Lab Report

Title: Lab 3 - Part 1
Notice: Dr. Bryan Runck
Author: Greg Kohler
Date: November 8th, 2023

Project Repository: *https://github.com/greg-kohler/GIS5571/tree/main/Lab3/Part 1*
Google Drive Link: N/A
Time Spent: 2

Abstract

The lab aimed to compare three different weight scenarios and how it impacted Dory's least cost path. This path goes from Dory's home to Whitewater State Park. Data included rasters for land cover and elevation. This data created a land use and slope layer for the land surrounding Dory's start and endpoints. These layers were reclassified to fit into a value of 1-5 based on her preferences. For this lab, I created three different weighting scenarios that compared the different ways the slope and land use could be weighed. The figures display the cost surface rasters reflecting cell-based costs and least cost paths. Results were verified by comparing the different least-cost paths, demonstrating the successful creation of distinct routes. The Python code's successful execution further confirmed the methodology and results. The results section and the final section of this lab compare and contrast the three different weighting scenarios.

Problem Statement

Figure 1 - Dory's Origin and Destination

The main problem for this part of the lab was to compare and contrast three different weighting scenarios for Dory to take from her house to her flyfishing location in Whitewater State Park (See Figure 1 for origin and destination). These preferences included no farmland,

gradual slope, and avoiding water if possible. This lab involved testing out different cost weights to visually compare how this could change her route.

Table 1 - Resources Used

#	Requirement	Defined As	(Spatial) Data	Attribute Data	Dataset(s)	Preparation
1	MN Geospatial Commons	Two input datasets downloaded from a URL	Elevation and Land Use Rasters	Land Use Type	Land Use Elevation	Data had to be downloaded, extracted and clipped.
2	ArcGIS Pro Notebook	Jupyter Notebooks used to store and run code	N/A	N/A	N/A	N/A
3	ArcGIS Pro Tools	Tools use to create point data	Point Data	Name of Points	Dory's Points	Had to be created
4	Google Maps	Website used to get coordinates of North Picnic Area	Coordinates	N/A	Dory's Points	N/A

Input Data

For this section of the lab, the data was obtained from the MN GeoSpatial Commons. This data was all raster data used to create the cost analysis. The land cover data was used to identify land uses, such as agricultural land or wetlands. This was essential for calculating the cost based on Dory's preferences. The other raster layer was a DEM, which was used to calculate the slope. This was essential for adhering to Dory's preference of walking along a gradual slope. The only data not from the MN GeoSpatial Commons was a point feature class I created to define Dory's origin and destination.

Table 2 - Input Data

#	Title	Purpose in Analysis	Link to Source
1	NLCD 2019 Land Cover, Minnesota	Raster dataset used to classify land use for Minnesota	Mn GeoSpatial Commons
2	Minnesota Digital Elevation Model - 30 Meter Resolution	DEM data used to create a slope layer	Mn GeoSpatial Commons
3	Dory's Points	Point data used to identify Dory's origin and destination	Created locally

Methods

The process of obtaining the data and utilizing it was the same as the previous lab (Figure 2). I started with two URLs from the Minnesota Geospatial Commons, one for Land Use and one for Elevation. For each of these, I used the requests.get(), checked the status code, and wrote the zip file to my disk. I then extracted the files from the zip file (OpenAI, 2023).

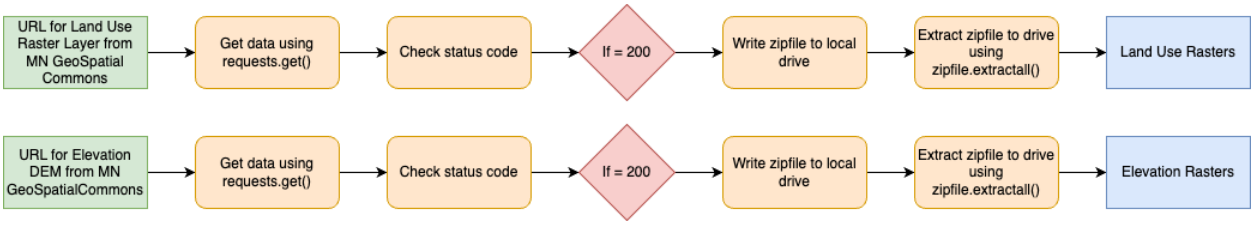

Figure 2 - Downloading and Extracing Data

After obtaining the two raster layers, I chose to extract only the parts of both layers I needed. This was done to minimize processing time (See Figure 3). I first began by creating a point feature class with Dory's house and her destination at Whitewater State Park. With this dataset, I created a buffer of 10 miles around each point. Then, using the arcpy function ExtractByMask, I clipped out the buffer area from each raster (OpenAI, 2023). This left me with two raster layers that correspond with a buffer of 10 miles around each point. Additionally, I used the arcpy Slope function to calculate the slope from the elevation layer.

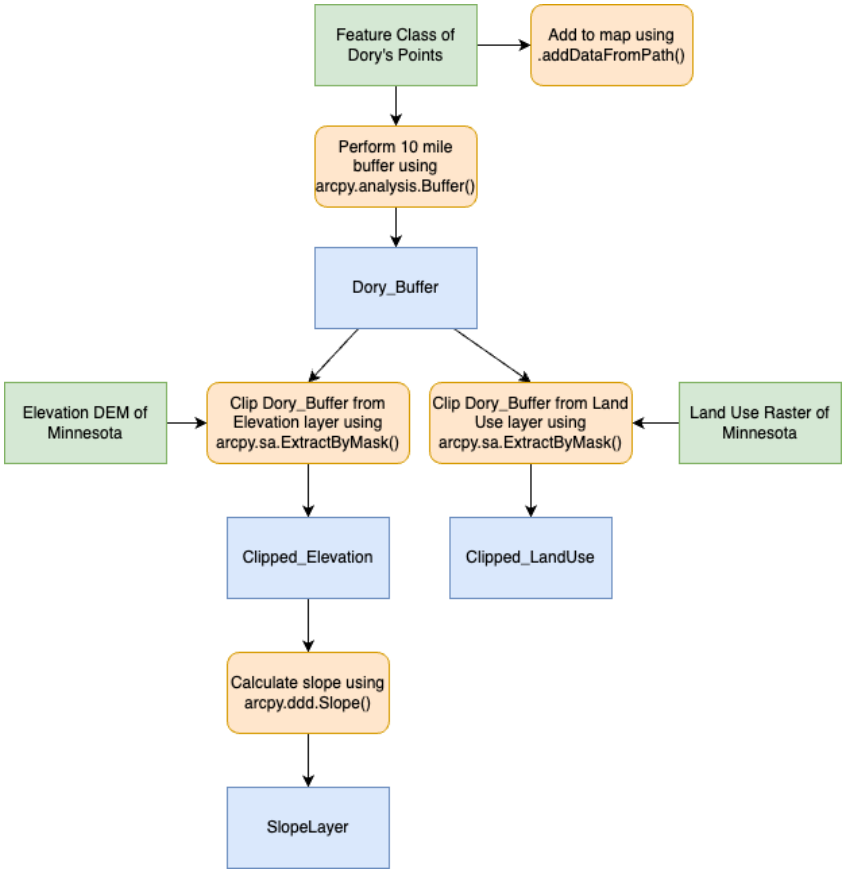

Figure 3 - Buffer and Clip from Raster

After getting the clipped rasters, the next step was to reclassify the raster values to match the chosen cost values, see Table 3 for how I chose to reclassify the layers.

Table 3 - Reclassification Values

Cost Value	Slope (Max Slope for Each)	Land Use
1	2.94	All Others
2	6.76	
3	11.76	Woody Wetlands, Emergenct Herbaceous Wetlands
4	21.17	Pasture
5	79.98	Cultivated Crops / Open Water

For slope, I had all of the slope values divided into 5 sections, using the Quantile method. This meant the more gradual slope values had a lower cost, while the steeper slopes had a higher cost. I chose a 1-5 scale as this seemed simple for use with both layers. For land use, I chose to have cultivated crops and open water be reclassified as a 5, the most costly value. This was because Dory did not want to walk through farmland and probably did not want to swim. I chose to have pasture as 4, as it would be muddy similar to farmland, but is likely preferred over normal farmland. I chose to set wetlands to 3, as Dory does not like to cross water without her waders, so I felt setting these wetlands as a middle-cost value would make the most sense. All other land uses were set to the lowest cost value, 1. I chose to do this as Dory did not specify any other land use preferences. The process of reclassifying these values was done with the arcpy function Reclassify, as seen in Figure 4 (OpenAI, 2023).

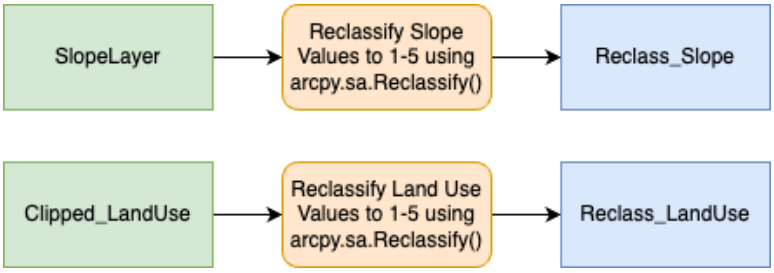

Figure 4 - Reclassifying Rasters

The previous steps were all preprocessing that was already achieved in the previous lab. The methods specific to this lab involved testing out three different weighting scenarios. See Figure 5 for a summary of this. I began with both reclassified rasters and had them in a list. This list was looped through by two different for loops. The first for loop iterated through to find the first raster in the weight equation. The second for loop iterated through to find the second raster in the weight equation. These two nested loops allow for the addition of more rasters in the future. Within these nested loops, was a third nested for loop (OpenAI, 2023). This loop iterated through a list of cost weights, in this case, 0.25 and 0.5. This nested loop also allows for the easy addition of different weights, but for this lab, I only wanted to create three different scenarios. One where Slope is 75% of the weight, one where Slope and Land Use have equal weight (50%), and one where Land Use is 75% of the weight.

As the for loops iterates through the different weights, it multiplies each raster by the appropriate weight, combines these weighted rasters with an addition equation, and then saves it to the local disk. For the weight of 0.5, I created an equation to only create one raster, so that there would not be a duplicate.

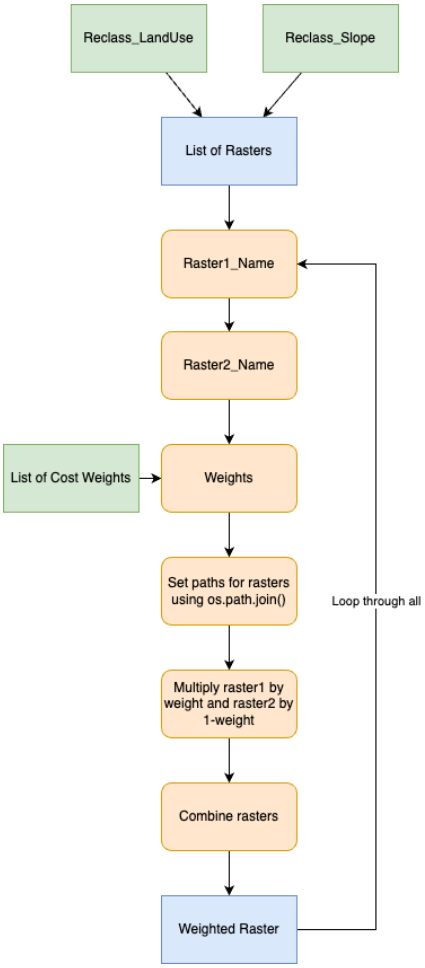

Figure 5 - Combining Rasters with Different Weights

The final process was to use the Cost Surface rasters created to create a least-cost path between Dory's House and Whitewater State Park. This was done by iterating through the list of Cost Surface rasters and using the arcpy function LeastCostPath, see Figure 6.

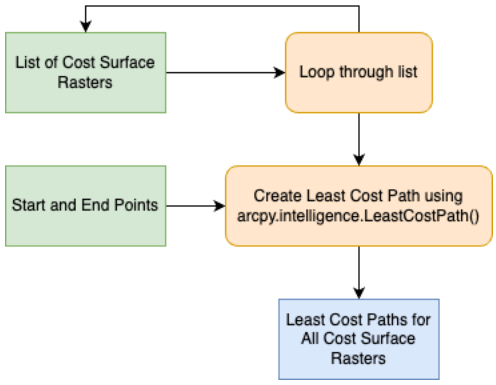

Figure 6 - Creating Least Cost Paths

Results

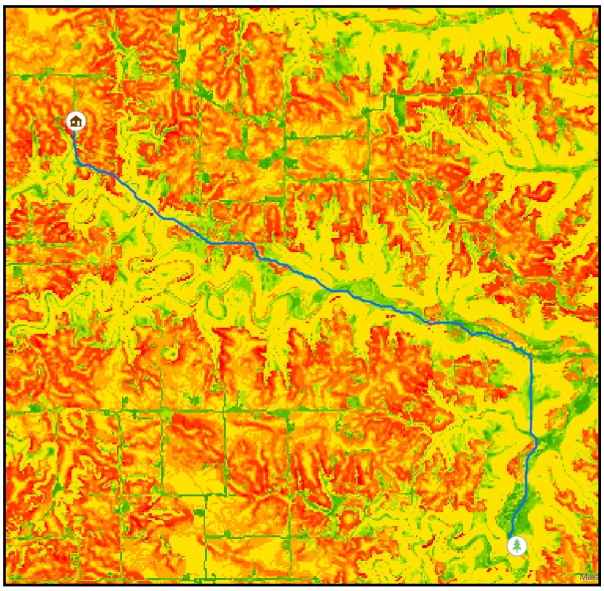

Figure 7 - Equal Weights

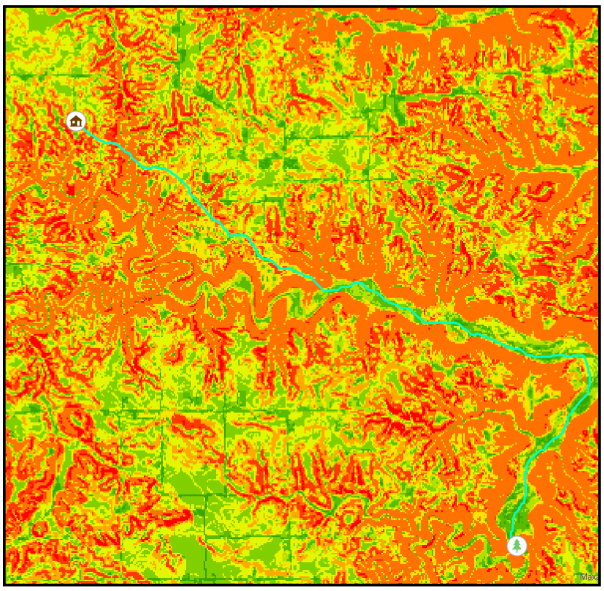

Figure 8 - Slope 75% of Weight

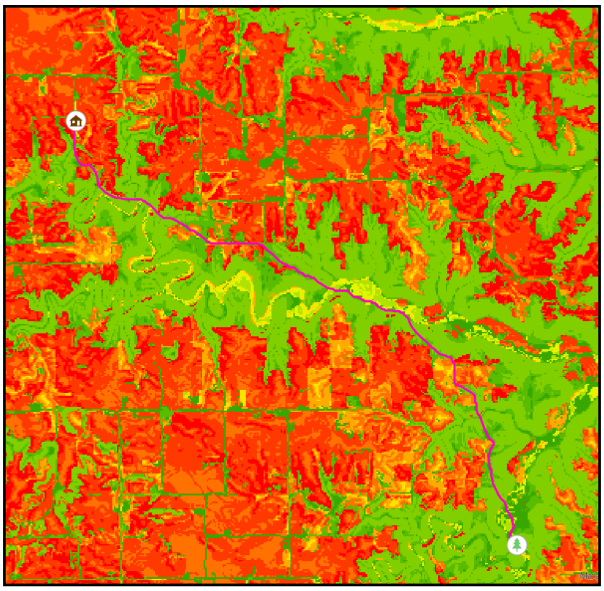

Figure 9 - Land Use 75% of Weight

For this lab, three different maps were created to represent three different weighted scenarios for Dory. These maps all show a cost surface raster, which represents the cost for each cell based on the preferences stated earlier. These maps also showcase the least cost path, which is a calculation of the least cost to take between two points. See Figure 7-9 for the cost surface raster of all three scenarios. For these maps, dark green is the lowest cost value, while dark red is the most expensive. Light greens, yellows, and oranges are middle-cost values.

To compare and contrast the three different scenarios, we can visually and numerically see how the different weights impacted Dory's least cost path. At first glance, all three of these scenarios have a path that follows the riverbed to some degree. The first scenario is where Slope and Land Use have equal importance, see Figure 7. In this scenario, Dory has no strong preference for land use or slope. It shows her cheapest route would follow along the riverbed for the majority of the time. The total length of this path is 14,984 meters and the total cost of this path is 27,628 units. The second scenario is one where the slope is 75% of the total cost weight, and land use is only 25% (See Figure 8). In this scenario, slope has a larger influence over the cost than land use. The least cost path for this scenario is the longest in distance, as Dory follows the riverbed more closely to avoid steep slopes. The total length of this path is 16,239 meters and the total cost is 28,728 units. This scenario is the longest and most "expensive." The third scenario is one where Land Use is 75% of the weight and Slope is 25% (See Figure 9). In this scenario, land use has a larger influence over the cost than the slope. Visually, this path is more similar to the equal weight scenario. This least-cost path is 13,702 meters long and the total cost is 22,428 units. It is the shortest and "cheapest" of the three weighting scenarios, as Dory cuts along areas with a higher slope instead of areas with more undesirable land use.

Results Verification

To verify our results, we can use the same method used in the previous lab, to compare the shape of our least-cost paths for all three scenarios. Looking at Figure 10, we can see that all

three paths take a different route to get to Whitewater State Park. This verifies that the code successfully created three different least cost paths based on the cost surface rasters. To further verify our results, we can see that the least cost path for the 75% weighted slope mainly stayed along flat areas, while the 75% weight land use goes over areas with steeper slopes to avoid unfavorable land uses. We can also see that the equal weight least cost path finds a compromise between the two.

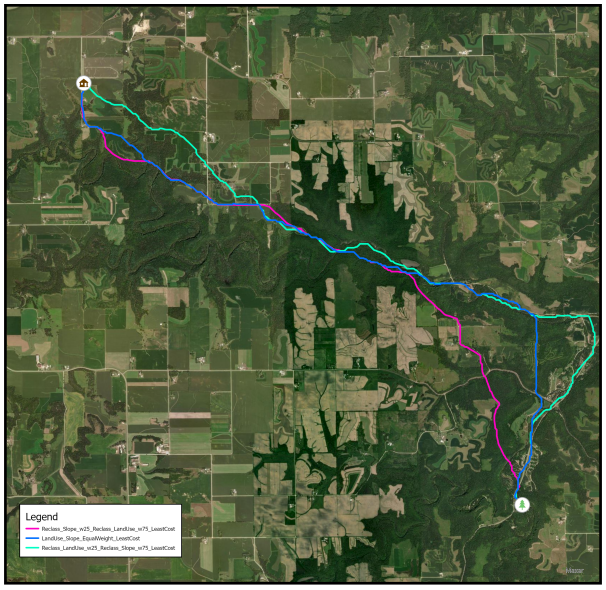

Figure 10 - All Least Cost Paths

Other than looking at the least-cost paths for all three scenarios, we can also look at our code to verify the results. Looking at Figure 11, we can see the code successfully runs and generates the three different scenarios. This successfully run code verifies that the scenarios are generated by multiplying the land use and slope rasters by a chosen weight and adding them together.

```
arcpy.env.workspace = r"\\Mac\Home\Documents\ArcGIS\Projects\Lab2-2\Lab2-2.gdb"
output_folder = r"\\Mac\Home\Documents\ArcGIS\Projects\Lab2-2\DorysPath.gdb"

# List of input raster names
input_raster_names = ["Reclass_Slope", "Reclass_LandUse"]

#List of Weights
weight_scenarios = [0.25, 0.5]

# Nested loop to process each combination of input rasters and weight scenarios
for raster1_name in input_raster_names:
    for raster2_name in input_raster_names:
        for weight in weight_scenarios:
            if weight == 0.5:
                output_name = f"LandUse_Slope_EqualWeight"
            else:
                output_name = f"{raster1_name}_w{int(weight * 100)}_{raster2_name}_w{int((1 - weight) * 100)}"
            #Skip if loop wants to pair the same rasters together
            if raster1_name == raster2_name:
                continue

            # Paths to rasters
            raster1 = os.path.join(arcpy.env.workspace, raster1_name)
            raster2 = os.path.join(arcpy.env.workspace, raster2_name)

            # Create raster combinations
            raster1_weighted = arcpy.Raster(raster1) * weight
            raster2_weighted = arcpy.Raster(raster2) * (1 - weight)
            output_raster = raster1_weighted + raster2_weighted

            # Save the output raster
            output_raster.save(os.path.join(output_folder, output_name))
            print(f"{output_name} Created and Saved")
```

```
Reclass_Slope_w25_Reclass_LandUse_w75 Created and Saved
LandUse_Slope_EqualWeight Created and Saved
Reclass_LandUse_w25_Reclass_Slope_w75 Created and Saved
```

Figure 11 - Combining Weighted Rasters Code

Discussion and Conclusion

This lab report is very similar, if not almost identical to the previous lab. Most of the preprocessing work discussed in this lab had already been created in the previous lab. The main difference in this lab was there was a greater emphasis on the three weighting scenarios and how they were similar and different. Luckily, I had already worked on creating three different weighting scenarios in the previous lab, so this section just required a deeper dive.

The scenario where land use is 75% of the weight and slope is 25% had the shortest and cheapest path. This makes sense to me. Since there are roads that cut through undesirable land use types, the least cost path can find shortcuts through these "expensive" land uses. In this scenario, the weight of the slope is also lesser, so it is cheaper to use a steeper path that shortens the overall path. In contrast, the scenario where slope has 75% of the weight and land use only has 25% is the longest and most expensive. This also makes sense to me, the land between Dory's house and the park has a steep river valley, but this river valley is flat and has optimal land uses at the bottom. Once Dory is at the bottom of the river valley, it would be "expensive" to climb out, so the path is longer and more expensive. Finally, as could be expected the scenario with equal weights finds a middle ground. It is, however, still considerably more expensive than the scenario where land use is 75% of the weight. This is probably because with equal weights, the cost of the slope, especially in the river valley, adds considerably to the total cost. Interestingly, all three of the paths chose to follow the riverbed in some way. This is likely because a straight line between the two points would require traversing both steep slopes and undesirable land choices. This deeper comparison between the weights allowed me to look at the length and total cost of each path and gave further insight into how different weights impact the cost surface.

References

ChatGPT. (2023). *ChatGPT*. Chat.openai.com; OpenAI. https://chat.openai.com/

draw.io. (n.d.). Www.drawio.com. Retrieved October 9, 2023, from http://www.drawio.com

Google Maps. (2023). *Google Maps*. Google Maps; Google. https://www.google.com/maps

Minnesota Geospatial Commons. (2023). Minnesota Geospatial Commons; State of Minnesota. https://gisdata.mn.gov

Self-score

Category	Description	Points Possible	Score
Structural Elements	All elements of a lab report are included **(2 points each)**: Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score	28	**28**
Clarity of Content	Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5 minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**.	24	**24**

Reproducibility	Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified.	28	**28**
Verification	Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**.	20	**20**
		100	**100**