Automatically Generating Random Walking Trails: Goals and Expected Methodology

Terry Kang

October 25, 2017

Abstract

This document outlines the goals of the extra credit project and the expected methods used to achieve that project. The project is to design an application that can automatically generate walking trails in a particular area using a graph from data available on OpenStreetMap. The Euler Circuit algorithm and the A* algorithm will be used to generate a path along the graph that returns to its starting location, and does not visit an edge more than once, unless necessary.

1 Project Idea and Inspiration

The goal of the project is to create an application that can automatically generate trails to walk or run on. The generated trail, or path as the paper will refer to it from here on out, must meet these qualities.

- The path must start and end on the same starting point
- The path should not cross an edge[1] more than once, unless necessary[2]
- The path must be within the longitude, latitude borders defined by the user
- The path must fit, within some tolerance[3], the distance or time to expected to completion set by the user
- The path must visit points of interest, if specified and is possible[4].

I chose this project because I hope that it encourages me to go out more and visit new places. While I am quite fond of taking a run or walk, I found myself participating less in that activity. The areas that I usually walk along have been parks that I have to drive to get to. On top of that, many of those parks have already been exhausted in terms of site seeing. I have only been a resident of Georgia for 4 years, but I still don't think I explored what the state has to offer. All of these factors encouraged me build this program to hopefully visit new areas and go outside to take hikes more often.

2 Program Requirements

2.1 Project Goals

The program would take in several inputs. The user should enter two longitude, latitude coordinates that define the rectangular borders of the area they would like to walk in. The user then enters their starting location, and can then additionally enter either the distance or time of the expected walk. The program should have an option input for points of interest, where the user can enter in certain locations that they would like to visit along the path. The program then generate

[1] Edge is defined as the bridge joining two vertices as seen in graph theory.

[2] The idea of this trail is to allow exploration and sightseeing of new areas. Here necessary is defined as crossing an edge twice if the Eulerian Circuit halts without a loop

[3] This tolerance setting may be also set by the user as well

[4] This would occur if the specified points of interest is on a separate component of the graph specified

a path within the borders with a beginning and ending locations being the start location specified before. The path should also have a distance or time to complete within certain tolerances if specified and the path should go over each specified point of interest, if there is any. This path would most likely come in the form of a graphical outline of the path superimposed against a map. The program could also optionally export the path in the form of written instructions, telling the user on what road to turn and when.

This would be the base program, but if time permits there are stretch goals I would like to achieve. For example, having the program be either an app or web application that could be deployed onto a smartphone, so the user can track their progress on the path. Another stretch goal could be to add points of interest automatically based on review scores from a site like Tripadvisor using their api. These goals are, however, stretch goals and is not expected to be completed for the project.

2.2 Necessary Tools

The expected tools of the project involves OpenStreetMap. OpenStreetMap is crowd-sourced collection of map data. The data available on OpenStreetMap would be used to generate the graph. The dataset provided by OpenStreetMap is organized as a list of nodes, or vertices, and a list of ways, or edges, between each node. The data used here would differ from typical graph theory since each node and way would have a defined position in space. The data is formated in an xml file that can be accessed by either downloading the file before hand or getting the file through the OpenStreetmap api. This project would use the api.

The programming language Python 3.5 is expected to be used for its flexibility and its software packages. The scientific computation library numpy is expected to be used, allowing me to use powerful n-dimensional array objects for storing and manipulating the generated list of nodes that define the path. The library ElementTree is expected to be used to open and read the xml file. Matplotlib is a 2D plotting library for python, and this library is expected to be used to display the graph and the path.

Two primary algorithms are expected to be used. The Euler Circuit algorithm finds a Euler Circuit, which is a circuit in a graph that does not visit an edge more than once, and I would use the program for just that purpose. Since the program can halt when encountering odd nodes the algorithm would be modified to visit a way more than once if necessary (This is described in the expected methodology section below).

The A* algorithm, a modification of the Dijkstra's algorithm, finds a path between two nodes (this path is not guaranteed to be the shortest path possible, but has a nice balance between path distance and speed of execution). This would be used in the modification of the Euler Circuit algorithm when dealing with odd degrees.

All of this software is expected to be run on an ubuntu laptop.

3 Approach for the Project

3.1 Expected Methodology

Expected Methodology

Since each node is tagged with a road type in the the OpenStreetMap dataset, the graph should only consist of ways that can be walked on. The Euler Circuit algorithm shall be employed to generate a circuit where no way is being visited twice, but modified to prevent halting and allowing the repeated visits of an edge if necessary. The starting point of the algorithm would be the starting location specified in the inputs. The Euler Circuit algorithm would start as normal first employing the greedy algorithm to find a loop in the graph, adding to a list of nodes as the program runs. This list defines the ultimate path. If the greedy algorithm finds a node in the graph with an odd degree, unless the node is identified as "fixed even", that node is flagged. The program continues as before, but if the program finds another node of an odd degree, the program would look to the previous node in the list of nodes that have been flagged. The A* algorithm will then be employed from the second flagged node to the first flagged node. This path would then be appended to the list of node. The flags would be removed and replaced with an identifier that states the node as "fixed even", and the Euler circuit algorithm would continue as before. This modification should result in a path that loops back to the starting location and visits all nodes

available with a path that rarely goes over the same ways. The exception would be if the graph has several separated components. This possible issue will be ignored.

With this base path established, the program can then make modifications to the path to fit the parameters in the input section. The distance of the path can be found by finding the distance between the longitude and latitude coordinates of each node along the path, and converting that metric to kilometers. The length of the path can then be shortened to match the input parameters, by removing nodes in the path that do not contain the start node or points of interest and are bounded by the same node. The base path would also be modified randomly, cutting unnecessary paths out. This is to ensure the probability of the user getting the same path is low, the goal is to visit new places after all.