Version 3.1

I downloaded Notepad++ 2.7.5 on 3/24/15 after looking some on the internet. I configured it so that it would automatically save changes.

I also brainstormed ways to improve the efficiency of the algorithm. It comes in three ways: Use of Elevation API and some interpolation, reversing the directions of the queries to generate one-way roads and also change it so that intersections can only connect forwards, not backwards also, and streamlining the Branch algorithm. I will conduct research into how to extract the information from the Geocoding URL without violating CORS. As for interpolation, I had the idea that it could be done, since I have to simultaneously include provisions for the query generating results for all pts with the same coordinate; it may or may not be more efficient. For the intersections/Branch algorithm improvements, I came up with the idea of lopping off intersections that only have one connection, and merging intersections into one long path if they only have two connections. This will drastically cut down on computing time, and therefore decrease the amount of time needed to complete the calculations. Also, I had the idea of making intermediate significant points, so that the Branch optimization process is split into two steps. Significant intersections will have the most fuel efficient route found between it and the other nearby major intersections, and then those will ultimately have the algorithm applied to them. It will sacrifice some accuracy, but this is a heuristic change, and will likely be necessary for the algorithm to speed up.

3/24/15: 1 hour

3/25/15: 45 min

3/26/15: 45 min

I got a new editor, which is called Notepad++

3/27/15: 1.5 hours

3/28/15: 1 hour

3/31/15: 2 hours

I had the idea that it would be least complicated to find *only* the intersections that connected without data, because a) it would take more computing power to resort the data twice and b) be more complicated to code and. This way that I found, I find all the intersections from the original series data, and build lists of connections. Then, I strip intersections altogether that have only 1 or 2 connections, and rebuild the series data from the remaining intersections. Then, normal computation carries on.

This method appears to work, but does not noticeably lower amount of time required. Added an intersection limit.

I added a while loop to continually remove forbidden intersections until none remained. I also discovered that the connections arrays did not reset after each time, so they had artificially high numbers of intersections, nor did they count the same connection index only once. This was fixed, and the result was that only two intersections (start and end) remained. This was actually caused by an error in logic. The error was corrected, and the original basic while loop was replaced. The result is seen in output\_02.png. That does not represent all the intersections.

I timed the amount of time it takes to run parts of the program:

* Intersections: generation, adding signals, compiling data: 3193 ms
* Branches: calculation of: 13 ms
* Optimal: sorting through branches, finding the best route: 25 ms

*Sample: 6936 Millbridge Road to 330 Knollwood Street*

After some modifications (removing functions defined in loops and removing all traces of points having once been reversed) and the adding of backwards connections in addition to forwards to not remove quite so many intersections:

* Intersections: 3230 ms
  + Intersection\_gen(): 33 ms
  + Add\_signals(): 2346 ms
  + Intersection\_build(): 851 ms

I then made all functions inside the intersection\_build() function inside the Network function scope, and removed extra logic from the add\_signals() function, because the logic was not crucial to the performance of the algorithm. It essentially just double-checked things.

* Intersections: 929 ms
  + Intersection\_gen(): 39 ms
  + Add\_signals: 178 ms
  + Intersection\_build(): 713 ms

This is satisfactory. I now measured the branch algorithm for the same route, and it took more than 3 seconds to complete. I decided to shorten the maximum distance for the route to be 1.4 times the linear distance. The new time is in 900 ms range, which is also acceptable.

I also made all of the routes used in the directions service go both ways, and made the intersections algorithm search only forward, so it should only choose routes which the vehicle would go forward on, instead of illegal turns or wrong sides of the road.

4/1/15: 2.5 hours

Time: 9.5 hours