FINDING THE OPTIMAL PATH THOUGH A WEIGHTED GRID

Paul Bouthellier
Department of Computer Science and Mathematics
University of Pittsburgh-Greensburg
Greensburg, PA 15601
pbouthe@pitt.edu

In this paper we consider the problem of finding a path of fastest descent through a weighted grid. This problem has applications where a person/particle/army must move between two points on a map in the shortest amount of time. Different terrains and/or obstacles can cause objects to have to move at different speeds at different points. As the general problem is difficult to solve analytically, we shall derive a technique to come up with a suboptimal solution, then show how this solution can be improved on.
A Java program can be used to list possible paths and GeoGebra can be used to study possible paths on the fly using nodes.

The most general version of this problem is given as follows:

Goal:

- Move from start to end positions.
- On each square one can move at a speed given by sij

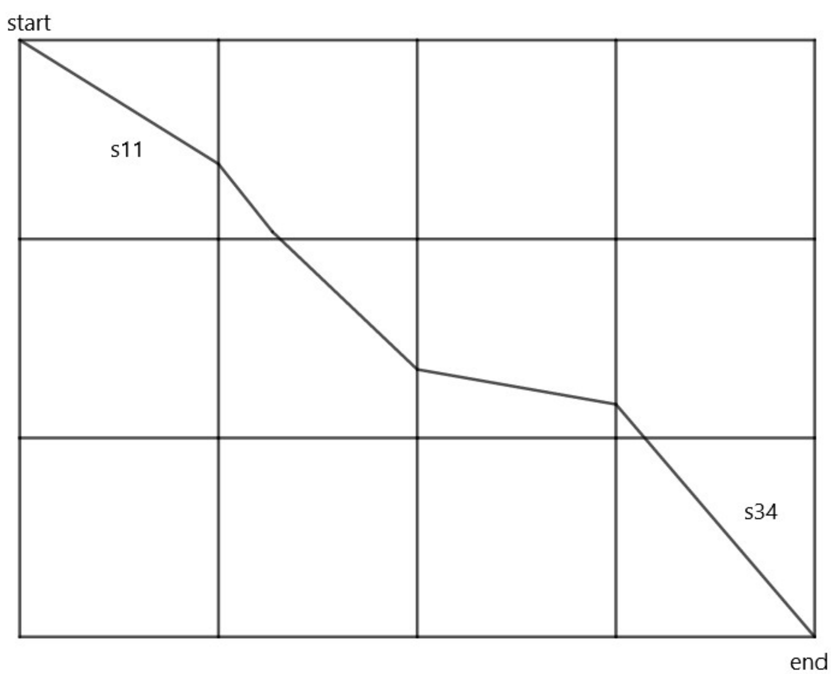

Figure 1

This problem can be viewed as a generalization of THE BRACHISTOCHRONE PROBLEM by Bernoulli (1696) [2]: Given two points A and B, find the path along which an object would slide (without friction) in the shortest time from A to B if it starts at A in rest and is only accelerated by gravity (or general velocities over subintervals).

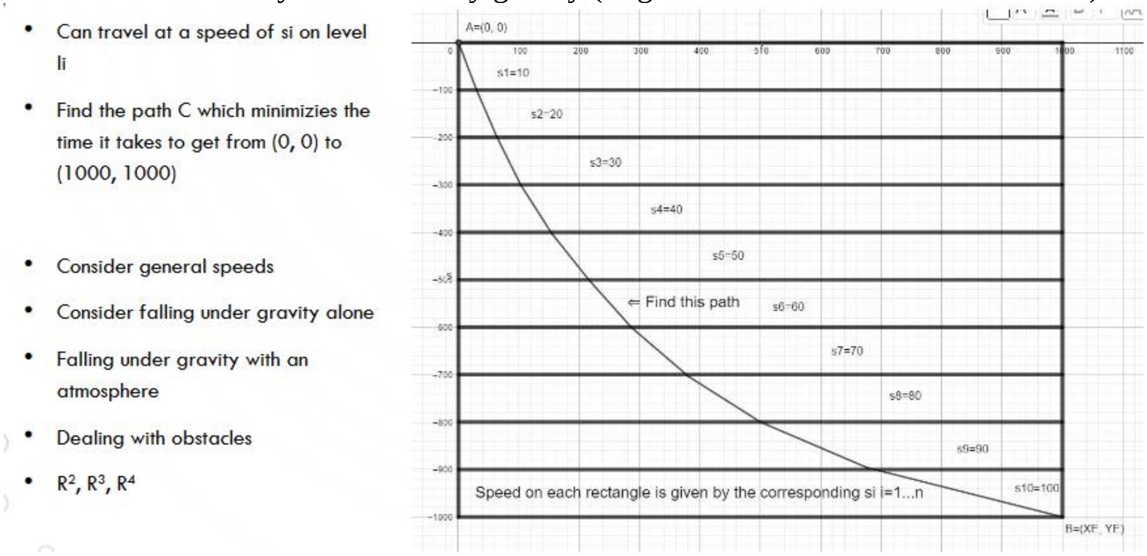

Figure 2

To study the grid problem in the 2x2 case is a straightforward problem in calculus and numerical analysis as there are only two variables to deal with [1]

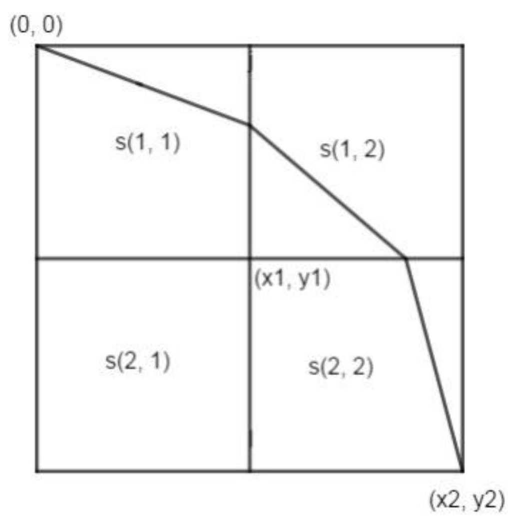

Figure 3

However, for a general nxm grid, the problem becomes far more difficult as we have more variables to solve for.

For example:

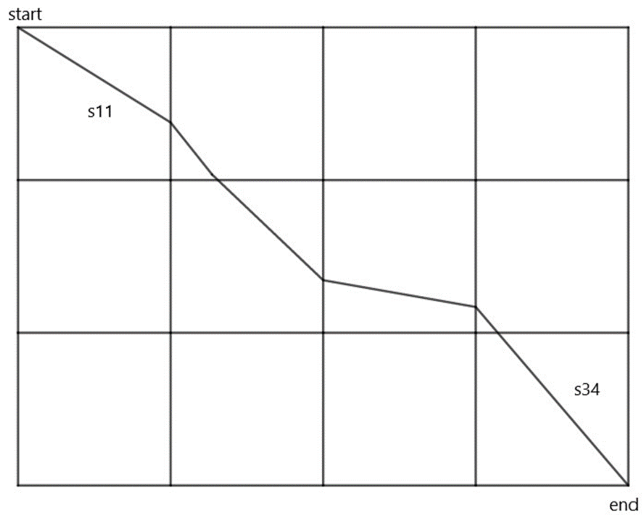

Figure 4

involves 5 parameters over which to optimize the path.

Hence, to deal with the general case, we shall approximate a solution.

Approximating a Solution

Approximate a solution by using the centers of each square-which we will call nodes. We shall create a weighted grid where the weights are computed from the sij and will be the time between nodes. We will create an optimal path (timewise) through this grid

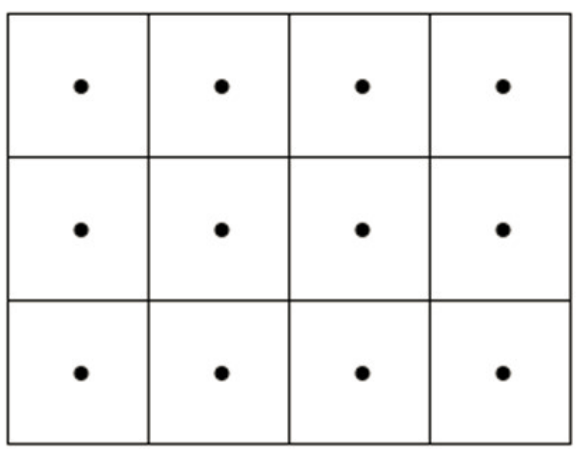

Figure 5

then complete our approximation of an optimal path as follows:

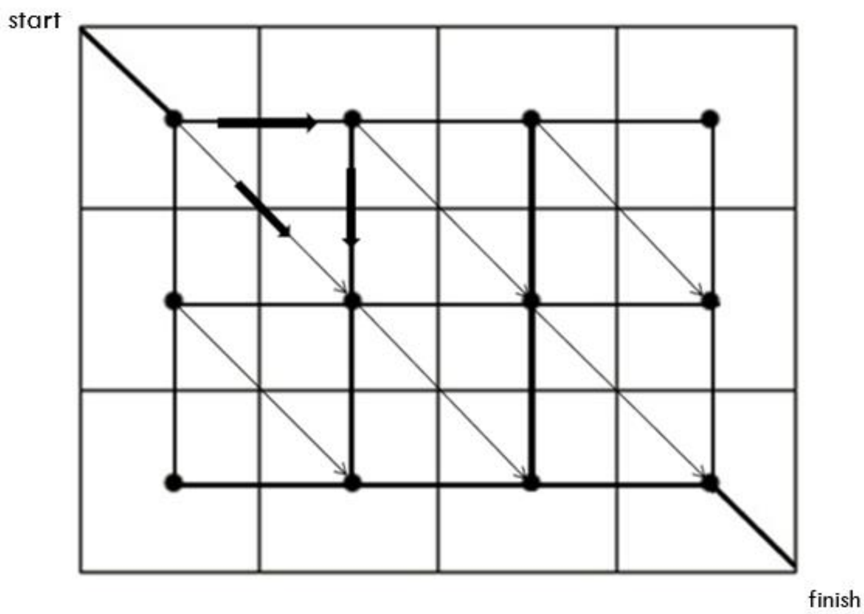

Figure 6

As shown in Figure 6, we shall consider a rectangular grid of n rows and m columns of nodes. As the objective is to move from the top-left to the bottom-right of the grid, we shall allow three types of movements between nodes:

- One node right
- One node down
- One node diagonally right/down

Between each pair of nodes is an edge giving the time it takes to travel between the two nodes.

Figure 7 below shows a 3x4 grid, the possible movements between nodes, three time units as an example of the time between nodes, and a sample path.

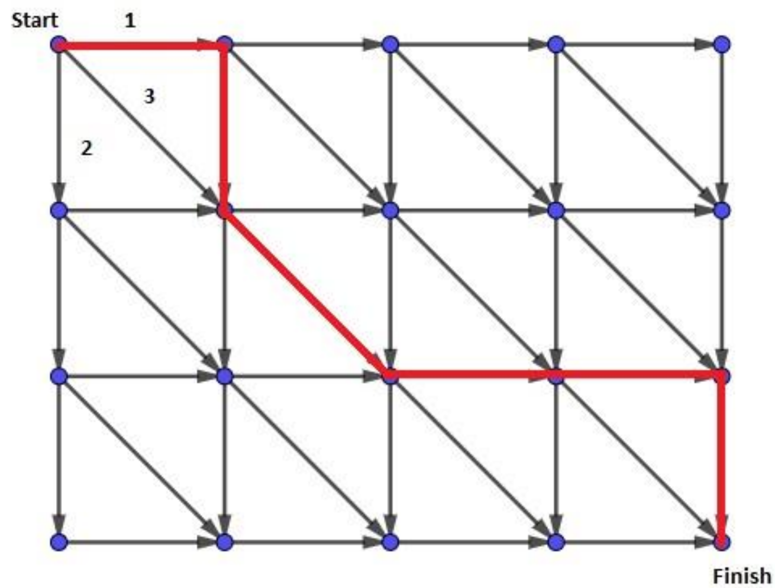

Figure 7

The number of possible paths through a nxm grid, where movement between nodes is as defined above, is given by Delannoy Numbers D(n, m) [5] defined by:

$$D(m,n) = \sum_{k=0}^{\min(m,n)} \binom{m+n-k}{m} \binom{m}{k}$$

For our illustration in Figure 7: there are D(3,4)-129 possible paths.

Note: We can compute the number of paths D(m, n) using: 1) combinatorics, 2) difference equations, and 3) generating functions [5],[6]. The number of such paths are defined as Delannoy numbers, who studied such paths (without considering weights between nodes).

Table of Delannoy Numbers. Rows and columns begin at 0.

n \ m	0	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1	1
1	1	3	5	7	9	11	13	15	17
2	1	5	13	25	41	61	85	113	145
3	1	7	25	63	129	231	377	575	833
4	1	9	41	129	321	681	1289	2241	3649
5	1	11	61	231	681	1683	3653	7183	13073
6	1	13	85	377	1289	3653	8989	19825	40081
7	1	15	113	575	2241	7183	19825	48639	108545
8	1	17	145	833	3649	13073	40081	108545	265729
9	1	19	181	1159	5641	22363	75517	224143	598417

Figure 8

For small problems, using the Java programming language, we can write a program using tree diagrams and lists to generate a list of all possible paths.

The beginning a such a program is given as follows:

```
public class GridPaths3 {
    static class TreeNode {
        int val;
        List<TreeNode> children;
        public TreeNode(int val) {
            this.val = val;
            this.children = new ArrayList<>();
        }
    }
```

(This is a good problem for a class in computer programming.)

Example

Given the following weights on our grid which represent the times between nodes

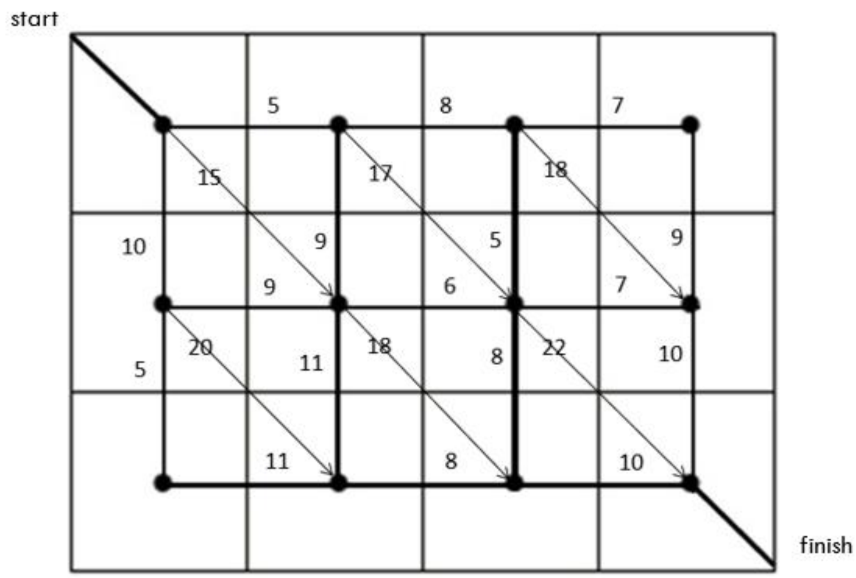

Figure 9

Using Dijkstra's algorithm [3],[4], we get the following optimal path through the nodes:

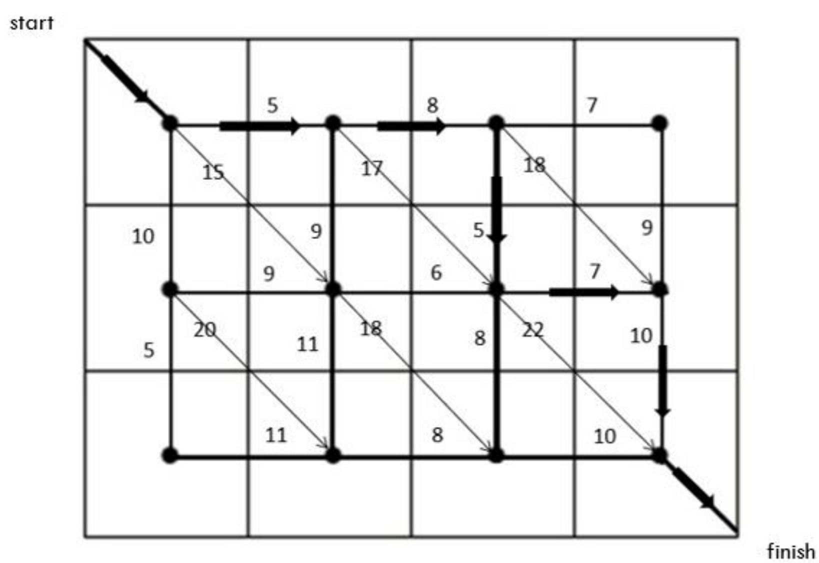

Figure 10

Improvements

Improvement 1:

Write a program using numerical techniques [1] to alter the position of the red nodes to get a smaller travel time.

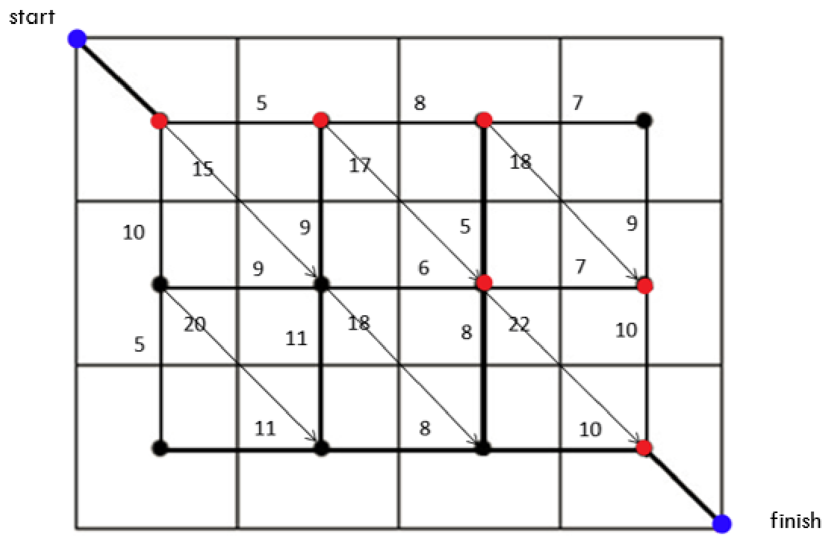

Figure 11

Improvement 2:

Once a node is reached, recalculate the optimal path based on changing weather and road conditions.

Improvement 3:

Model the times between nodes as random variables and find the route that has the greatest probability of getting to the final destination in a given period of time or less.

Using GeoGebra

Go back to a 2x2 example of our original problem:

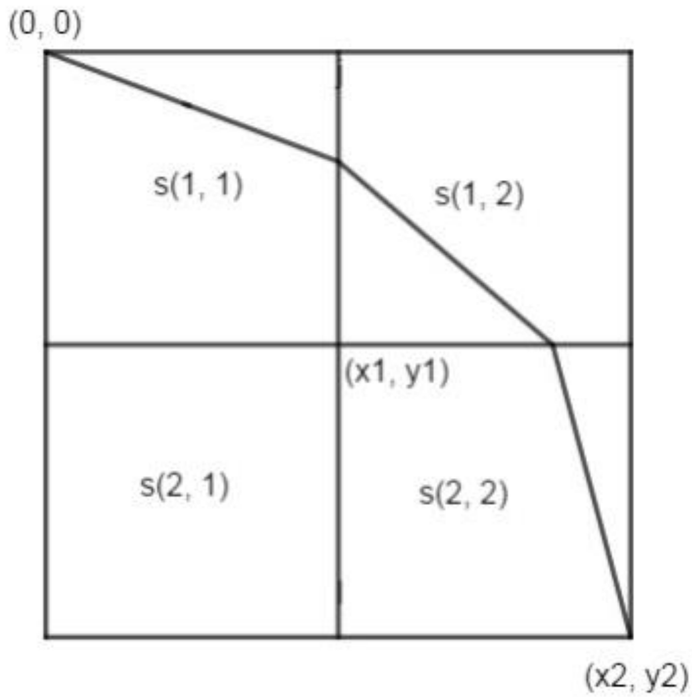

Figure 3

Our goal is to find a path from (0, 0) to (x2, y2) in minimum time.

Letting

s(1, 1)=10, s(1,2)=50, s(2, 1)=30, and s(2,2)=20 and
(x1, y1)=(100, 100) and (x2, y2)=(200, 200)
We can easily see the fastest route is through squares I, II, and IV-as illustrated above.

Using GeoGebra we can create the following file:

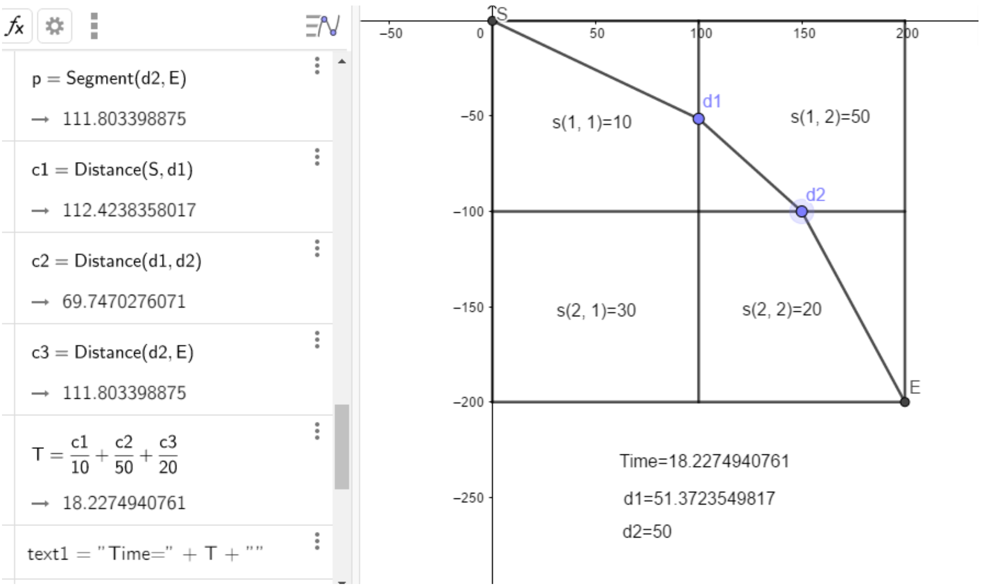

Figure 12

Creating the nodes d1 and d2 and deriving equations for the time is takes along the path defined by d1 and d2 by moving d1 and d2 we can quickly find d1 and d2 as follows:

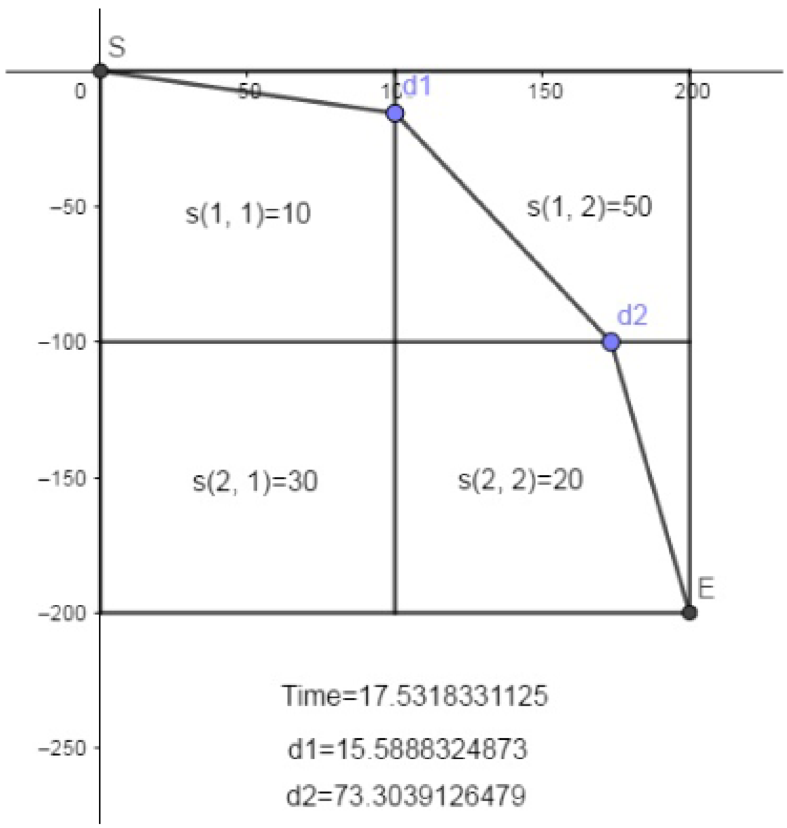

Figure 13

It can be shown [2] that the actual fastest time though the grid is T=17.53175658 time units. This method can be extended to the general mxn case using the above techniques.

Summary

To attain the minimum time through a weighted grid:

- For small cases, say a 2x2 case-the problem can be solved by numerical analysis.
- For larger cases, approximate the solution using a weighted grid and Dijkstra's algorithm.
- Then use this first approximation and Monte-Carlo methods to get a better approximation of the optimal solution.
- Another problem that needs to be studied is that of the nonuniqueness of solutions. If multiple solutions yield the same minimum time we could choose the shortest path.

References:

[1] *Lagrange Multipliers and the Calculus of Variation in Game Design* Proceedings of the 2020 ICTCM https://www.pearson.com/content/dam/one-dot-com/one-dot-com/us/en/files/ICTCM20-Proceedings-Bouthellier.pdf

[2] *Creating a Path of Fastest Descent Through a (2x2) Gird* Proceedings of the 2022 ICTCM https://www.pearson.com/content/dam/one-dot-com/one-dot-com/us/en/files/ICTCM22bouthelliericctcm2022.pdf

[3] *Dijkstra's Algorithm* https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[4] *Dijkstra's Algorithm*: The Shortest Path Algorithm https://www.analyticssteps.com/blogs/dijkstras-algorithm-shortest-path-algorithm

[5] *Delannoy Number* https://en.wikipedia.org/wiki/Delannoy_number

[6] *Delannoy Number* https://mathemathworld.wolfram.com/DalannoyNumber.html