the start point to the endpoint one tile at a time and the cost of moving from one tile to the other is given by the constant e to the power of the delta in the height, from the current point / tile to the next. This function can provide costs in the range of e^{-255} to e^{255} and those are the maximum deltas from the possible values of the height when going down in height or going up in height. We must note the special case when the height of the current tile and the next tile are the same, which causes a delta of 0 and a cost of 1.

Chebyshev Distance

Minimum distance in tiles

Analysis

Since we use a Chebyshev (chessboard) movement, we might want to consider the shortest path from the current tile to the goal..

Page 2

Minimum Height Cost Chebyshev Distance

Straight line in height too

Analysis

Since we use a Chebyshev (chessboard) movement, we might want to consider the shortest path from the current tile to the goal also in heigth.

Page 2

Follow the Goal

Try to correct the height path

Analysis

Since we have a third dimension (height), we can use a delta in each tile that gets us closer to the goal height instead of looking in the XY plane.

Page 2

Analysis

Minimum Height Cost Chebyshev Distance

Straight line in height too

Using one of the approaches to find a heuristic, the one to try and simplify the problem, we will analyze the possibility of using a Chebyshev distance on the XY plane of the map with a straight line on the height across that line. The Chebyshev distance is given by the expression:

$$d = \max(|x_g - x_0|, |y_g - y_0|)$$

For the cost to be equal to this, it would require a map with a unform height, with no changes in it. That is not the real case.

Going in a straight line on the height will require that the change in height be equal among all the tiles on the Chebyshev path. Thus, the cost of each tile will be given by the slope of the line like this

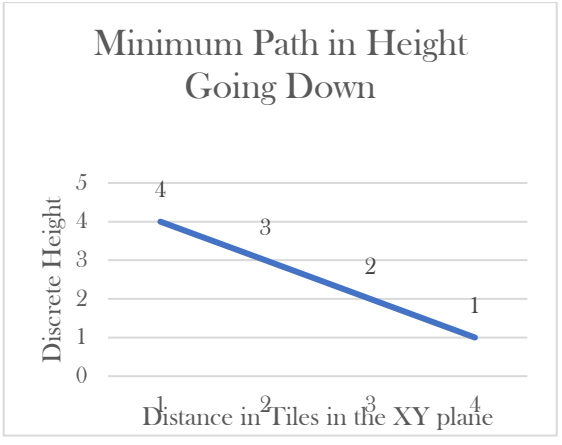
$$Cost(n) = e^{(hg - hn) / d}$$

where hg is the height at the goal and hn is the height at the tile being evaluated.

The total cost would then be given by,

$$Total\ Cost = d\ e^{(hg - hn) / d}$$

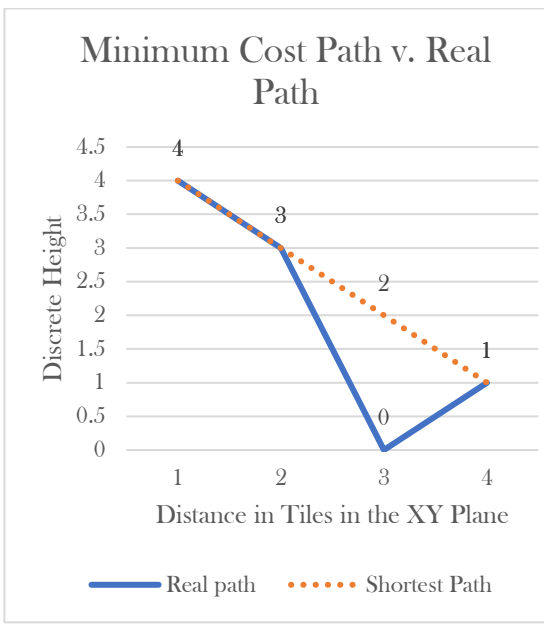
As we see from the Chebyshev Distance Analysis below, it is not an admissible heuristic. Thus, this predicted cost expression that uses it as a factor will also be not admissible, and it will try to steer the results to a straight line. But we can use a technique to reduce the value of this cost equation to the level that becomes an admissible heuristic as the remaining cost it predicts will not be greater than the actual remaining cost. We can use a weight less than one multiplied by this expression to get an admissible expression that will steer slightly the path into a straight line causing it to traverse less tiles to the goal.

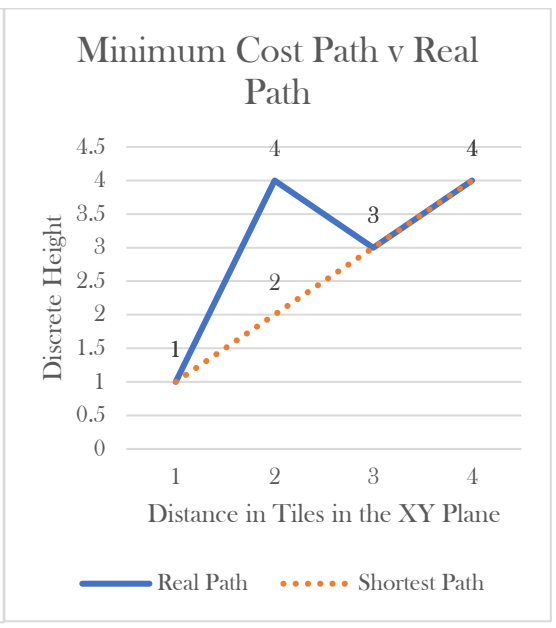


Minimum cost path in height is a straight line in height across the Chebyshev straight line distance.

Dividing d by powers of 2 we got to the following expression for an admissible heuristic:

$$h(n) = 0.000125\ d\ e^{(hg - hn) / d}$$





Examples of a shortest path in height vs the real path.

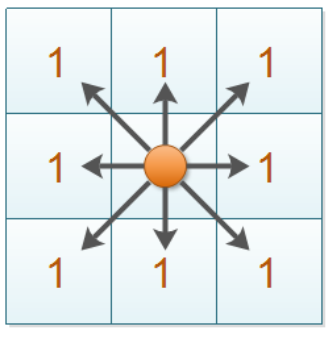
Analysis

Chebyshev Distance

Minimum distance in tiles

Since we have a Chebyshev (chessboard) movement, we might want to consider the Chebyshev distance as a heuristic. As the Manhattan distance the result of this path will have a uniform cost of one (1) per each tile traveled along this path. But here you can travel diagonally with a cost also of one (1).

We must verify that the cost prediction of this Chebyshev expression would be less than or equal to the actual remaining cost for all cases. We have that this expression will assume a cost of one per tile and lacking any obstacles on the map, this expression will get us a straight line going from the origin into the goal. The cost is based on height, and the minimum cost might not lay in a straight line to the goal, so this is not admissible as a heuristic.



$$\max(|x_1 - x_2|, |y_1 - y_2|)$$

Chebyshev distance example

Analysis

Follow the Goal

Try to correct the height path

The cost equation for this problem is set as

$$cost(h_0, h_1) = e^{h1 - h0} = e^{\Delta h}$$

The Dijkstra's Algorithm function is given by:

$$g(n) = cost(n-1) + cost(n, n-1)$$

The priority queue used in Dijkstra's Algorithm select the element with the lowest value in this function as the next to be evaluated. Given that the Δh can be negative or positive, the greatest change in height going down will cause the lowest cost, with the lowest cost possible being e^{-255}. From this we can see that Dijkstra's Algorithm prefers to search first on the deepest places of the map.

Now, knowing this, we need to verify if this is our best option. As we can see on the example paths comparison of a path going down, if the path goes in a straight line and we take a deeper path than necessary, it ends up costing more as the ascending to get again to the desired height will cost much more as the Δh is positive (continued on Page 3).

(Continued from Page 2)

Follow the Goal

A smarter path in height

A smarter approach is to try to get closer to the height goal on each tile we go. Also, smaller steps in height will cost less that big changes in height when going up as can be seen in the example paths going up. That, the priority queue takes care of. But not on a path going down, as the algorithm prefers the big drops going down. We can get a sense of direction in height by comparing the height of the candidate tile to the height of the goal. If the height of the candidate tile is higher than the height of goal tile, we need to go down. If the height of the candidate tile is lower than the height of the goal tile, we need to go up. We do not want to go too far from the goal height when we move. We would have various cases:

- We need to go up to reach the goal, but the candidate tile goes down.
- We need to go down to reach the goal, but the candidate tile goes up.
- We need to go down to reach the goal, but the candidate tile goes too way down, lower than the goal.
- We need to go up to reach the goal, but the candidate tile overshoots and goes higher than the goal.
- We are at the same height as the goal.

Based on these cases, we have a basis to build a multi part heuristic. We will need to analyze and prove/show that each of its parts gives an estimate of the remaining path cost that is less than or equal to the actual remaining path cost. We will use a strategy to try and get a straight line in height. We will add "corrections" to the Dijkstra's Algorithm to try and change its behavior from trying to go into the deepest path to the shortest path possible in height (a straight line). Hopefully this will make the algorithm visit less nodes to obtain the optimal solution.

We need to go up to reach the goal, but the candidate tile goes down.

In this case, we will add a "correction" that will consider some of the cost of coming back to the height before we take the candidate tile. This would involve defining the heuristic for this case as

$$h(n) = e^{+1}$$

To prove if this is an admissible heuristic we need to find if this value is less than the actual cost for all the cases this applies to. The intuitive proof is that if you increase the distance in height between the goal and the candidate tile, you will have to decrease that distance to get to the goal. So, if we add just one unit of change in height required to come back to the starting height of the parent, this is the lowest cost possible for a height change and this is always less than the actual remaining cost as we still must get to the goal height. Doing this will increase the priority assigned to the candidate tile, discouraging the algorithm from evaluating it in the short term.

We need to go down to reach the goal, but the candidate tile goes up.

In this case, we will add a "correction" that will consider some of the cost of coming back to the height before we take the candidate tile. This would involve defining the heuristic for this case as

$$h(n) = e^{-\Delta h}$$

The greatest jump down in height, provides the less cost. The intuitive proof is that if you increase the distance in height between the goal and the candidate tile, you will have to decrease that distance to get to the goal. This is the lowest cost possible for a height change going down in this range and thus is always less than the actual remaining cost as we still must get to the goal height. Doing this will increase the priority assigned to the candidate tile, discouraging the algorithm from evaluating it in the short term.

We need to go down to reach the goal, but the candidate tile goes too way down, lower than the goal.

If we undershoot going down, we need to come up to reach the goal height. So, we will add just one unit cost going up to the candidate tile

$$h(n) = e^{1}$$

An intuitive prove for this is that if go way down of the required goal, we will need to at least come back to the goal height at some point. By using the cost of the minimum height change of one, as it is a discrete value, we guarantee that we do not go over the actual remaining cost. Doing this will increase the priority assigned to the candidate tile, discouraging the algorithm from evaluating it in the short term.

We need to go up to reach the goal, but the candidate tile overshoots and goes higher than the goal.

If we overshoot going up, we must come down back to the goal height. So, we will add just one unit cost going down to the candidate tile

$$h(n) = e^{(hg-h1)}$$

An intuitive prove for this is that if go way out of the required goal, you need to come back down to the goal at some point. This is the lowest cost possible for a height change going down in this range and thus is always less than the actual remaining cost as we still must get to the goal height. Doing this will increase the priority assigned to the candidate tile, discouraging the algorithm from evaluating it in the short term.

We are at the same height as the goal.

For this case, we will make no adjustments to the Dijkstra's algorithm as we are at the same height of the goal.

$$h(n) = 0$$

With a heuristic of zero, there is no way we can have an estimate of the work remaining greater than the actual work.

Also, this will apply if:

We need to go up to reach the goal, the candidate tile goes up but does not overshoot.

We need to go down to reach the goal, the candidate tile goes down but does not undershoot.

After experimenting with this alternative, we found out that there must be some error in our analysis as some cases we cannot get the optimal path with this. In other cases, it performs spectacularly, but it does not work in all cases, thus it is not an admissible heuristic.

(Extra Heuristic)

Minimum Height Cost Variant

Variate the value of the heuristic with an exponential expression

Reading through the extra papers provided for the project we found a technique that weights a heuristic, giving it less weight at the beginning of the path and more weight to the heuristic at the end of the path. We decided to use an exponential expression that will create a factor between zero and one that will grow until it reaches one on the tile next to the goal. This factor will be defined as the following:

Scaling Factor = $e^{1 / d} / e$

This way the factor will grow as d diminishes as we approach the goal until the maximum value is 1 when d = 1.

Having this Scaling factor, we can increase the weight we had used for the Minimum Height Cost Chebyshev Distance heuristic we developed to increase further its effect as we get closer to the goal. The exponential factor will lower its effect at the beginning of the search, bringing it lower than the original equation at that point, even with the increased weight.

We increased this factor four-fold to 0.0005 to get to the following expression:

$$h(n) = (e^{1 / d} / e)\ 0.0005\ d\ e^{(hg - hn) / d}$$