

Finding minimum energy expenditure route for user-selected mobility parameters and locations

Cole Anderson

Department of Geography, Environment, and Society

University of Minnesota Twin Cities, Minneapolis, MN

Github Repo: <https://github.com/and04671/EnergyMobile/tree/Final-Items>

1 Abstract

The goal for this study is to define and find the least energy route between a start and end address for different levels of mobility as selected by the user. This is accomplished via Python script and ArcGIS routing tools. The base data utilized for the project are a feature dataset of metro area roads, 30-meter DEM of Minnesota, and 12 LIDAR LAS blocks from MN DNR. While the prototype program effectively “solved” the problem in the initial study, it calculated energy incorrectly for a single travel mode, displaying results to a clunky UI. This version works via the same design with a number of fundamental improvements. Primarily, it relies on more reasonable energy calculations, creates a more accurate network dataset, allows user input, and utilizes reasonable restrictions. It is also designed to output a cost surface from a start location using the same information. The results will again be verified for feasibility; pedestrians must be able to traverse the path, the route should be found to be the lowest possible energy for the travel mode, and all cases must properly follow set restrictions. The results indicate a significant improvement in both form and function over the first prototype. The results produce different routes for different weighting in a way that makes sense if traced out. Though it may make more sense to add an accessibility module to an existing map platform, this project effectively builds on and displays GIS skills acquired in the graduate program.

2 Introduction and Background

The primary inspiration for this project was frustration with the amount of terrain accessibility information available to people with limited mobility on the hilly and layered University of Minnesota Twin Cities campus. This lack of information often leads wheelchair users down non-traversable routes, causing unnecessary irritation getting to events and class activities. This is not the only accessible routing blind spot at UMN; the University also does not provide disabled student access to interior building maps showing stairs, narrow doors, elevators, and general terrain.

The secondary inspiration for this project was AccessMap, a program created by Dr. Anat Caspi at University of Washington. AccessMap shows possible sidewalk routes based on slope limitations in Seattle. It is part of a larger initiative to make pedestrian travel safer and more convenient for physically disabled people. While that larger project is being piloted in a handful

of other cities, the profound lack of up-to-date municipal sidewalk information has created an impedance. The resulting objective was a Minneapolis proxy to AccessMap without sidewalk data in an ArcGIS and Python environment. This application factors slope *and* distance into a combined energy calculation. Ideally, the user can insert a start and destination, along with how much relative weight is placed on both slope and distance. In return, a router will show a visual route based on those settings.

The first prototype met the primitive criteria for these objectives. It was able to find the most efficient route based on the incorrect information it was given and returned routes to the user. However, it was not customizable and was largely numerically inaccurate. The purpose of this project extension was to add user customization, better energy calculations, and restrictions.

3 Data

Road Data: The program requires road data with lengths; the names are important for adding restrictions and hierarchy. This particular dataset is an amalgamation of road data from all Greater Metro counties. The dataset is in UTM 15 NAD83 and the horizontal units are in meters. It is updated continually but was originally published in 2020.

DEM Data: The program also requires elevation data in order to calculate slope. This 1:24,000 dataset from USGS captures elevation above sea level in 30 m resolution for the state of Minnesota. The height units are in feet while the horizontal units are meters, which will have to be standardized. It was originally published in 2004, but is continually updated.

LAS Data: Bridges and crossable man-made features not visible on the DEM require sections of high-resolution Lidar to produce accurate slope information and thus an accurate network dataset.

4 Materials and Methods

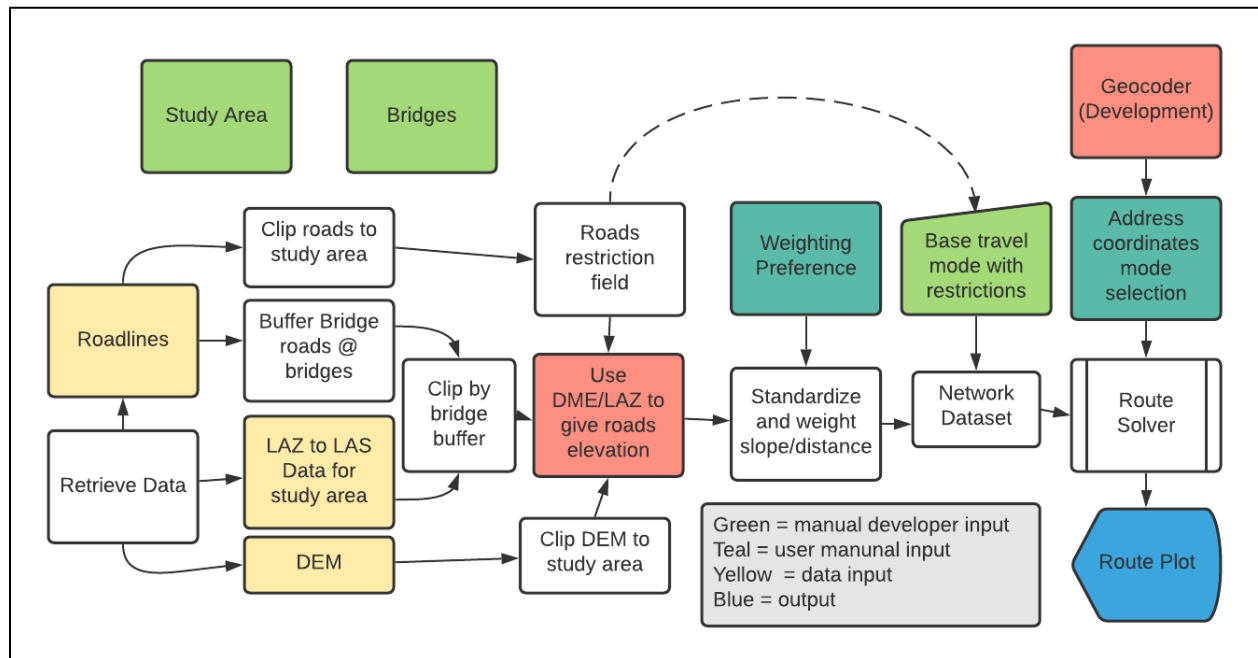


Figure 1: Spatial Workflow

Make sure python has access to the required modules for the project: arcpy, requests, json, zipfile, geopy, and utm. Use request 'get' commands and unzipping function to download the required datasets from MNGeocommons and MN-DNR and extract them to readable files.

Create a new geodatabase and feature dataset. Clip the DEM and roads file down to the study area, placing the result in the feature dataset. Of the remaining roads, select out just the ones that are bridges. Buffer these selected areas, and place the result in the feature dataset. The DEM will go into the geodatabase outside of the feature dataset

To format the LIDAR data, make sure all LAZ files are downloaded into one folder. Translate LAZ files to LAS files, and then to one raster file, using ArcPro's LAS to \diamond tool and Mosaic to Raster. Correct the raster so it is in the UTM 15 projection. Copy the raster into the geodatabase outside of the feature dataset. Use the buffered bridge roads from earlier to mask off just the needed portion of the LIDAR raster. Make sure this final raster is copied into the geodatabase outside of the feature dataset.

Next, find the surface information for the clipped down roads. Use the DEM first and then the LIDAR, use the AddSurface values tool to give roads elevation values and average slope. Only the pixels under the small LIDAR file will be reclassified from their original DEM values. Now, standardize the new average slope field and road distance fields from 1-100. The user reference ratios can be added here, but it will make more sense to add them to the network costs later

Use the clipped roads file with slope information to create a Network Dataset in ArcGIS Pro. It will appear inside the feature dataset. Create a new travel mode with 'Energy' as the impedance attribute. Energy is calculated under 'costs', and is equal to: (user value for distance)*(distance) + (user value for slope)*(slope). The user values are components of a ration, like 10:1, 1:1, etc. Since the router has to find the lowest energy route for each situation, the values between situations matters little, thus the values can add to anything instead of just 1 or 100. Add two restrictions: Avoid 35W and Avoid 94. These restrictions can be activated/deactivated from the Travel Modes menu.

Now create and define the network Route Solver. It is one automatic script in the final code. This receives a mode of travel and output path name. Create a blank point feature class and ask for user defined addresses. Translate addresses to UTM-15 coordinates and use a cursor to add them to the new blank feature class. The router inputs this points file and a temporary version of the network dataset called a network layer. 'Build' the network under the network file's properties tab. Run the router. The router will return the correct route for the situation to the feature dataset. Use the points layer to create a service area map for the parameters. Delete the temporary network layer, and the points feature class. For each change in cost ratio, edit the cost under the network properties, click ok, and then rebuild the network. Make sure to change the name of the output layer to identify the parameters.

5 Results

5.1 Right/Wrong

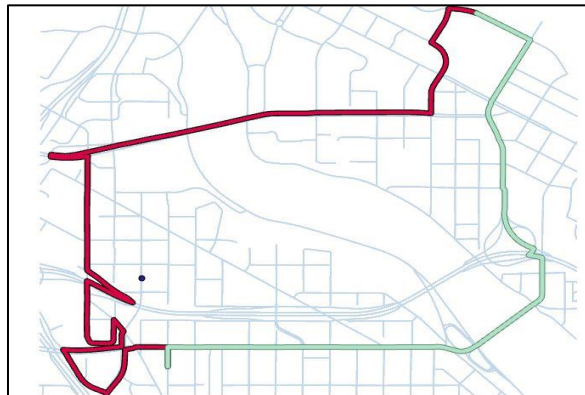



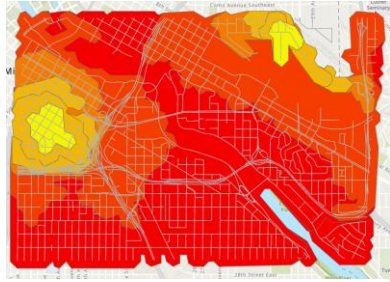
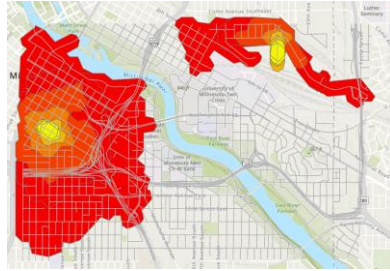
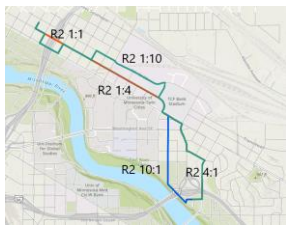
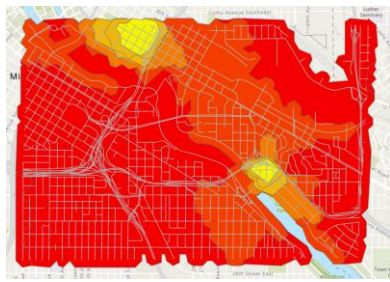
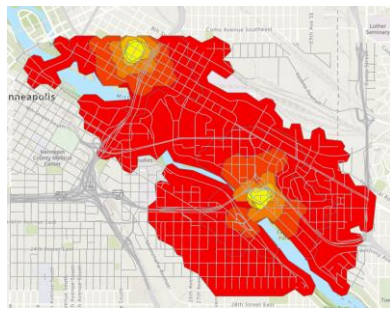

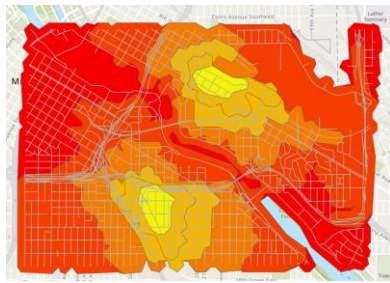


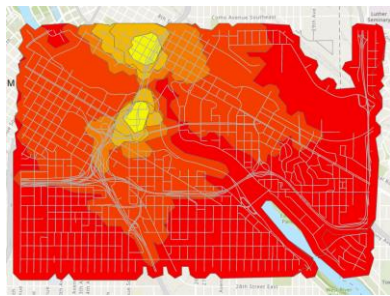

Figure 2: Equation Difference Map

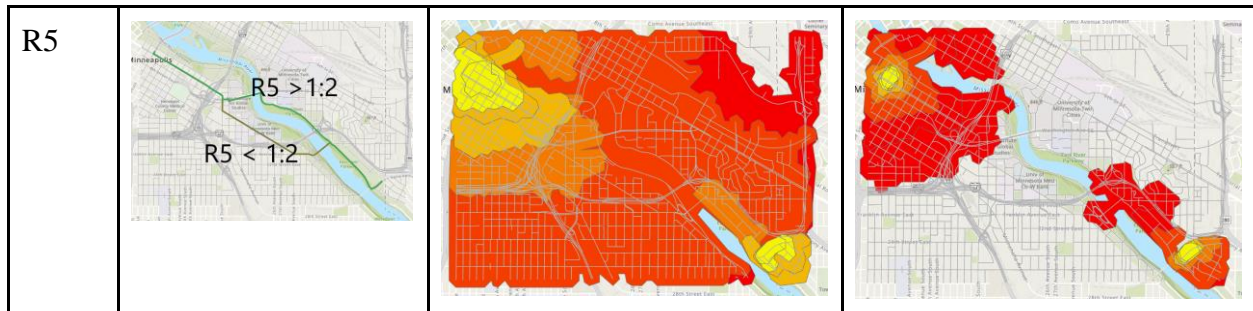
This mapping shows the effect of correct formula format on results. The correct green one is in VBScript and structured as `result = 1*[Distance]+1*[Slope]` in the code block. The red route is from using `[Distance]+[Slope]` without the constants out front, and is not correct.

Below are a number of different routes set to different parameters. The parameters are indicated by the route name. We can see that the routes do vary between parameters for the same route in a reasonable way, so the cost equation changes are working as desired. Restrictions were turned

both off and on for each route; sometimes this changed no routes, sometimes 1 or 2. Most of the routes do not have a freeway in their sensical path.

5.2 Results Table

| | Routes | Service Area 1:40 | Service Area 40:1 |
|----|---|--|---|
| R1 |  |  |  |
| R2 |  |  |  |
| R3 |  |  |  |
| R4 |  |  |  |



6 Discussion

An important trend to notice in the results is that most routes have a sort of ‘tipping point’ at a specific ratio, between two main routes. This point is not necessarily at 1:1. This means that the route is an absolute optimization of one quality or the other past a specific point. No matter how the ratio is pushed outward, the route will not change further, because it cannot.

The cost surfaces are somewhat revealing. One can get much further on smaller ratio maps than larger ones (1:40 results > 40:1 results). Those maps have the slope multiplied by many, and since this value is typically smaller (There are outliers that seem to skew this even with standardization). On the 40:1 maps, distance has much more weight, pushing the cost boundaries inward. It is hard to know if these maps can be seen as equivalent, since the constant on the distance and slope terms are not additive to 1 or 100; the actual value is arbitrary, because the router only needs to compare routes where all roads are under that condition. Both sets do add to 51 in this case, though, making them possibly absolute value comparable. As such, these maps should be used only to draw loose conclusions. Next time, a better decision would be to eliminate the odd outliers that made the slope values so much smaller, and always keep the proportions as part of the same number between preferences (ie, $\frac{1}{3}$ and $\frac{2}{3}$, $\frac{1}{4}$ and $\frac{3}{4}$, 1 and 0, etc) to allow more sensical comparison.

6.1 Challenges

There were a number of challenging factors in creating this model version. A major roadblock was how Python Arcpy handles network properties; a travel mode cannot be created via code, but only through the manual user interface. Furthermore, travel modes cannot be saved independently of their network, and therefore have to be recreated from scratch with any network deletion. This wastes a lot of time and creates a gap in an otherwise automated script. The cost equations involved in the travel mode cannot have their weighting modified via code either. On the other hand, if variable weighting is completed on the source feature class prior to building the network, this recreation has to be done with every user selection.

An accurate slope measure for this scenario continues to be evasive. Calculating through either the average slope function or the Zmax/Zmin used in the first model inevitably creates some error. Neither of those measures give the true slope of every point along the road segment. For example, a humped road has an average slope of 0, but realistically we know that hills do not

have a slope of zero at any point but the top. The problem where any zero slope nixed the energy calculation has been eliminated by changing the energy equation to an additive model.

A third challenge was merging LAS and DEM data to create an accurate terrain model with bridges. Originally, only DEM data was used, but this created scenarios where any road crossing a river bank was given a cliff-like slope; most of the bridges crossing the river in this area are entirely flat. The inclusion of LAS data meant certain DEM pixels needed to be replaced. The answer to this problem turned out to be a simple iterative surface information operation, but Mosaic to Raster options and various conditional raster operations were tested many times before that solution was found.

All these improvements having been discussed, there are still a number of issues with this model version. We have come to the realization that highway data is not a proxy for sidewalks. The slope cost for each segment needs to be calculated instead using a derivative function, but how exactly would take more research. An equation based ‘score’ using proportions is difficult to translate exactly to a level of difficulty; few people think about mobility concerns in terms of percentages. It might be better to provide the user the base information on distance and slope and allow them to evaluate which route is best, while being given different routes ideal for various criteria. The model is still a local program. It needs to be reconfigured for an online application like those in ArcGIS Online.

This table organizes the primary challenges solved or improved over the course of this second prototype:

| Problem | Thought Process | Solution |
|------------------------|--|--|
| Accurate Slope Formula | The previous slope formula used $(Z_{\max} - Z_{\min})(\text{distance})$ for each segment. Several examples prove this is inaccurate. It is my initial impression that it really should be $(Z_{\text{start}} - Z_{\text{end}})(\text{distance})$, or a derivative. The second example shows that even this formula does not factor in the ‘hump’ or ‘valley’ (So average slope also is inaccurate) | Switched over to using average slope. A better estimate but not ideal |
| Slope = 0 Issue | In the original equation, a slope of 0 gave 0 cost regardless of distance. That isn’t true. First thought is to remap the slope from 0–1 to 1–100. The remapping ratio is important, too much will make the slope too much of a factor. This might be avoidable by remapping distance to the same ratio | Original energy equation was changed to avoid this issue. Before: $\text{slope} * \text{distance}$ Now: $\text{slope} + \text{distance}$ |

| | | |
|---------------------------------------|--|---|
| Relative Difficulty of Slope/Distance | Prototype 1: Distance * slope = Energy Let the user decide this: All we care about is proportions of difficulty(from slope or distance) for the mode of travel, not the absolute value calculation of this difficulty. | Original energy equation was changed. Before: slope*distance Now: slope+distance Users weight distance and slope differently after both are standardized |
| Restrictions | Users cannot cross interstates or cliffs. | Added restriction field earlier and referenced in solver or restriction fields added to network dataset |
| Address Geocoding | The user should be able to enter a start and end address to run the router on. This address needs to be converted to coordinates for the router to use the information. Geocoding seems to be the best Arc tool for this. | Tried Nomatim geocoder and received poor performance, switched to ArcGIS Geocoding and got more accurate locations. |
| Bridges | Elevations and slopes over bridges take the riverbank vs. the bridge. | Used Lidar data over bridges and banks |
| Router Cost Failure | Router not giving different routes for vastly different parameters. The cost equation is not correct or somehow not changing between routes | Proper format is key: $x*field + y*field$ even if x or y is 1 is the correct form |

6.2 Improvements

This second prototype is a significant improvement because it simply takes variables into account. Standardizing and weighting values was the most important change, as it allows better user customization. The better bridge data prevented the code from interpreting those locations as dangerous high slope areas. The additional cost surface provides an interesting alternate way to portray the result. The route solver is much neater and automated. The addition of addresses to coordinate geolocators makes the program easier on people who do not think in terms of UTM coordinates.

Passes Least Energy Path Heuristics: The algorithm needs to deliver the least energy route for the travel mode. It is difficult to establish if this is true, because the software can test thousands

of possible routes in under a second, while it would take a very long time for a human to double check. However, there are some heuristic tests and approximations that can be used to check for a value close to the correct answer. The simplest test is a simple mental scan. To anyone who knows the mapped area well (like most UMN college students), a visual comparison to mental topography is helpful for verification. Does the route run down steep hills or river valleys, or stay on flat urban terrain? Additionally, we know that the length of the least energy route is greater than or equal to the shortest distance route of that same preference proportion. It is possible for the lowest energy and shortest route to be identical, but the above rules are not broken in any case.

Pedestrian Feasibility: This means the path is actually walkable. It must connect with walkways of the same elevation and offer the proper continuity. For example, a pathway across the UMN bridge coincides with sidewalks for most of the ways, but the output traversal at the West Bank Skyway is not possible for pedestrians without entering a university building to drop a level. Pedestrians should not ever walk off an overpass or onto a freeway. The program assumes the point moving along the path is travelling on the road, something that will have to be fixed when sidewalk data is available.

Entry and UI: The input interface needs to accept clicks or addresses and convert them to coordinates to graph on the map. The UI should use accessible colors and fonts in the output and entry prompts.

6.3 Further directions

This program is rather primitive compared to the directive at UWS. Our model misses a major point of the AccessMap project in that it does not include sidewalk data and curb cuts, critical to realistic pedestrian travel. The University of Minnesota contains a number of more efficient paths that are not considered because the model only uses roads. A more concerted effort towards standardizing sidewalk data is required for a true pedestrian model to work in most cities. Seattle is an exception, not the norm. Is reducing mobility to an equation even a good idea? There needs to be as much user input as possible; no equation can be translated directly to exhaustion. Allowing the user to make the path decision based on a neat display of more basic data might be a better option. Ie, a map showing colored segments for various slopes (like AccessMap) and a service area style option.

This semester project was primarily meant as a learning experience, and in that sense, it accomplished its goals. We became much more familiar with the concept of network GIS and the various tools offered to analyze it, especially ESRI's. Trying to find parallel tools in different models was an interesting experiment. ArcGIS Pro revealed that even its strong automated

traveling-salesman route models have annoying weaknesses. The hours of experience in writing and troubleshooting code greatly improved our Python skills.

Citations

- ESRI. (2021). *What is the network analyst module (arcpy.nax)*. ArcGIS Pro Help. Retrieved May 10, 2021, from <https://pro.arcgis.com/en/pro-app/latest/arcpy/network-analyst/what-is-the-network-analyst-module.htm>
- ESRI. (2021). *Create a network dataset*. ArcGIS Pro Help. Retrieved May 10, 2021, from <https://pro.arcgis.com/en/proapp/latest/help/analysis/networks/how-to-create-a-usable-network-dataset.htm>
- ESRI. (2021). *Route*. ArcGIS Pro Help. Retrieved May 10, 2021, from <https://pro.arcgis.com/en/pro-app/latest/arcpy/networkanalyst/route.htm>
- De Neef, M. (2013, May 29). *Gradients and cycling: How much harder are STEEPER CLIMBS?* The Climbing Cyclist. Retrieved May 10, 2021, from <http://theclimbingcyclist.com/gradients-and-cycling-how-much-harder-are-steeper-climbs>
- Isenburg, M. (2019). *Free and lossless lidar compression*. LASzip. Retrieved December 19, 2021, from <https://laszip.org/>
- N. Bolten, S. Mukherjee, V. Sipeeva, A. Tanweer and A. Caspi, "A pedestrian-centered data approach for equitable access to urban infrastructure environments," in *IBM Journal of Research and Development*, vol. 61, no. 6, pp. 10:1-10:12, 1 Nov.-Dec. 2017, doi: 10.1147/JRD.2017.2736279.
- Python Software Foundation. (2021, July). *Developer Interface*. Requests: HTTP for humans™. Retrieved December 19, 2021, from <https://docs.python-requests.org/en/latest/>
- Metropolitan Council. (2020, March 3). Road centerlines (Geospatial Advisory Council Schema). Retrieved May 10, 2021, from <https://gisdata.mn.gov/dataset/us-mn-state-metrogis-trans-road-centerlines-gac>
- U.S. Geological Survey. (2004, January 5). Minnesota digital elevation model - 30 meter resolution. Retrieved May 10, 2021, from <https://gisdata.mn.gov/dataset/elev-30m-digital-elevation-model>

TASKAR Research Center, (2016). *Accessmap*. University of Washington. Retrieved December 19, 2021, from <https://www.accessmap.io/>

The Land Management Information Center, MN Planning (2012). (Hennepin County LAZ tile data) [FTP Server]). Minnesota: MN DNR. Retrieved October 20, 2021 from <https://resources.gisdata.mn.gov/pub/data/elevation/lidar/county/hennepin/laz/>