EleNa: Elevation-based Navigation

The Heisenbug_Reloaded Team
Ashish Ranjan
Abhishek Somani
Srikanth Prabala
Deep Chakraborty
Carlos Daniel Mondragón Chapa

Purpose

The usual Navigation System tries to get the shortest or the fastest path from one point to another. However, in some cases, having the ability to further customize the path is desirable. Such a case could arise if, while working out, following a path that maximizes or minimizes the elevation gain is more convenient than the usual shortest path in order to get better training results. Therefore, the EleNa project tries to find the best path between two points according to the user preferences regarding elevation gain and some distance constraint with respect to the shortest path.

Features

- EleNa calculates the path between two points that maximize or minimize the elevation gain constrained to a percentage increase with respect to the shortest path.
- The origin and destination points along with the distance constraint are set using the command line.
- The origin and destination points are set using coordinate values.
- The resulting path is drawn on a map.

Use cases

The user can set an origin and a destination by using the corresponding coordinates (latitude and longitude) and a distance constraint determined by a percentage that represents the extra distance with respect to the shortest path. Once the coordinates and the distance constraint are set, the user can choose to minimize or maximize the elevation gain. The algorithm will then output the calculated path on a map.

Model Proposals

All models calculate the resulting path within a preselected area in the map. The area contains the origin and the destination and it can be seen as a rectangle that contains all the possible paths.

1. The first proposal does an exhaustive search of all the possible paths from origin to destination. It then finds the elevation gain in each path and chooses the optimal path depending on the user settings. This algorithm works in a reasonable amount of time if the desired extra distance to be travelled with respect to the shortest path is not larger than 10%.

2. The second proposal is much faster than the first one as it uses Dijkstra's algorithm with a slight modification. Instead of taking into consideration only the elevation gain, the distance between nodes, or some impedance function that gives weight to the edges connecting the nodes; the algorithm uses the same greedy approach by selecting the best node based on the lowest elevation gain, but then looks into the next possible node based on the shortest distance. This approach guarantees a shorter elevation gain compared to that of the shortest path, but keeps the increase in distance relative to the shortest path within the desired percentage.
3. The third proposal finds the shortest path from origin to destination and gets the amount of nodes along that path; the amount of nodes in the shortest path is NSP. The algorithm then finds all the paths from origin to destination that have at most MNP nodes such that $MNP > NSP$. The difference between MNP and NSP is some value, v, that was empirically chosen by gradually trying different numbers and checking the performance of the algorithm with each one. Finally, the algorithm picks the path with the lowest elevation gain from all the paths that contain $NSP + v$ or less nodes.

Experimental Design

For all the models, the same origins and destinations are randomly generated. Then, the shortest path (SP) and the path with the least elevation gain (LEG) is calculated for each origin and destination point using Dijkstra's algorithm. Afterwards, all algorithms are tested against the output of Dijkstra's algorithm. If the algorithm works properly, then the distances along the paths and the elevation gains should be similar.

Suppose that Model 1 outputs a path MoP_1, if we compare this path to SP it should be within the constraint set by the user and have a lower elevation gain, which, at the same time, should be close to the one in LEG.

Findings and Conclusions

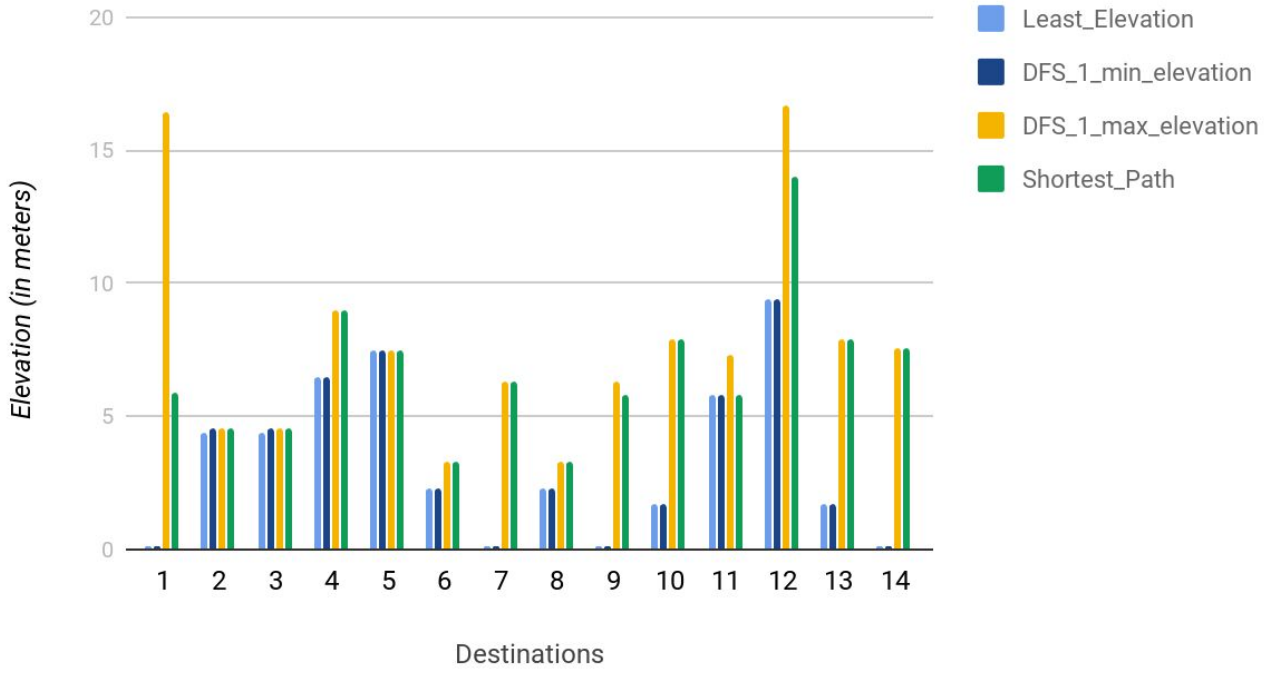
1. Model 1

Below are the results of the first Model after testing it with 14 different destinations. Graph 1 depicts the elevation gain in meters for the different paths. The goal of the algorithm is to find a path that has less elevation gain than the shortest path. As can be seen, all the paths achieve the desired result whenever it is possible. Similarly, the elevation gain is maximized whenever possible. The algorithm is able to find a path that has more elevation gain than the shortest path most of the times.

Nevertheless, for all the results to be meaningful, the distance from origin to destination should be close to the shortest path. Graph 2 makes it easy to see that in every case the algorithm was able to find a path that was the same as the shortest path in terms of distance, while maximizing or minimizing the elevation gain.

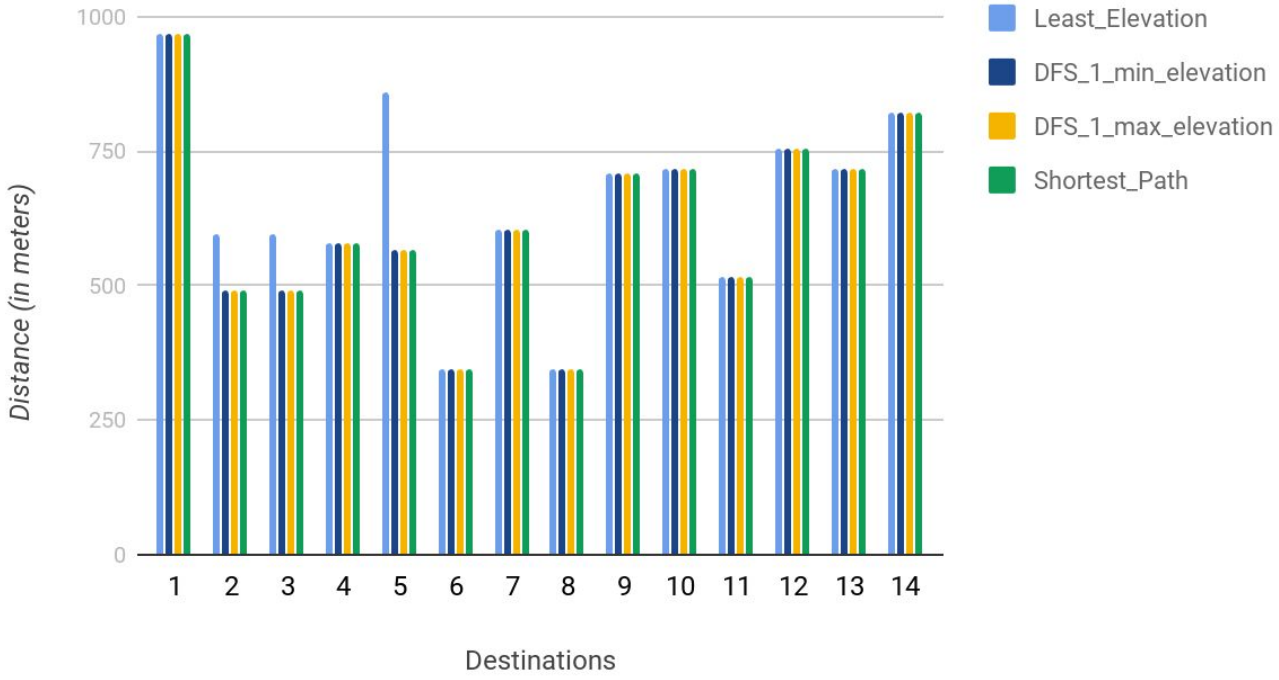
It is important to mention, however, that the origin and destination were very close from each other (less than 1 km). Otherwise, the algorithm would have taken much more time to compute for 14 different destinations. Also, this means that using this algorithm in real situations would be impossible, as the user would have to wait for a long time to get a single result.

Path Elevation comparison for DFS1



Graph 1

Path Length comparison for DFS1



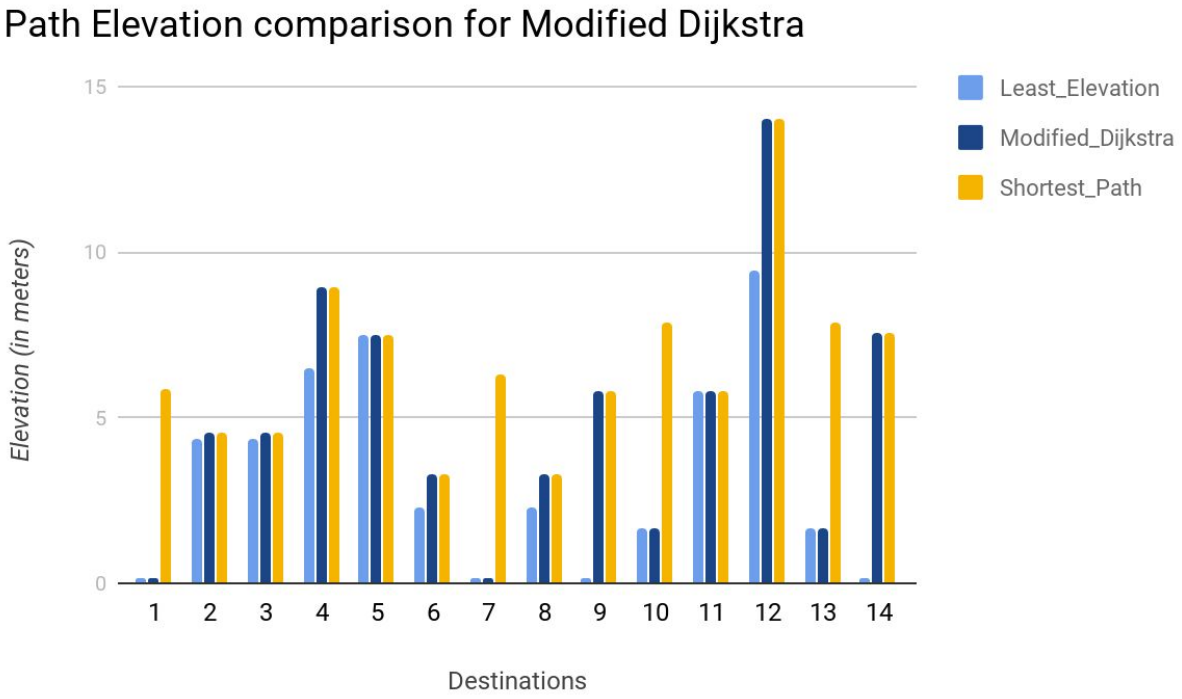
Graph 2

The algorithm also outputs the calculated path in a map. As can be seen, both the path on the left and the right are of the same distance; the only difference is that the path on the left contains the least elevation gain, whereas the right one has the maximum elevation gain from origin to destination.

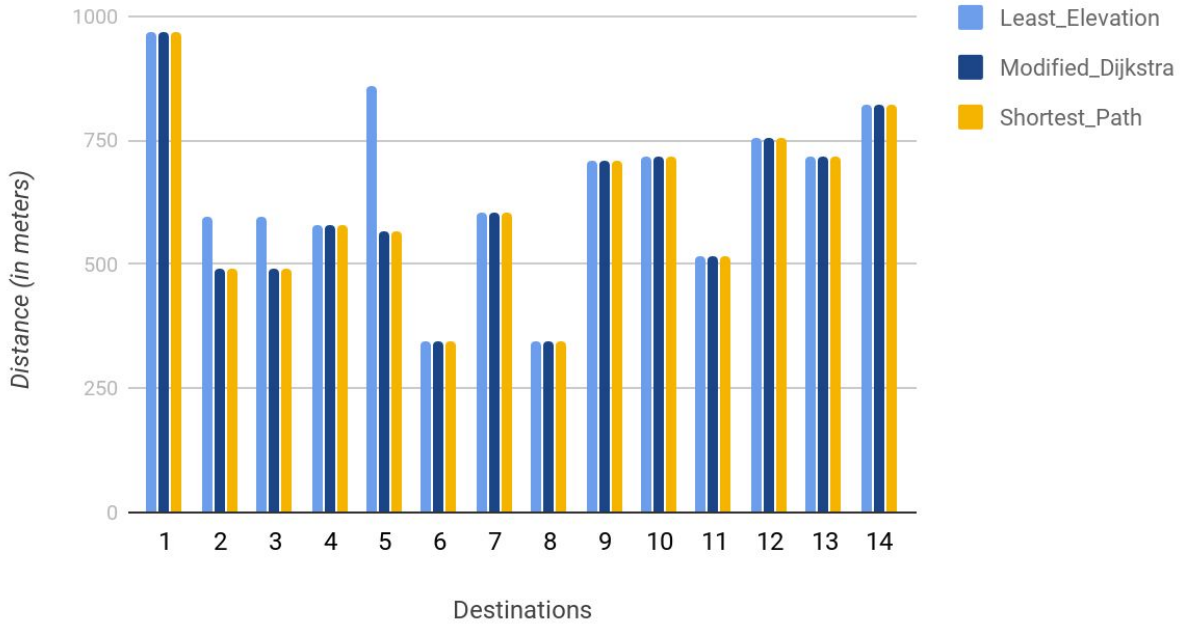
2. Model 2

This model is arguably the best of the three, since, regardless of the distance between origin and destination, it is able to arrive to a reasonable answer in a short amount of time. One of its problems, however, is that the model is only capable of calculating the path with the least elevation gain. Therefore, only the paths with the least elevation gain found by the algorithm are shown the in graphs below. Graph 3 shows how nine out of the fourteen times, the algorithm was able to find a path that was either the best or reasonably close to the optimal path. Most of the times the path with the optimal elevation gain was not chosen, because of the distance constraint, as shown in Graph 4. Also in many cases, since the difference in elevation gain between the shortest path and the optimal path is not very different, the algorithm keeps the shortest path.

Nevertheless, there were situations when the algorithm did not perform very well: Destination fourteen clearly has a path that is much better than the shortest path in terms of elevation gain and requires the user to travel the same amount of distance. This problem hardly ever happens, but it is important to mention that the algorithm has flaws, although it is considered to be the best among the three because of the computation time.



Graph 3



Graph 4

Similarly, Model 2 outputs the path in the map.

3. Model 3

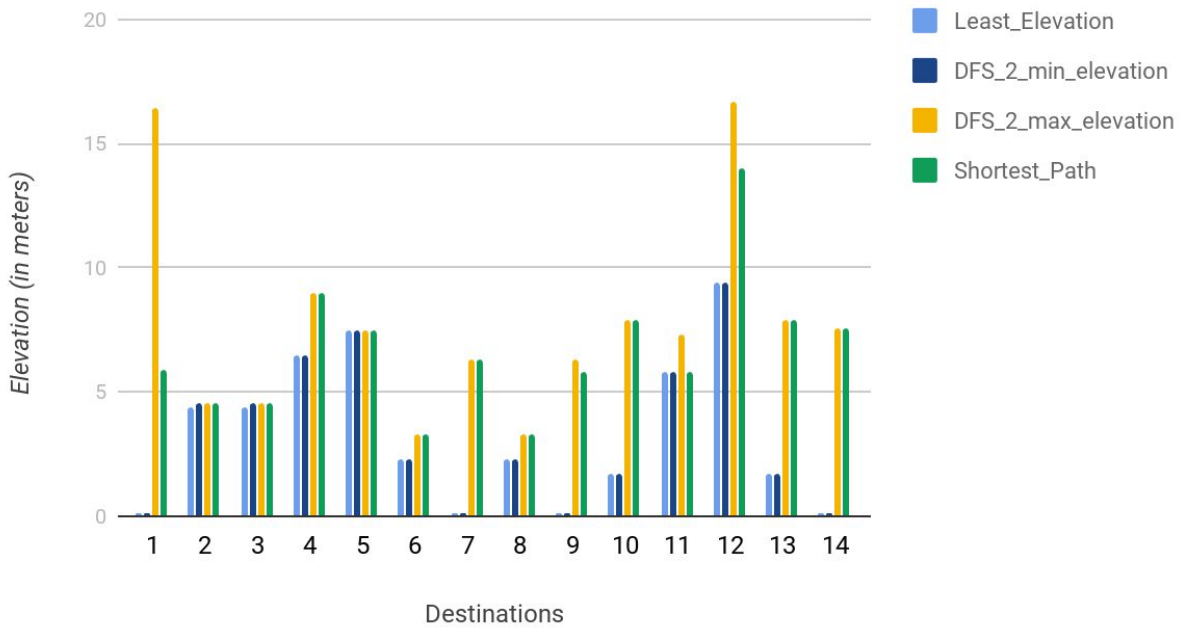
The third model is very similar to the first model, as it also does an exhaustive search of all the paths that are within the distance constraint placed by the user and calculates all the elevation gains. The difference is that it restricts the distance of the resulting path using the amount of nodes between origin and destination. Therefore, this model is slightly faster than the first one. Nevertheless, it is not fast enough to be compared to the second model.

Since the model is so similar to the first one, the algorithm achieves the same results for the same origins and destinations. Graph 5 depicts the elevation gain in meters for the different paths. The goal of the algorithm is to find a path that has less elevation gain than the shortest path. As can be seen, all the paths achieve the desired result whenever it is possible. Similarly, the elevation gain is maximized whenever possible. The algorithm is able to find a path that has more elevation gain than the shortest path most of the times.

Similarly, it can be shown that the results are valid by looking at Graph 6. Most of the times, the algorithm was able to find a path that was the same as the shortest path in terms of distance, while maximizing or minimizing the elevation gain.

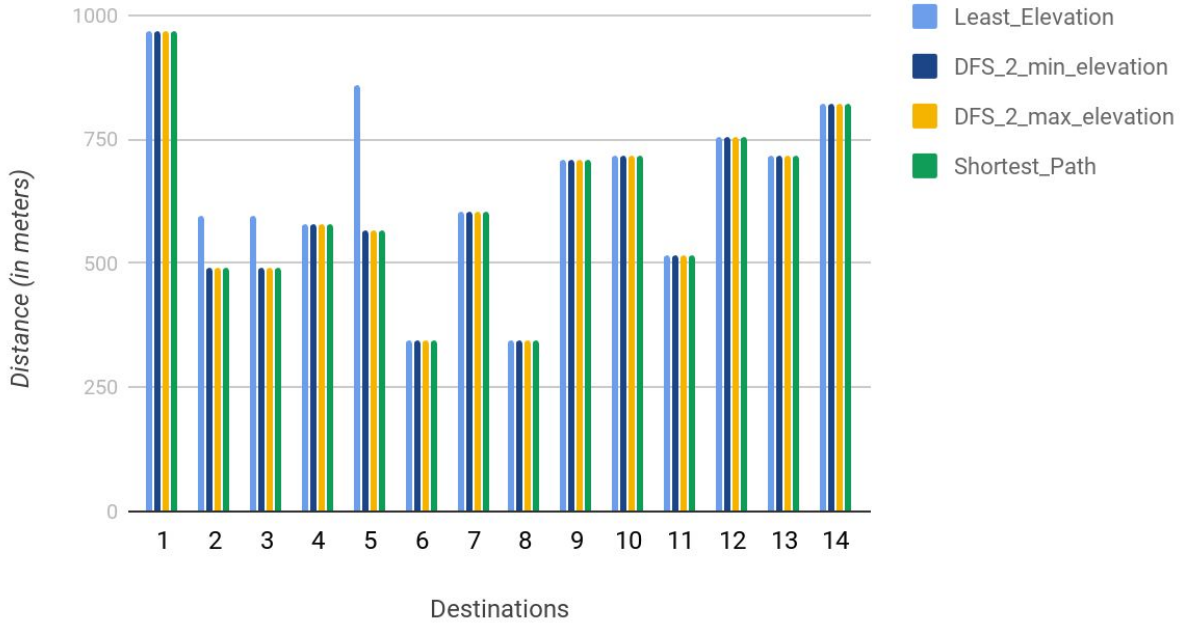
Also, just like in Model 1, the origin and destination were very close from each other (less than 1 km). Otherwise, the algorithm would have taken much more time to compute for 14 different destinations; which means that using this algorithm in real situations would be impossible, as the user would have to wait for a long time to get a single result.

Path Elevation comparison for DFS2



Graph 5

Path Length comparison for DFS2



Graph 6

Achieving least elevation gain is a trivial problem because of Dijkstra's algorithm, otherwise, an exhaustive search of all the paths would have been needed. However, adding a distance constraint to the elevation gain makes the problem much harder, since the path with the least elevation gain may not be within the desired distance constraint. Therefore, finding the optimal path in long distances takes too long if all the paths want to be checked. In case of finding the maximum elevation gain, the same problem is encountered, since the algorithm could potentially take the user all the way to the Alps before arriving to the destination.

All the proposed models would not take the user that far; nevertheless, finding the optimal path would take an exponential amount of time for the first and third model as the distance between the origin and destination increases. The second algorithm would perform extremely well, computationally speaking, even in large distances; however, in some cases it will not get the optimal path. It can only guarantee that the resulting path is within the user's distance constraint and that the elevation gain is not larger than that of the shortest path. Moreover, this model cannot find the maximum elevation gain and as the results suggested, none of the models would be able to find this result in a reasonable amount of time.