Alternative Fuel Station Location Optimization in the Twin Cities Metropolitan Area

Luke Zaruba
GIS 5571: ArcGIS I (Spatial Data Science I)
December 14, 2022

Abstract

With renewable energy and sustainable transportation becoming key focuses for urban planning and development in the coming decades, infrastructure for supplying sustainable alternative fuels for transportation will become a key issue, especially given the United States' reliance upon gas and oil, along with the country's aging energy system. In order to efficiently and effectively provide adequate alternative fueling infrastructure to the nation, GIS and spatial data science techniques can be used to optimize coverage and minimize costs. This project aims to use two prominent methods developed in the field of operations research, for optimizing the rollout of alternative fueling stations in the Twin Cities Metropolitan Area (TCMA) by maximizing coverage and minimizing resources. The project will show how the techniques can be used at a local or regional level, but the analysis is scalable, and can be used at much smaller scales (larger extents), like across the U.S. Interstate System.

In [1]:

```
# Import Packages
import numpy as np
import pandas as pd
import arcpy
import arcgis
import os
```

Preparation

In [2]:

```
# Dissolve Counties to Study Area
study_area_name = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\study_area"
study_area = arcpy.management.Dissolve("Counties", study_area_name, None, None, "SINGLE_PART", "DISSOLVE_LINES", '')
```

In [3]:

```
# Clip Candidate Locations to 7 County Metro Area
candidate_clip = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\candidate_locations_Clip"

candidates = arcpy.analysis.Clip("candidate_locations", study_area, candidate_clip)
```

In [4]:

```
# Create Random Points to Simulate Demand
gdb = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb"

for i in range(3):
    arcpy.management.CreateRandomPoints(gdb, f"demand_{i}", study_area, "0 0 250 250", 1000)
```

In [5]:

```
# Create Network
arcpy.na.CreateNetworkDataset(r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\Network", "NetworkFCR2021", "FunctionalClassRoads", "NO_
arcpy.na.BuildNetwork("NetworkFCR2021")
```

Out[5]:

Messages

Location Set Coverage Problem (LSCP)

In [6]:

```
# Model LSCP for a range of weights, on the three sample inputs
for i in np.arange(0.5, 2.01, 0.5):
    for d in range(3):
        # Make Model
        model = arcpy.na.MakeLocationAllocationAnalysisLayer("NetworkFCR2021", f"LSCP_{i}_{d}", "Driving", "TO_FACILITIES", "MINIMIZE_FACILITIES", 15000, 1, "POWER", i
        arcpy.na.AddLocations(model, "Facilities", candidates, "Name Name #;FacilityType # 0;Weight # 1;Capacity # #;CurbApproach # 0", "20000 Meters", None, "Function
        arcpy.na.AddLocations(model, "Demand Points", f"demand_{d}", "Name # #;Weight # 1;GroupName # #;ImpedanceTransformation # #;ImpedanceParameter # #;CurbApproach
        arcpy.na.Solve(model, "SKIP", "TERMINATE", None, '')
```

In [7]:

```
# Get Chosen Facility Counts for Each Weight/Demand Combo
for i in [0.5, 1.0, 1.5, 2.0]:
    for d in range(3):
        selName = fr"LSCP {i}\LSCP_{i}_{d}\Facilities"
        selected = arcpy.management.SelectLayerByAttribute(selName, "NEW_SELECTION", "FacilityType = 3")
        ct = arcpy.management.GetCount(selected)

        print(f"Weight {i} & Demand Dataset {d}:   {ct} facilities chosen")
```

```
Weight 0.5 & Demand Dataset 0:   35 facilities chosen
Weight 0.5 & Demand Dataset 1:   33 facilities chosen
Weight 0.5 & Demand Dataset 2:   33 facilities chosen
Weight 1.0 & Demand Dataset 0:   34 facilities chosen
Weight 1.0 & Demand Dataset 1:   34 facilities chosen
Weight 1.0 & Demand Dataset 2:   33 facilities chosen
Weight 1.5 & Demand Dataset 0:   34 facilities chosen
Weight 1.5 & Demand Dataset 1:   33 facilities chosen
Weight 1.5 & Demand Dataset 2:   34 facilities chosen
Weight 2.0 & Demand Dataset 0:   34 facilities chosen
Weight 2.0 & Demand Dataset 1:   33 facilities chosen
Weight 2.0 & Demand Dataset 2:   34 facilities chosen
```

In [8]:

```
# Create Merged Dataset of all Facilities
facility_datasets = [fr"LSCP {i}\LSCP_{i}_{d}\Facilities" for d in range(3) for i in [0.5, 1.0, 1.5, 2.0]]
output_merge_name = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\Facilities_Merged"

merged = arcpy.management.Merge(facility_datasets, output_merge_name)
```

In [9]:

```
# Select by Attribute to Eliminate Facilities not Chosen
selected = arcpy.management.SelectLayerByAttribute(merged, "NEW_SELECTION", "FacilityType = 3")

# Calculate Summary Stats to Find Counts per SourceOID
table_lscp = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\LSCP_Selected_Stats"

arcpy.analysis.Statistics(selected, table_lscp, "OBJECTID COUNT", "SourceOID")
```

Maximal Coverage Location Problem (MCLP)

In [10]:

```
# Model MCLP
for i in range(15, 26, 5):
    for d in range(3):
        # Make Model
        model = arcpy.na.MakeLocationAllocationAnalysisLayer("NetworkFCR2021", f"MCLP_{i}fac_{d}", "Driving", "TO_FACILITIES", "MAXIMIZE_COVERAGE", 20000, i, "LINEAR",
        arcpy.na.AddLocations(model, "Facilities", candidates, "Name Name #;FacilityType # 0;Weight # 1;Capacity # #;CurbApproach # 0", "20000 Meters", None, "Function
        arcpy.na.AddLocations(model, "Demand Points", f"demand_{d}", "Name # #;Weight # 1;GroupName # #;ImpedanceTransformation # #;ImpedanceParameter # #;CurbApproach
        arcpy.na.Solve(model, "SKIP", "TERMINATE", None, '')
```

In [11]:

```
# Get Chosen Facility Counts for Each Weight/Demand Combo
for i in [15, 20, 25]:
    for d in range(3):
        selName = fr"MCLP {i}\MCLP_{i}fac_{d}\Demand Points"
        selected = arcpy.management.SelectLayerByAttribute(selName, "NEW_SELECTION", "FacilityID IS NOT NULL")
        ct = arcpy.management.GetCount(selected)

        print(f"Number Facilities {i} & Demand Dataset {d}:   {ct} demand points covered out of 1000 total")
```

```
Number Facilities 15 & Demand Dataset 0:   982 demand points covered out of 1000 total
Number Facilities 15 & Demand Dataset 1:   987 demand points covered out of 1000 total
Number Facilities 15 & Demand Dataset 2:   987 demand points covered out of 1000 total
Number Facilities 20 & Demand Dataset 0:   995 demand points covered out of 1000 total
Number Facilities 20 & Demand Dataset 1:   997 demand points covered out of 1000 total
Number Facilities 20 & Demand Dataset 2:   997 demand points covered out of 1000 total
Number Facilities 25 & Demand Dataset 0:   995 demand points covered out of 1000 total
Number Facilities 25 & Demand Dataset 1:   997 demand points covered out of 1000 total
Number Facilities 25 & Demand Dataset 2:   997 demand points covered out of 1000 total
```

In [12]:

```
# Create Merged Dataset of all Facilities
facility_datasets = [fr"MCLP {i}\MCLP_{i}fac_{d}\Facilities" for d in range(3) for i in [15, 20, 25]]
output_merge_name = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\MCLP_Facilities_Merged"

merged = arcpy.management.Merge(facility_datasets, output_merge_name)
```

In [13]:

```
# Select by Attribute to Eliminate Facilities not Chosen
selected = arcpy.management.SelectLayerByAttribute(merged, "NEW_SELECTION", "FacilityType = 3")

# Calculate Summary Stats to Find Counts per SourceOID
table_mclp = r"C:\gitFiles\GIS5571\Alternative Fuel Station Optimization\data\Project_FGDB.gdb\MCLP_Selected_Stats"

arcpy.analysis.Statistics(selected, table_mclp, "OBJECTID COUNT", "SourceOID")
```