Foundations of Intelligent Systems, Lab1

Hitesh Sapkota

October, 11, 2018

1 Input Interpretation

We are given a two inputs. First one is a map, dimension of 395×500, with the appropriate color values according to the pixel coordinates. Second one is a text representation of the elevation corresponding to each pixel in the map. For the Classic event testing, we are given a three text files: white, brown, and red. Similarly, for the Score-O, we are given two files: control points corresponding to a nice hilly area, and the control points representing all over the park.

2 Task to be Performed

First, we need to define a two cost functions: g(n) and h(n). The g(n) gives a cost from source to the given point and h(n), also called heuristic cost function, gives a estimated cost from given point to a destination. When choosing a heuristic function we need to make sure that it is admissible and optimal. Once we define g(n) and h(n), we perform A* search to find the optimal path between each pair of the controls.

In case of the Classic event, we visit the controls in a particular order. We apply A* starting from first two consecutive pair. This gives a optimal path between first two controls. We repeat this process until we get the optimal path between all consecutive pairs starting from source to destination.

In case of the Score-O, we first find out the order of the given control points in such a way that we travel the optimum path. Once we get the order, we use the same process as that of the Classic event.

In the subsequent sections, we describe terrains factors and our approach to compute the h(n) and g(n). Then, we describe about the A* search followed by the different event types result. Finally, we describe about the implementation.

3 Terrains

I have hard-coded the terrain factor for each terrain type. The terrain factor for the given terrain type are as follows:

- Open Land :4

- Rough meadow :0.5
- Easy movement forest :2.5
- Slow run forest :2
- Walk forest :1
- Impassible vegetation :0
- Lake/Swamp/Marsh :0
- Paved road :4
- Footpath :3
- Out of bounds :0

I assume that we can achieve a maximum speed to the open land, and paved road. Similarly, we have the minimum speed in the rough meadow. Further, we can not travel through the impassable vegetation, lake, swamp and, marsh. The factors are defined accordingly.

4 Cost so far from a source (g(n))

We compute the g(n) for the pixel x using following formula,

g(n)=$prev\_g(n)$+ cost to travel in the pixel x (c(x))

We determine the c(x) as the ratio of the Euclidean distance of the pixel x (d_x) and the speed in the particular pixel (s_x) i.e.,

c(x)= d_x/s_x,

The Euclidean distance is computed as ,

$d_x = \sqrt{x^2 + y^2}$,

where, x is the longitude and y is the latitude.

We compute the speed based on the assumption that a person has a avergae walking speed of 1.388m/s. The slope is given as,

s_x=$1.388 \times terrainfactor_x \times norm\_slope_x$,

We determine the slope as,

$slope_x = (elevation_y - elevation_x)/d_x$

Where, $elevation_y$ is the elevation of the previous (adjacent) pixel.

If $elevation_x$ is greater than $elevation_y$, that indicates we are going into the upward direction. So, the slope becomes negative where as if the case is opposite then we go to the downward direction and the slope becomes positive.

Since, the slope can be negative. So, we normalize the slope between 0 and 1 using following formula,

$norm\_slope_x = (slope_x - min\_slope)/(max\_slope - min\_slope)$

Here, the maximum and minimum slope are pre-computed from the elevation. In our case,

$min\_slope = -1.354$, $max\_slope = 1.354$.

During normalization, more steep the hill in the downward direction, higher the slope and vice versa. This means, if there is more sloppy downward hill then our speed increases.

5 Heuristic function (h(n))

The heuristic function is defined as the ratio of the Euclidean distance between current pixel to the destination divided by the estimated speed i.e.,

$h(n) = d_{xd}/est\_speed$

Where, we define the estimated speed as,

$est\_speed = 1.388 \times terrain\_factor \times norm\_slope$

In order to find the terrain factor, we follow following procedure,

- randomly pick p-pixels enclosed in the areas of (x_c, y_c) and (x_d, y_d). Where, (x_c, y_c) and (x_d, y_d) are the coordinates of the current pixel and the destination pixel respectively.
- find the pixel with the maximum terrain factor.
- find the terrain factor for the corresponding pixel

This guarantees the admissibility. This is because, if we choose the pixel having a maximum terrain factor, for a constant slope, we get the maximum achievable speed in the path. This indicates the minimum heuristic cost

In order to compute the slope, we follow the similar procedure as that of for the terrain factor.

- randomly pick the p-pixels enclosed in the areas of (x_c, y_c) and (x_d, y_d). Where, (x_c, y_c) and (x_d, y_d) are the coordinates of the current pixel and the destination pixel respectively.
- find the pixel with the maximum normalized slope.

6 A* Search

For the A* search, we start from the source node. For each given point, we determine all the neighbors present using a successor function. Then, we compute the total cost for each successor and then those nodes are added to the priority queue (with the total time as a priority). Before inserting to the priority queue, we discard the successor nodes if they are already the parent of the given node or is of impassable type or is out of bound node. We then pick the node with the minimum cost as our next node. We continue this process until we find the destination with the minimum cost.

7 Classic Event

In this case, we have a number of controls and we need to visit them in the sequence. For this, we pass the current control point and the subsequent control point as the source and destination in A* search algorithm. We continue this until we visit all the points in the given sequence.

8 Score-O Event

In case of the Score-O event, we can visit the given controls in any order but we need to make sure that we return to the origin in a most optimal way. This is the travelling salesman problem. In order to solve this we use the Held-Karp algorithm. Held-Karp is the dynamic programming approach to find the optimal route in the given set of points. We first compute the heuristic cost function from the one control point to another. Note that the heuristic cost function is asymmetric between pair of nodes. Then, we pass the adjacency matrix to the Held-Karp, which returns the optimal sequence of the control points to visit.

After getting the order of the control points to visit, we follow the procedure same as that of the Classic Event.

9 Planning for someone else

We change the cost function by modifying the terrain factors and effect of the slope. Modification on each are discussed below:

9.1 Modification on the terrain factors

We make the following assumptions before defining the factors:

- someone walks faster in the open land than in the paved road.
- someone is habitual and fast to walk on a footpath than in a paved road.
- someone does not prefer the rough meadow and considers as a impassable.

Based on the assumptions, we redefine the terrain factors as:

- Open Land :4
- Rough meadow :0
- Easy movement forest :2.5
- Slow run forest :1.5
- Walk forest :1
- Impassible vegetation :0
- Lake/Swamp/Marsh :0
- Paved road :3
- Footpath :3.5
- Out of bounds :0

9.2 Modification on the Slope Effect

Suppose the assumed person walks slowly in both downhill as well as uphill in compassion to the straight path. In such case, we perform the normalization of the absolute value. In other words, we map the value in the range of [0, 1.354] to [1, 0].

10 Implementation

Implementation is performed into the Python3. Imported library includes colormap, sys, queue, PIL, numpy, math, and random.

11 Results

11.1 Classic Event

Figures 1, 2, 3 are the outputs for the white, brown, and red courses respectively. The red line is the optimal output path and black dots are the controls.

The time taken by each course is shown in the table 1 with the corresponding cost.

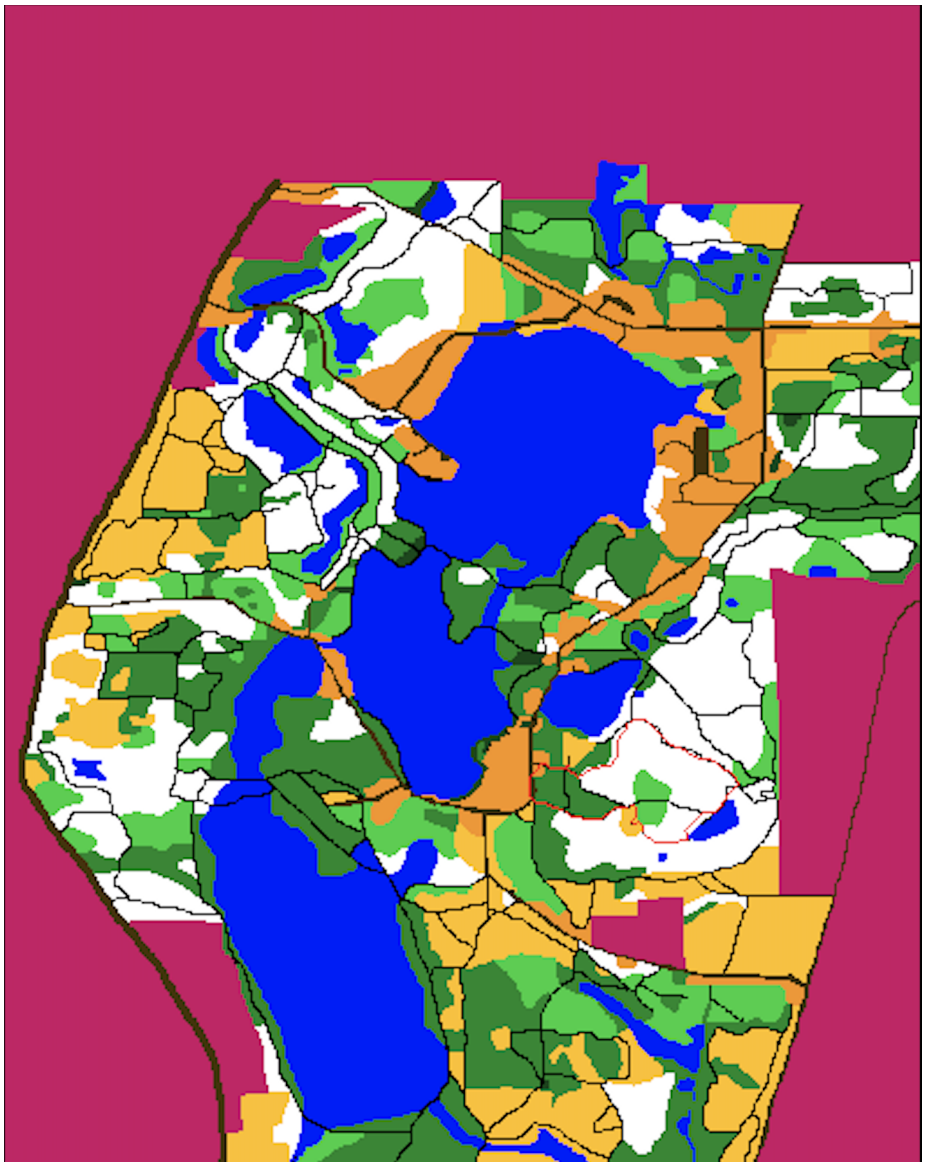

Figure 1: The corresponding route for the white courses

11.2 Score-O Event

The Figures 4, 5 below show the path for the controls provided into the esstesker and allpark respectively. The order of nice hilly control points to be visited:

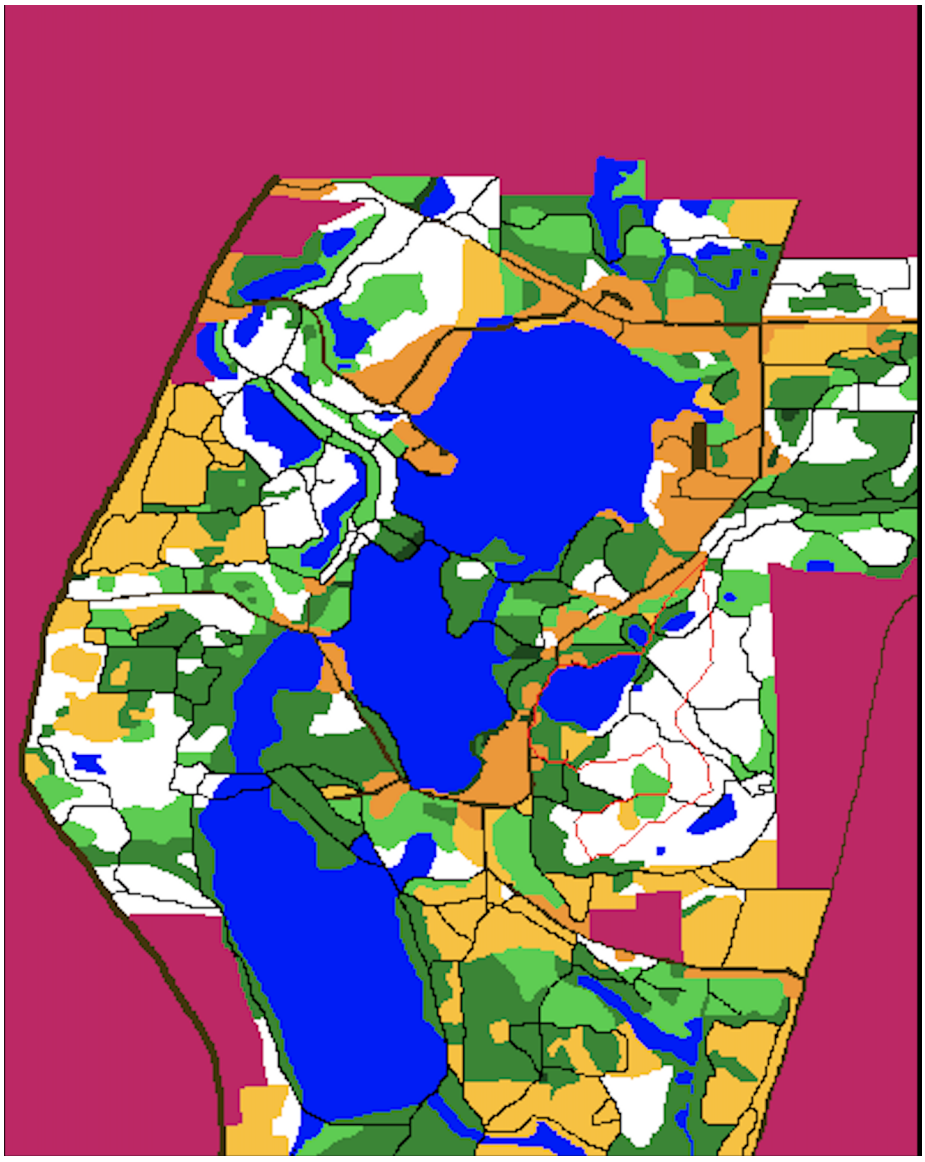

Figure 2: The corresponding route for the brown courses

$(230, 327), (292, 310), (372, 243), (339, 224), (317, 241), (306, 275), (299, 296), (306, 312), (325, 328)$, $(282, 329), (230, 327)$.

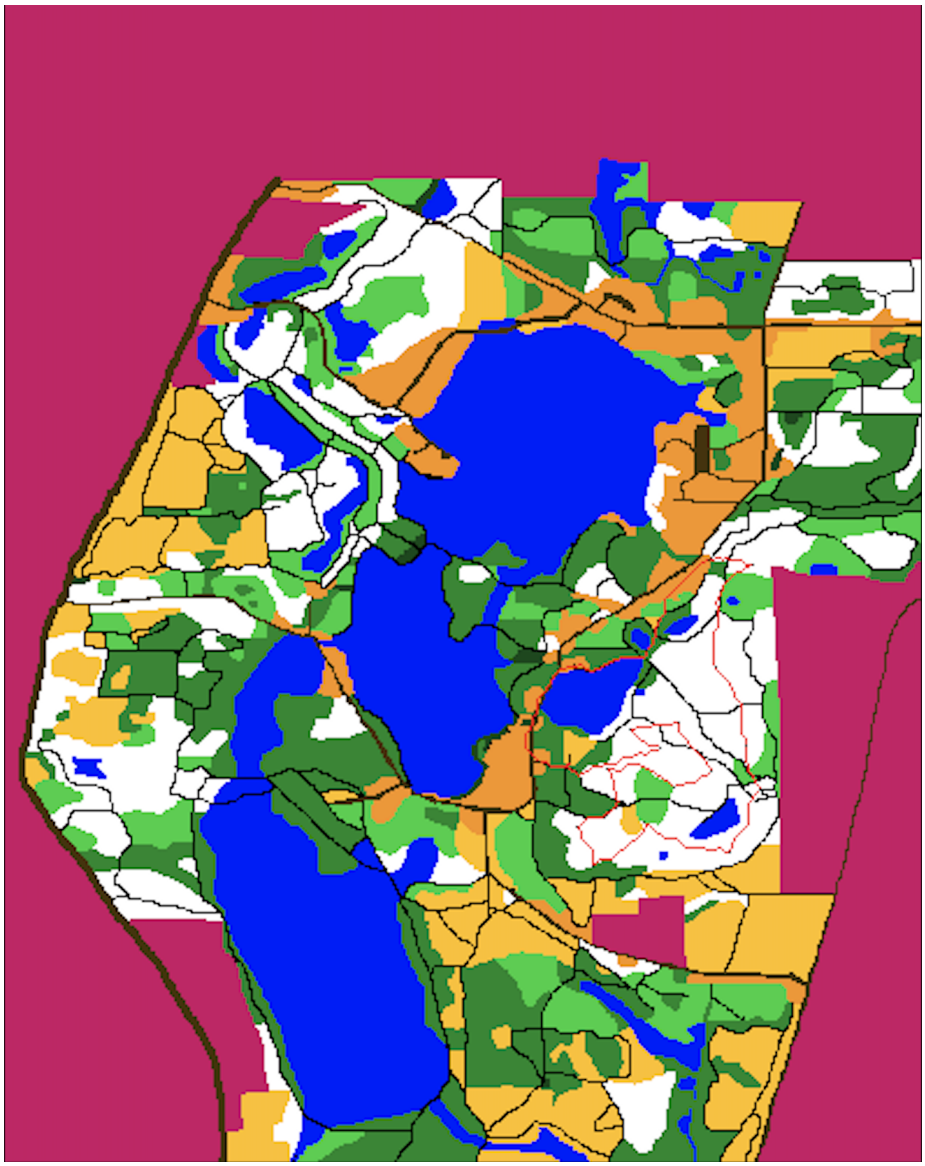

Figure 3: The corresponding route for the red courses

The sequence of all parks:

$(177, 192), (186, 137), (236, 91), (311, 201), (339, 224), (390, 111), (370, 166), (317, 241), (325, 328)$
$, (125, 229), (177, 192).$

Table 1: Time and Cost for different courses in second

Course type	Path Cost	Execution time
White	1470	2.56
Brown	2345	67
Red	3187	6.6

The table 2 compares the time taken by randomly control sequence versus ordered control sequence produced by the Held-karp algorithm. From table, we can conclude that Held-Karp produces the more optimal path than using random sequence.

Table 2: Time taken with or without using Held-karp.

Controls type	with Held-Karp			without Held-Karp		
	execution time	time limit	travel time	execution time	time limit	travel time
Nice hilly	11.66	3600	2731	21.87	3600	3180
All Parks	33.73	5000	4850	100.27	5000	7929

11.3 Planning for someone else

11.3.1 Classic Event

In case of the white course we are not being able to find the path. This is because one of the control lies in the impassable rough meadow region.

In case of the brown course, the overall path appears to same as before. Total path cost seems to be lower i.e., 1219 . This might be due to the effect of the slope we are considering. The execution time is similar.

In case of the red course, the path cost, 1694, is much lower than that of the original one with the comparable execution time. Moreover, it takes the slightly different path

From above, we can say that the latter has better path costs than the former one.

11.3.2 Score-O Event

In case of the nice hilly area points, the path cost seems to be very low, 1395, than previous one with comparable execution time.

In case of the all parks, the path cost seems to be, 2626, lower than the previous one. Also, the optimal path, shown in figure 6, is also shorter than the previous one

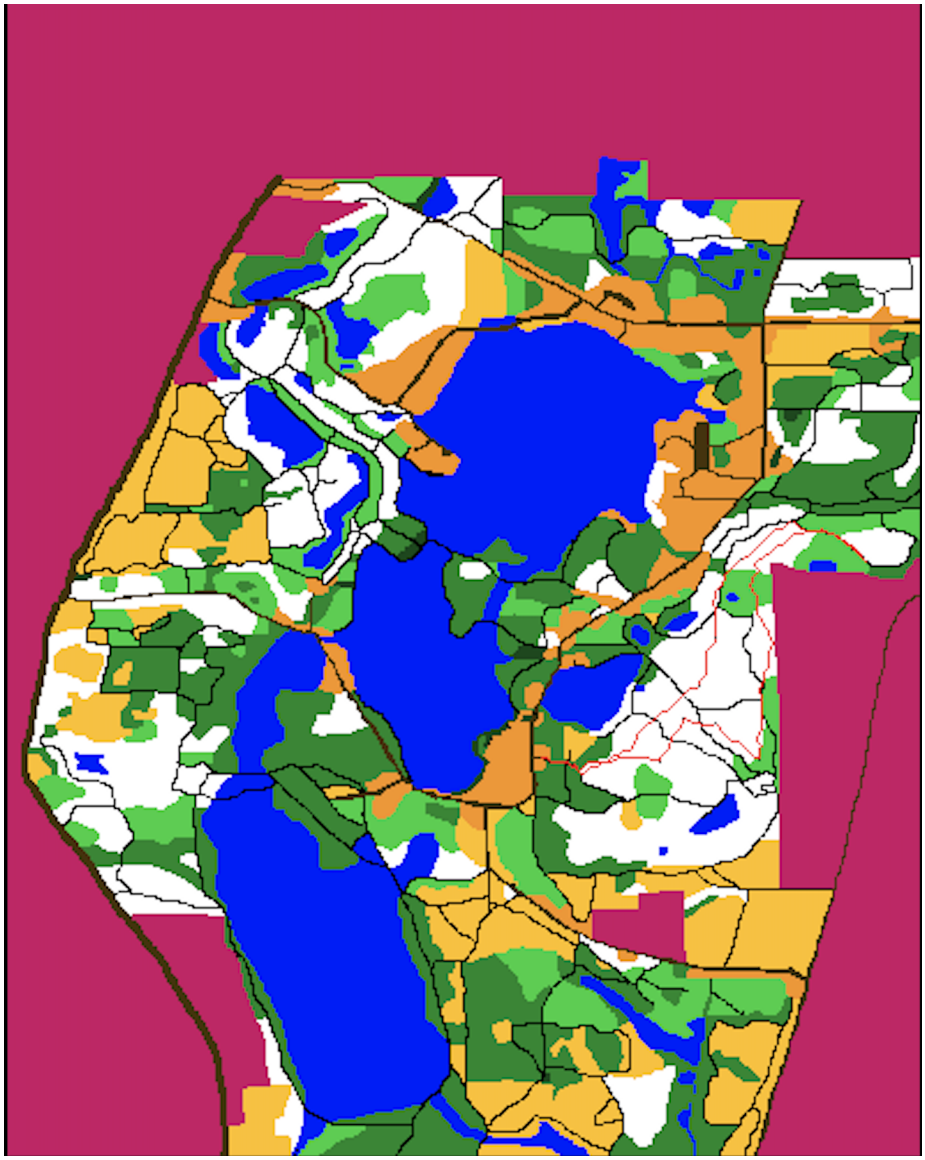

Figure 4: The corresponding route for the controls over a nice hilly area.

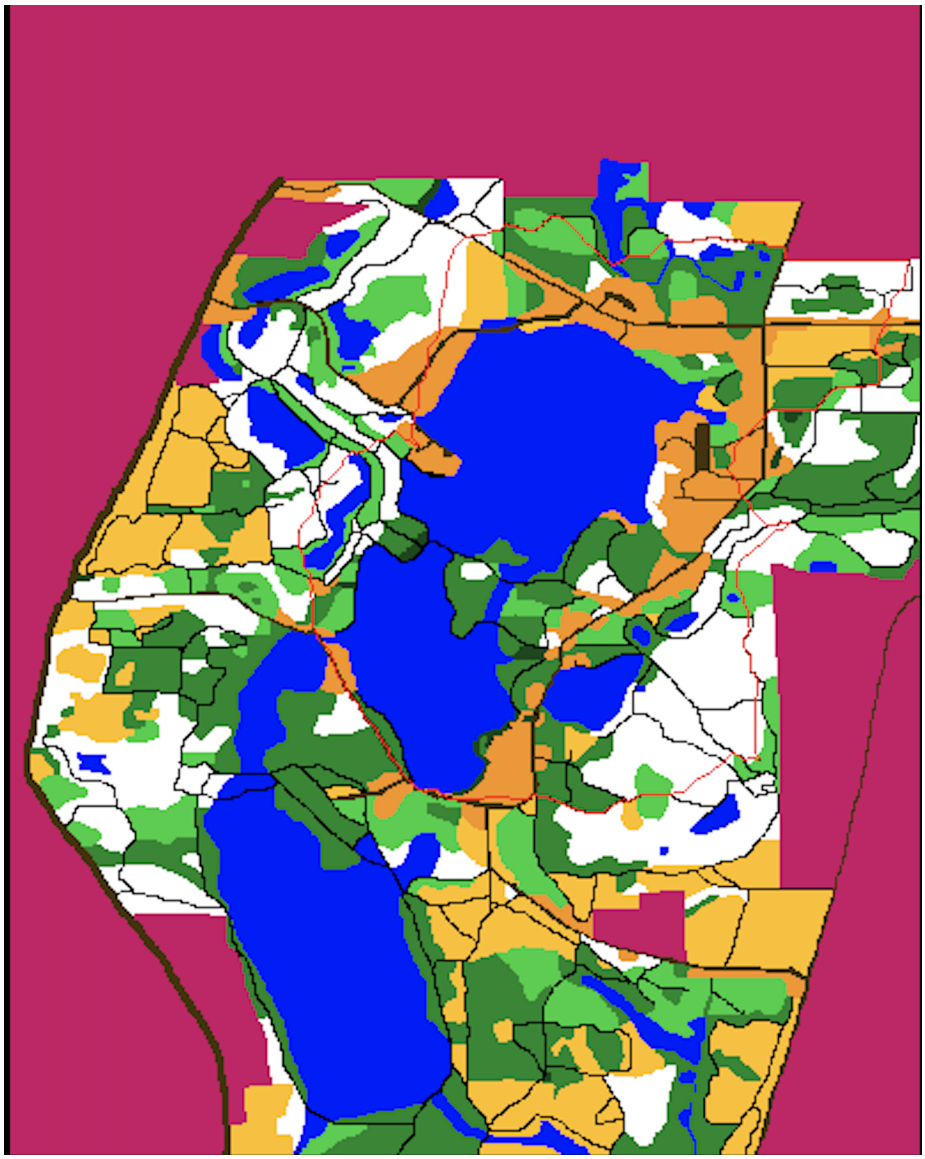

Figure 5: The corresponding route for all over the park.

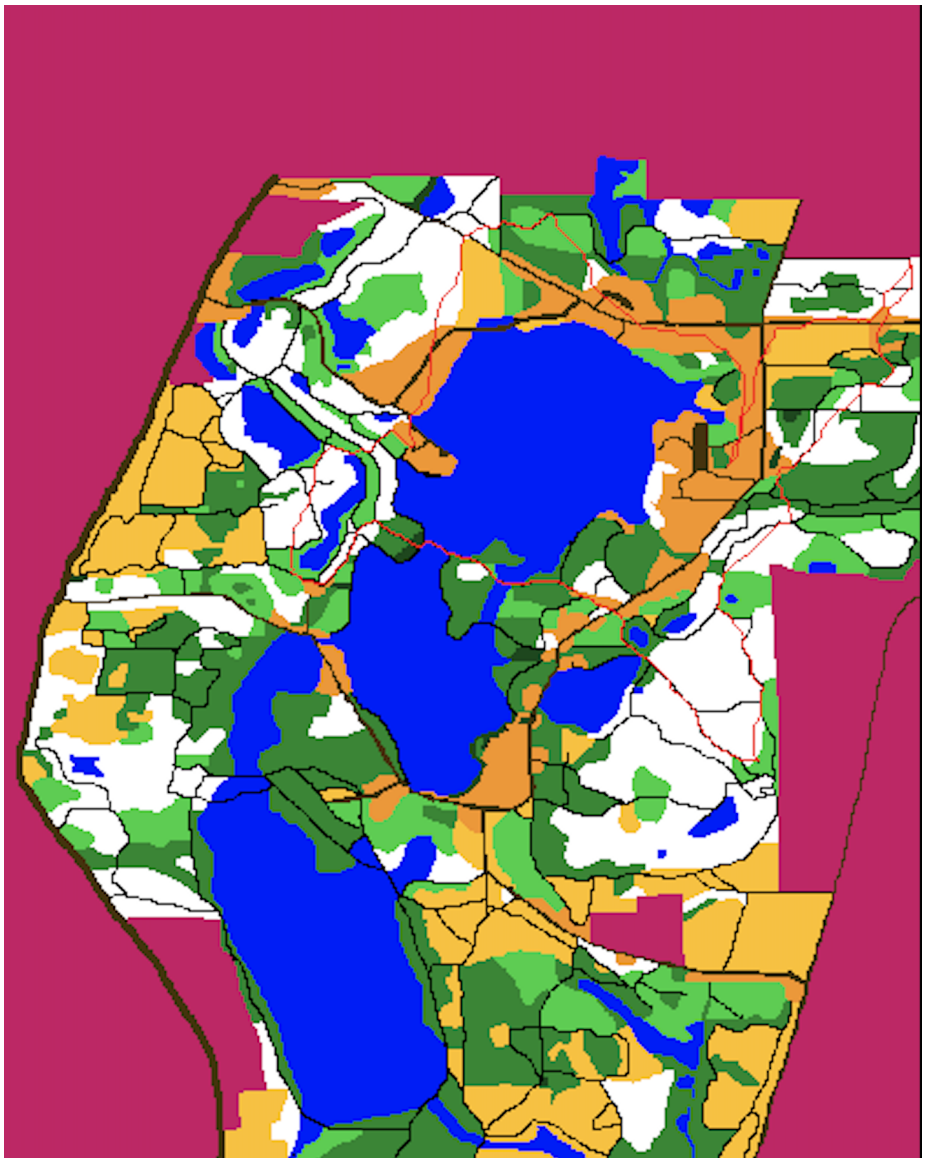

Figure 6: The corresponding route for the all over the park for different path costs