Lab02 Part 2: Cost Surface Analysis

- build an ETL to obtain relevant data (DEM and Land Classification from MNGEO)
- Create Cost Surface Model
- Map the range of Cost Surfaces given 'uncertain preferences and weights' (assuming this means experiment with different weights and preferences)

Downloading data from MNGEO (DEM, Land Classification, Streams)

In [1]:
```
import requests
import json
import pprint
import zipfile
```

In [12]:
```
packages = "https://gisdata.mn.gov/api/3/action/package_list"
groups = "https://gisdata.mn.gov/api/3/action/group_list"
tags = "https://gisdata.mn.gov/api/3/action/tag_list"

#I believe this locates all the tags within the MN geo commons (I am looking for 'biota' and 'boundary')
response = requests.get(tags, auth = ('user','pass'), verify = False)
# locate by 'imagery-basemaps'
response_l = requests.get(groups, auth = ('user','pass'), verify = False)

#converting the response from unreadable bytes to json
tags_json = json.loads(response.content)

tags_json_l = json.loads(response_l.content)

#list of all the tags from MN geo commons
#pprint.pprint(tags_json_l)
```

```
C:\Users\runac\AppData\Local\ESRI\conda\envs\arcgispro-py3-Lab2_clone\lib\site-packages\urllib3\connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gisdata.mn.gov'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
C:\Users\runac\AppData\Local\ESRI\conda\envs\arcgispro-py3-Lab2_clone\lib\site-packages\urllib3\connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gisdata.mn.gov'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
{'help': 'https://gisdata.mn.gov/api/3/action/help_show?name=group_list',
 'result': ['biota',
            'boundaries',
            'climatology',
            'economy',
            'elevation',
            'environment',
            'farming',
            'geoscientific',
            'health',
            'imagery-basemaps',
            'inland-waters',
            'intelligence-military',
            'location',
            'planning-cadastre',
            'society',
            'structure',
            'transportation',
            'utilities-communication'],
 'success': True}
```

Extracting Land Cover file

In [9]:
```
#I want "Impervious Surface Area by Landsat & lidar:2013-update -version 2" data labeled with the 'land cover' tag
groups = 'imagery-basemaps'
tag1 = "impervious surface area"
tag2 = 'land cover'
tag3 = 'landsat'
tag4 = 'lidar'
tag5 = 'object based image classification'
base_url = "http://gisdata.mn.gov/api/3/action/package_search?q="

package_information_url = base_url + groups + tag1 + tag2

#requesting all information associated with 'land cover' tag from MN Geo Commons
package_information = requests.get(package_information_url, auth = ('user', 'pass'), verify = False)

#converting all the information to a json dictionary
package_dict = json.loads(package_information.content)
#pprint.pprint(package_dict)

package_dict_result = package_dict["result"]['results']

pprint.pprint(package_dict_result[0]['resources'][2])

#ID to comfirm data
#My raster ID: data-id="139066ed-25df-43f5-b012-d0efce870cbf"
```

```
C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\urllib3\connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gisdata.mn.gov'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
{'cache_last_updated': None,
 'cache_url': None,
 'created': '2020-07-09T08:14:39.151808',
 'description': '',
 'format': 'tif',
 'gdrsReaGuid': '{36088bbd-413d-48a2-9208-731e735b48e7}',
 'hash': '',
 'id': '139066ed-25df-43f5-b012-d0efce870cbf',
 'last_modified': None,
 'mimetype': None,
 'mimetype_inner': None,
 'name': 'TIFF Raster',
 'package_id': '0fc5692f-2d29-40fd-82c0-ad68a1840f1e',
 'position': 2,
 'resource_type': 'tif',
 'revision_id': '347b8423-ea25-4a57-9f7b-e97aee7e17d3',
 'size': None,
 'state': 'active',
 'url': 'https://resources.gisdata.mn.gov/pub/gdrs/data/pub/edu_umn/base_landcover_minnesota/tif_base_landcover_minnesota.zip',
 'url_type': None}
```

In [10]:
```
r = requests.get(package_dict_result[0]['resources'][2]['url'])
open('tif_base_landcover_minnesota.zip', 'wb').write(r.content)
print('extracting the content...')

#unzipping the file and saving it to my desired loxal
with zipfile.ZipFile('tif_base_landcover_minnesota.zip', 'r') as zip_ref:
    zip_ref.extractall("E:/Fall 2021/ArcGIS1/Labs/Lab02/Lab02_CostSurface/MN_geo_data_pipeline")

extracting the content...
```

Extracting DEM file

In [24]:
```
groups = 'elevation'
tag1 = 'model'
tag2 = 'slope'
tag3 = 'elevation'
base_url = "http://gisdata.mn.gov/api/3/action/package_search?q="

package_information_url = base_url + groups

#requesting all information associated with 'elevation' tag from MN Geo Commons
package_information = requests.get(package_information_url, auth = ('user', 'pass'), verify = False)

#converting all the information to a json dictionary
package_dict = json.loads(package_information.content)
#pprint.pprint(package_dict)

package_dict_result = package_dict["result"]['results']

#pprint.pprint(package_dict_result[3]['resources'][2])

#ID to confirm data
#My geodatabase ID: 1c2f17f6-f7df-43de-9d96-03b49d867f77

r = requests.get(package_dict_result[3]['resources'][1]['url'])
open('fgdb_elev_30m_digital_elevation_model.zip', 'wb').write(r.content)
print('extracting the content...')

#unzipping the file and saving it to my desired loxal
with zipfile.ZipFile('fgdb_elev_30m_digital_elevation_model.zip', 'r') as zip_ref:
    zip_ref.extractall("E:/Fall 2021/ArcGIS1/Labs/Lab02/Lab02_CostSurface/MN_geo_data_pipeline")
```

```
C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\urllib3\connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gisdata.mn.gov'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
extracting the content...
```

extracting stream data

In [35]:
```
groups = 'inland-waters'
tag1 = 'dnr fisheries'
tag2 = 'rivers'
tag3 = 'stream survey'
tag4 = 'streams'
base_url = "http://gisdata.mn.gov/api/3/action/package_search?q="

package_information_url = base_url + groups + tag1

#requesting all information associated with 'elevation' tag from MN Geo Commons
package_information = requests.get(package_information_url, auth = ('user', 'pass'), verify = False)

#converting all the information to a json dictionary
package_dict = json.loads(package_information.content)
#pprint.pprint(package_dict)

package_dict_result = package_dict["result"]['results']

#pprint.pprint(package_dict_result[7]['resources'][1])
#Shapefile ID: 0ad76fbd-452a-47b6-aa15-4a6cb49928ea

r = requests.get(package_dict_result[7]['resources'][1]['url'])
open('shp_water_measured_kittle_routes.zip', 'wb').write(r.content)
print('extracting the content...')

#unzipping the file and saving it to my desired loxal
with zipfile.ZipFile('shp_water_measured_kittle_routes.zip', 'r') as zip_ref:
    zip_ref.extractall("E:/Fall 2021/ArcGIS1/Labs/Lab02/Lab02_CostSurface/MN_geo_data_pipeline")

print('Done!')
```

```
C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3\lib\site-packages\urllib3\connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gisdata.mn.gov'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
  InsecureRequestWarning,
extracting the content...
Done!
```

Cost Surface Model

In the GUI:

- cropped data to a smaller spatial extent (previously all statewide)
- created a shapefile for two points (the start and end locations for Dory's walk)

I made to new feature classes in th GUI interface of ArcPro, one creating the start and end points for Dory's walk, the other a rectangular extent used to cropped the statewide data down to a smaller size.

In [2]:
```
!pwd
```

Out[2]:
```
'D:\\Fall 2021\\ArcGIS1\\Labs\\Lab02\\Lab02_CostSurface'
```

In [3]:
```

#Reclassifying land cover data to rescaled values between 1-10,
    #where the higher the less desirable for walking

arcpy.ddd.Reclassify("landcover_impervious_stat.tif",
                     "Value",
                     "1 100 2;101 8;102 8;103 10;104 9;105 1;106 1;107 1;108 1;109 9;110 9", 
                     r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Reclass_landcover_py1",
                     "DATA")

#in this case I have decided Dory likes walking in forested areas, roads (impervious surfaces), and managed grasslands
# AND does NOT like walking in crop fields, rivers, or mines
```

Out[3]:
Output

D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Reclass_landcover_py1

Messages

Start Time: Tuesday, October 26, 2021 6:47:28 PM
Succeeded at Tuesday, October 26, 2021 6:47:33 PM (Elapsed Time: 4.41 seconds)

In [4]:
```

#convert raster elevation values to slope values :

#Distance accumulation for slope (binary function, i decided Dory will walk anything with a slope between -10 and 10 degrees)
out_distance_accumulation_raster = arcpy.sa.DistanceAccumulation("Start",
                                                                 None,
                                                                 "Clipped_DEM.tif",
                                                                 None, "Clipped_DEM.tif",
                                                                 "BINARY 1 -10 10",
                                                                 None,
                                                                 "BINARY 1 45",
                                                                 r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Distance_Accumulation_Raster",
                                                                 None,
                                                                 None,
                                                                 None,
                                                                 None,
                                                                 '',
                                                                 "PLANAR"); out_distance_accumulation_raster.save(r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Distance_Accumulation_Raster")


#rescaling "Distance_start0" between values 1-10
#chose the NODATA because we favor not gradual slope values
out_raster = arcpy.sa.RescaleByFunction("Distance_Slope_surface_py1",
                                        "MSSMALL 1 1 # # # #",
                                        10); out_raster.save(r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Rescale_Distance_Accumulation_Raster")
```

Optimal path based only on slope

In [5]:
```
#Optimal Path (makes a path based only on slope)
arcpy.sa.OptimalPathAsLine("End",
                           "Distance_Slope_surface_py1",
                           "Out_back_Dis_surface_py1",
                           r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Optimal_Path_Slope_Only",
                           "id",
                           "EACH_ZONE")

#This path is not ideal, since it doesn't consider Dory's perference for land cover type, thus try Optimal Regions
```

Out[5]:
<geoprocessing server result object object at 0x000018ED14FF0440>

Optimal path based on both landcover and slope

In [6]:
```
#COST_SURFACE(based on the previous outcomes landcover reclassified, and slope distance accumulation) equal weighted
out_raster = arcpy.sa.WeightedSum("Rescale_Dist_slope_small VALUE 1;Reclass_landcover_py1 Value 1"); out_raster.save(r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Weighted_Sum_Slope_Landcover")
```

In [7]:
```
# using Optimal regional Connections to find an optimal path (considers the cost surface wich is based on landcover and slope)
arcpy.sa.OptimalRegionConnections("Start_End_points",
                                  r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Weighted_Sum_Slope_Landcover",
                                  None,
                                  "Cost_surface_py1",
                                  None,
                                  "PLANAR",
                                  "GENERATE_CONNECTIONS")
```

Out[7]:
<geoprocessing server result object object at 0x000018ED14FDE1E0>

Optimal path favoring landcover more than slope

In [8]:
```
#COST_SURFACE(based on the previous outcomes landcover reclassified, and slope distance accumulation) favors landcover more than slope
out_raster = arcpy.sa.WeightedSum("Rescale_Dist_slope_small VALUE 1;Reclass_landcover_py1 Value 5"); out_raster.save(r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Weighted_Sum_Slope_Landcover2")
```

In [9]:
```
# using Optimal regional Connections to find an optimal path (considers the cost surface wich is based on landcover and slope)
arcpy.sa.OptimalRegionConnections("Start_End_points",
                                  r"D:\Fall 2021\ArcGIS1\Labs\Lab02\Lab02_CostSurface\Lab02_CostSurface.gdb\Output_Weighted_Sum_Slope_Landcover2",
                                  None,
                                  "Cost_surface_py2",
                                  None,
                                  "PLANAR",
                                  "GENERATE_CONNECTIONS")
```

Out[9]:
<geoprocessing server result object object at 0x000018ED14FFDDD0>

I tried inserting streams as a barrier but the resulting path was impossible to construct. Thus, I have not considered streams as a seperate input (it is used only during the landcover preferences portions), and as a result most of my paths force Dory to cross streams *multiple* times.

In [ ]:
```

```