@MUSA/CPLN

SPRING 2024

MUSA/CPLN CAPSTONE

MULTIMODAL OPTIMIZATION OF CO2 TRANSPORT IN TEXAS

REPORT

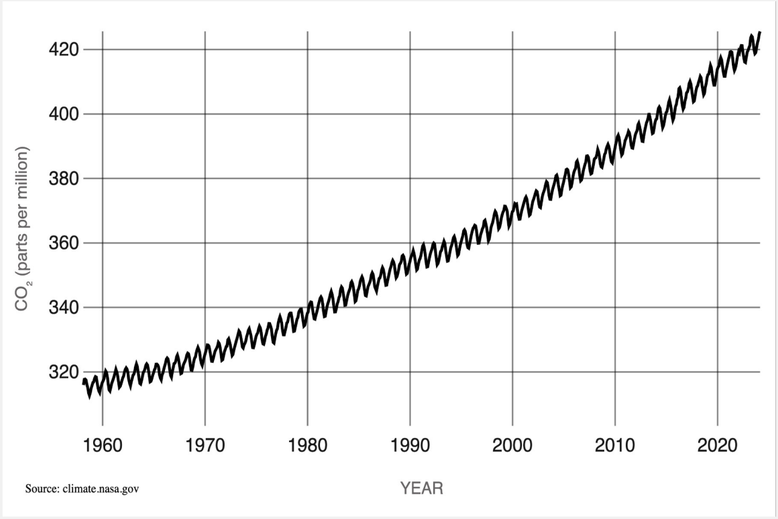
TABLE OF CONTENT

INTRODUCTIONS

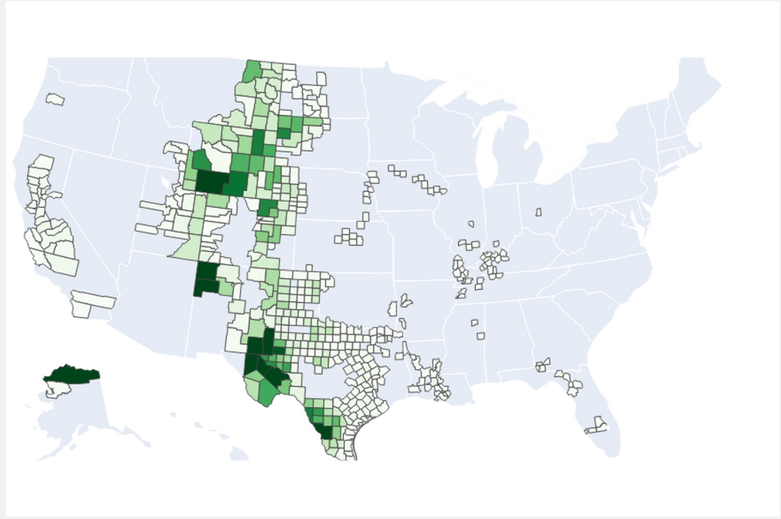
US CCS Facilities. (Congressional Budget Office, 2023).

Name of Facility	Date CCS Operations Began	Location	Type of Production	CO_2 Used for Enhanced Oil Recovery?	CO_2 Capture Capacity (Millions of metric tons per year)
Terrell	1972	Texas	Natural Gas Processing	Yes	0.5
Enid Fertiziler	1982	Oklahoma	Ammonia (Fertilizer)	Yes	0.2
Shute Creek	1986	Wyoming	Natural Gas Processing	Yes	7.0
Great Plains	2000	North Dakota	Hydrogen and Ammonia (Fertilizer)[a]	Yes	3.0
Core Energy	2003	Michigan	Natural Gas Processing	Yes	0.4
Arkalon	2009	Kansas	Ethanol	Yes	0.5
Century Plant	2010	Texas	Natural Gas Processing	Yes	5.0
Bonanza BioEnergy	2012	Kansas	Ethanol	Yes	0.1
Air Products	2013	Texas	Hydrogen	Yes	0.9
Coffeyville	2013	Kansas	Hydrogen and Ammonia (Fertilizer)[a]	Yes	0.9
Lost Cabin	2013	Wyoming	Natural Gas Processing	Yes	0.9
PCS Nitrogen	2013	Louisiana	Ammonia (Fertilizer)	Yes	0.3
Petra Nova	2017[b]	Texas	Electric Power	Yes	1.4
Illinois Industrial	2017	Illinois	Ethanol	No	1.0
Red Trail Energy	2022	North Dakota	Ethanol	No	0.2

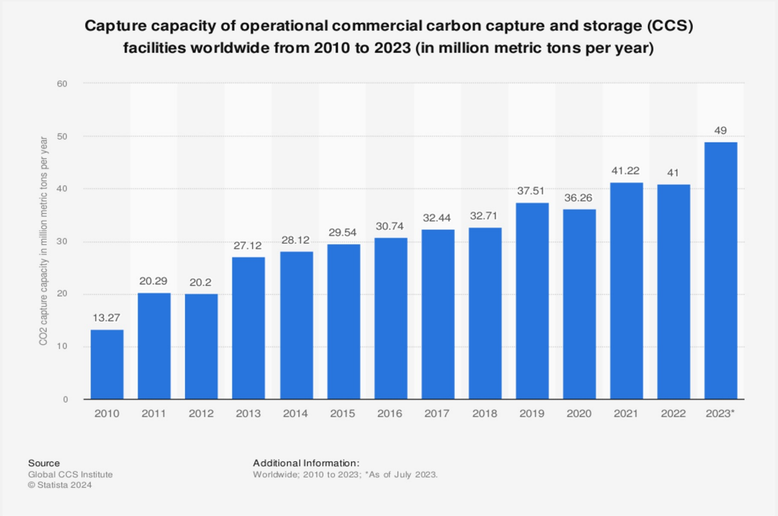
United States ranks the 1st regarding the capacity of operational carbon capture and storage facilities worldwide as of 2022, with 23.7 million metric tons per year.



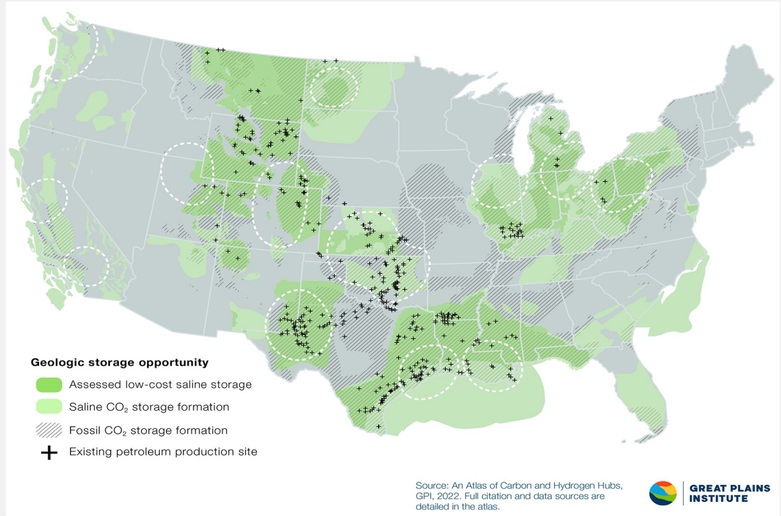
The Keeling Curve. (NASA, 2024).



Potential for Direct Air Capture. (Roads to Removal).

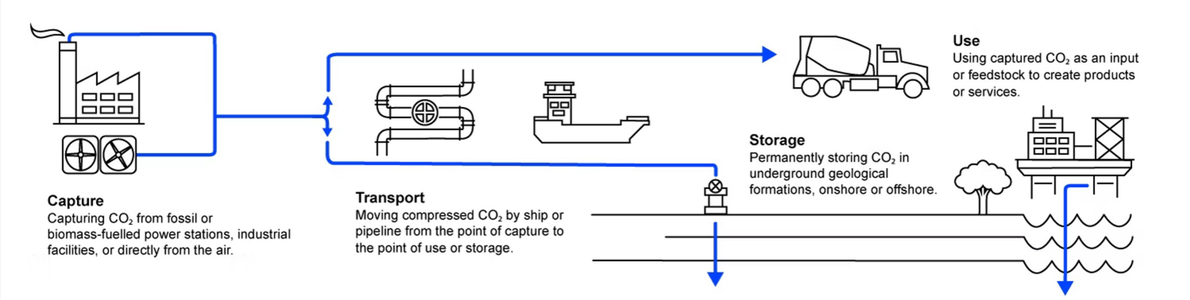


World wide CCS Capacity. (Statista, 2023).



Ample opportunities for secure geologic storage of CO2 in the US. (Great Plains Institute, 2022).

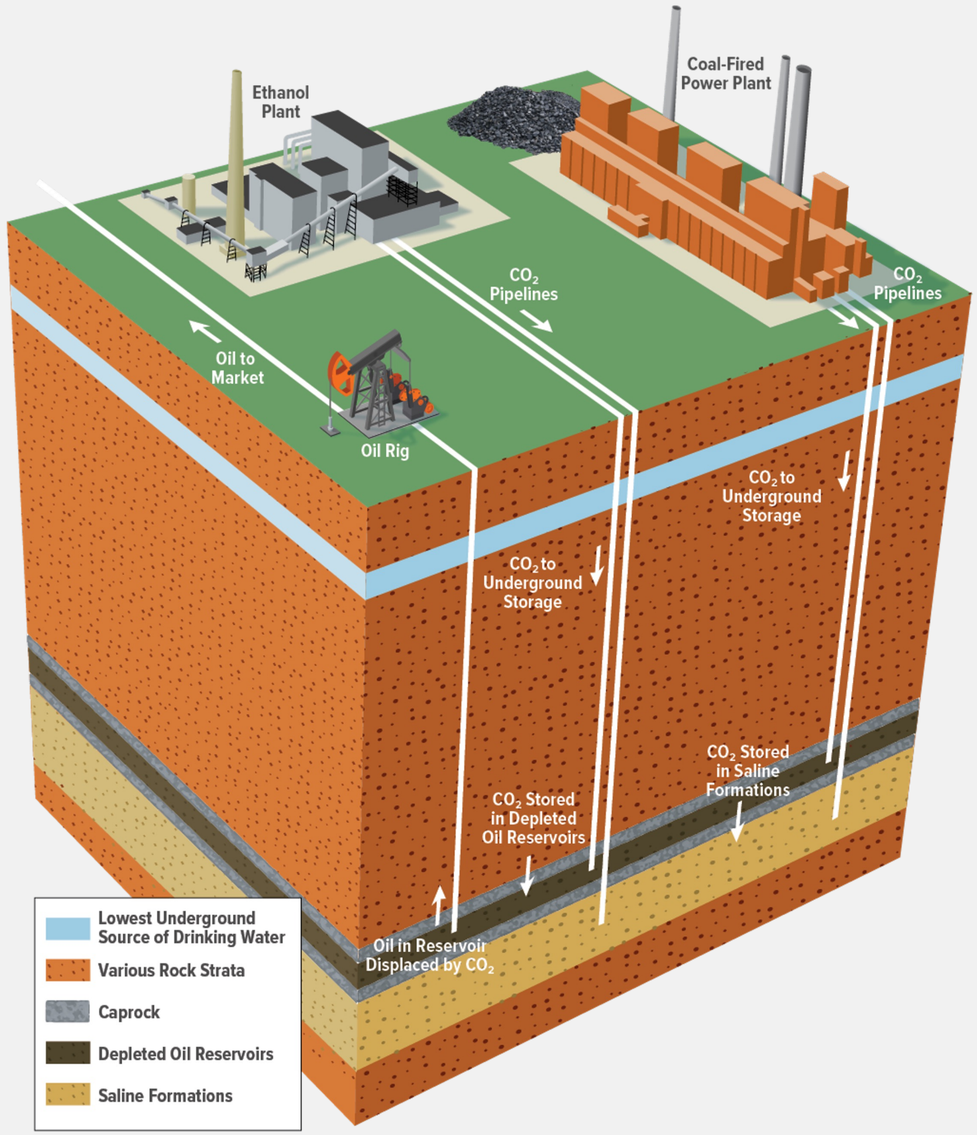
Background



CCS Process. (IEA, 2023).

United States has net-zero emission goal by 2050, 65% reduction by 2030.

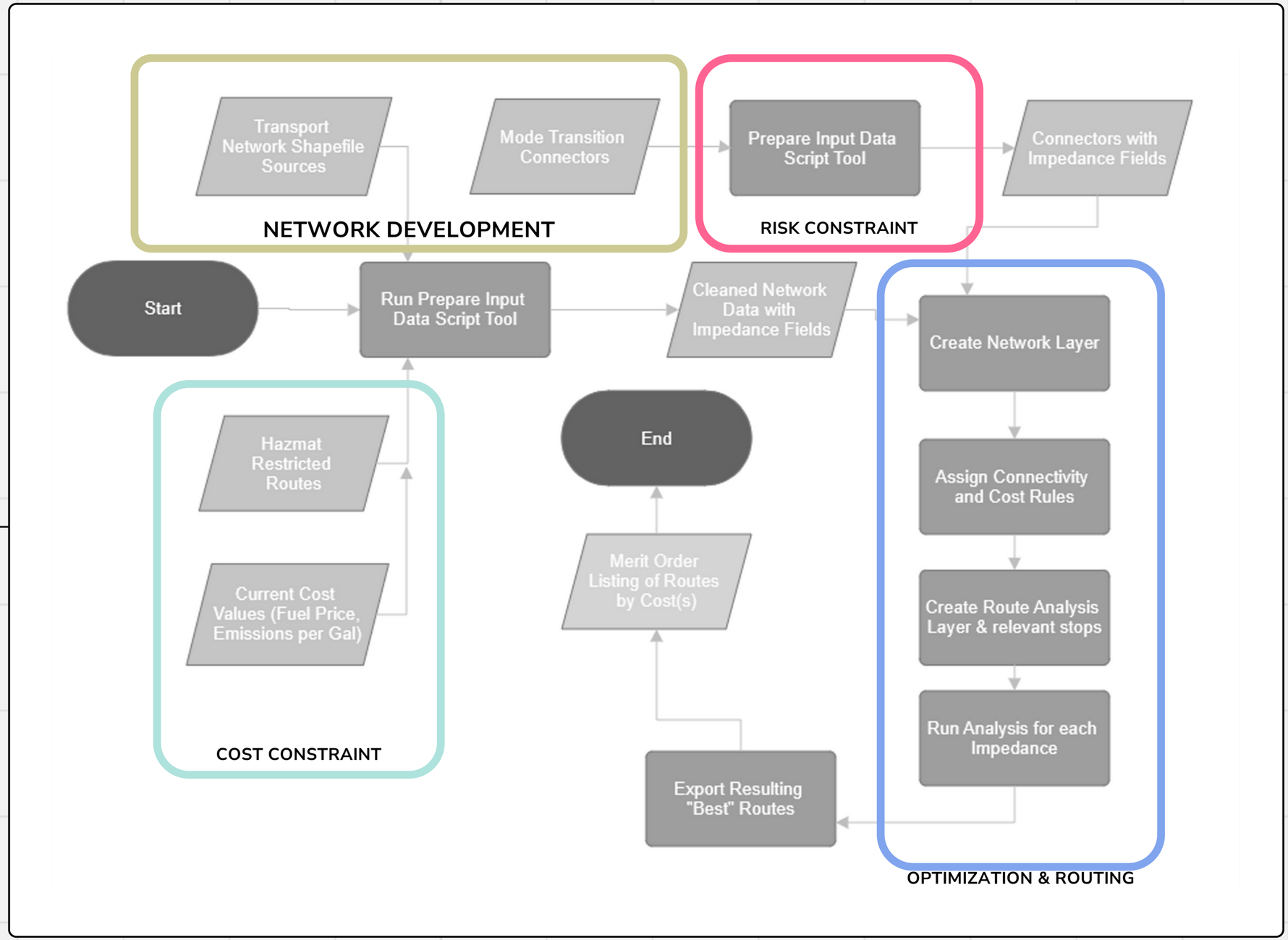
Clean Energy Conversion Lab (CECL), Kleinman Center for Energy Policy, focus on CO2 transportation cost & risk optimization, and repurposing railway and pipeline for CO2 transport.



Carbon Capture and Storage Procedure Demo. (Great Plains Institute, 2022).

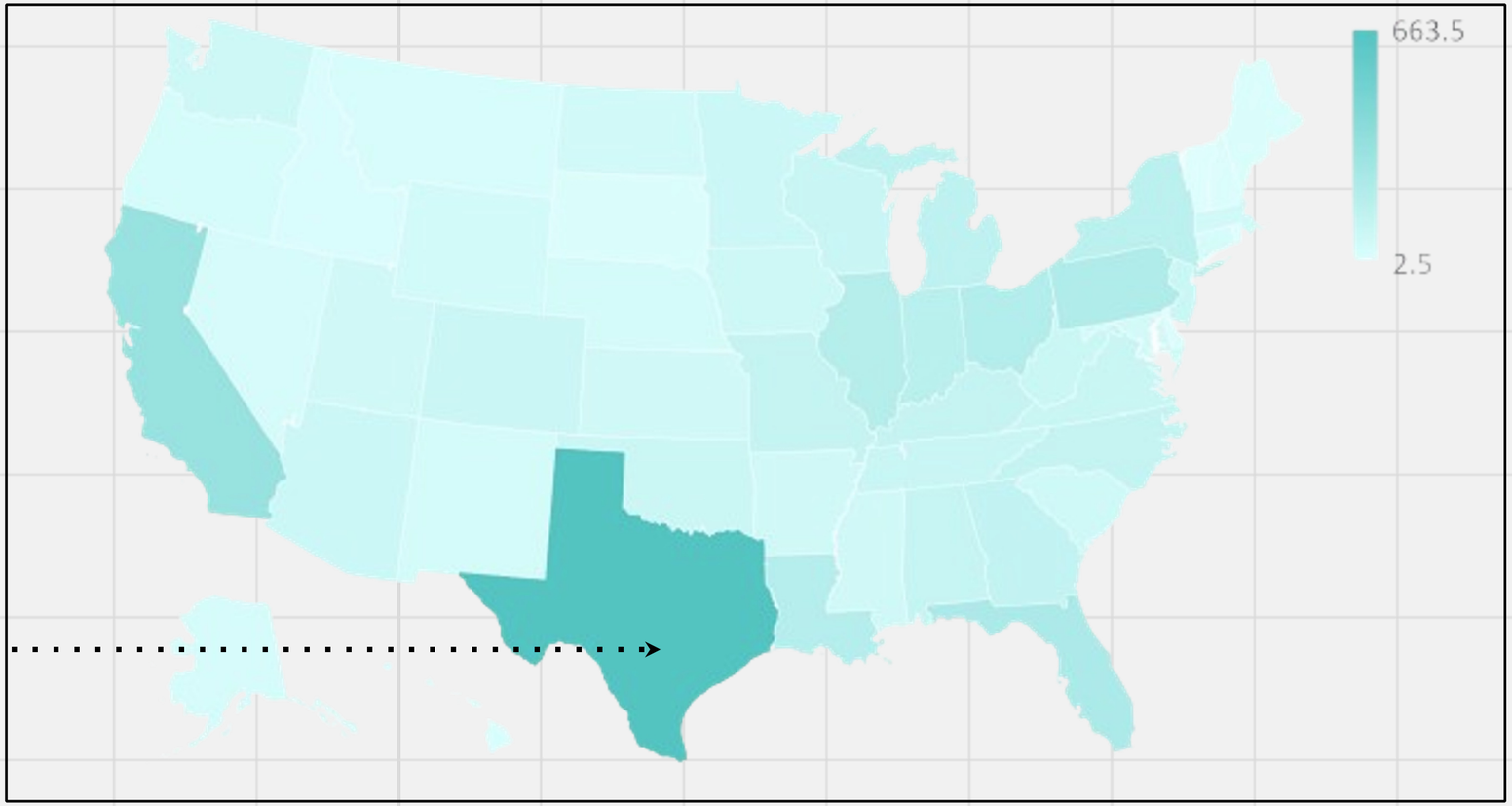
Our main workflow is to develop and calculate the risk and cost constraints of each road/link segment of our network, and optimize based on the constraints to minimize cost & risk.

METHODOLOGY



STUDY AREA

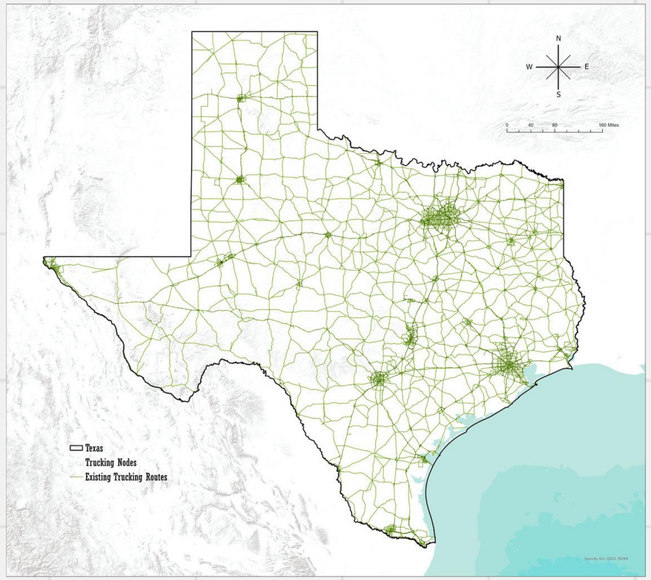
Texas is home to a diverse range of industries, including oil and gas, manufacturing, and energy production, which collectively contribute to significant CO2 emissions.



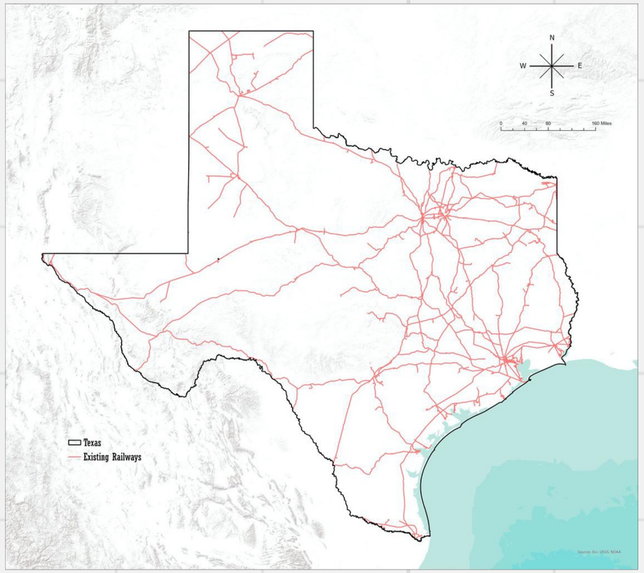
CO2 EMISSION BY STATE (TONS)

STUDY AREA TRANSPORTATION NETWORK

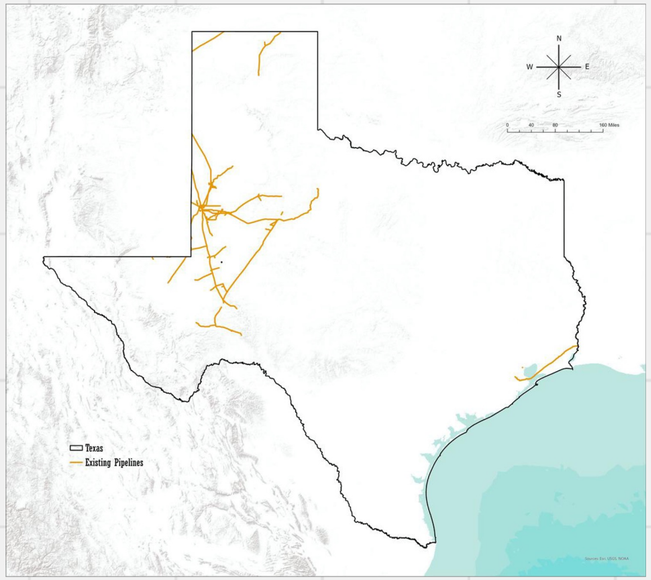
Our network model includes trucking routes, railways and pipelines



TRUCKING ROUTES

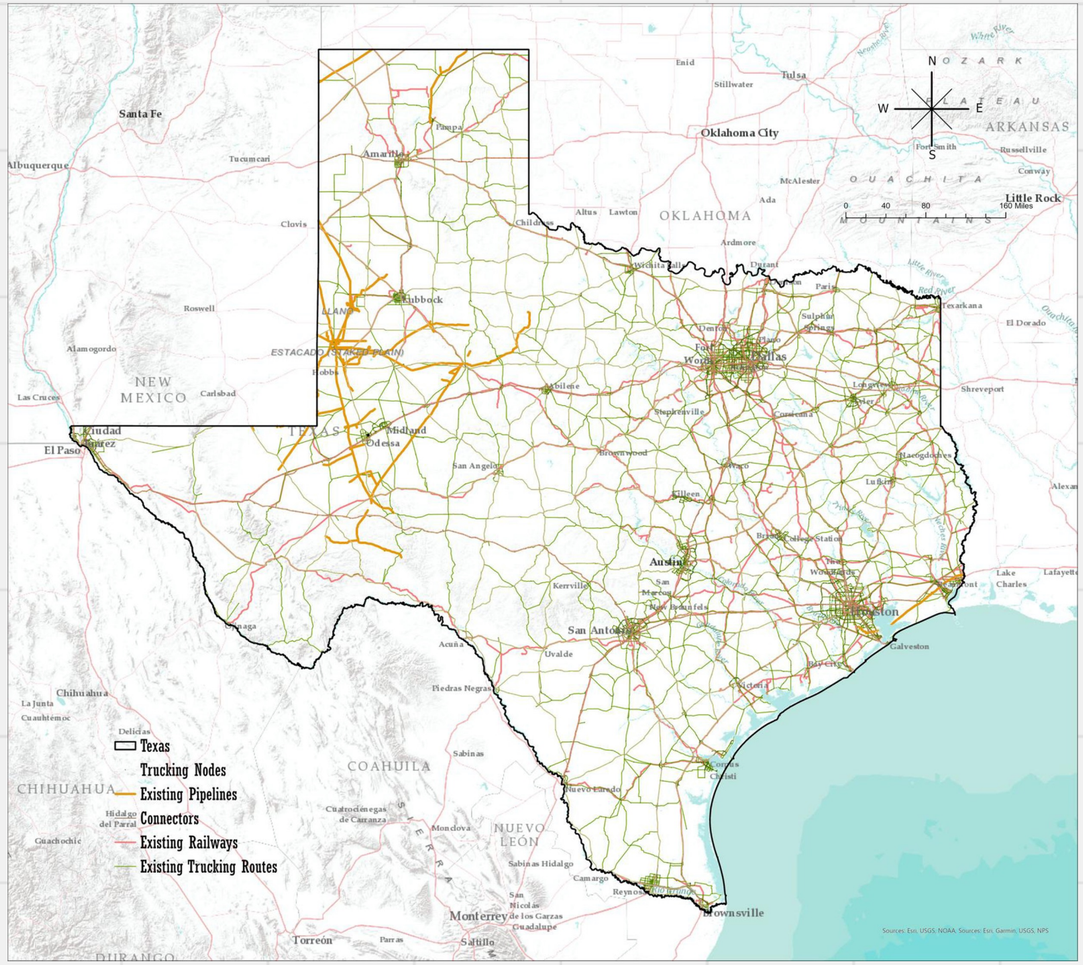


RAILWAYS



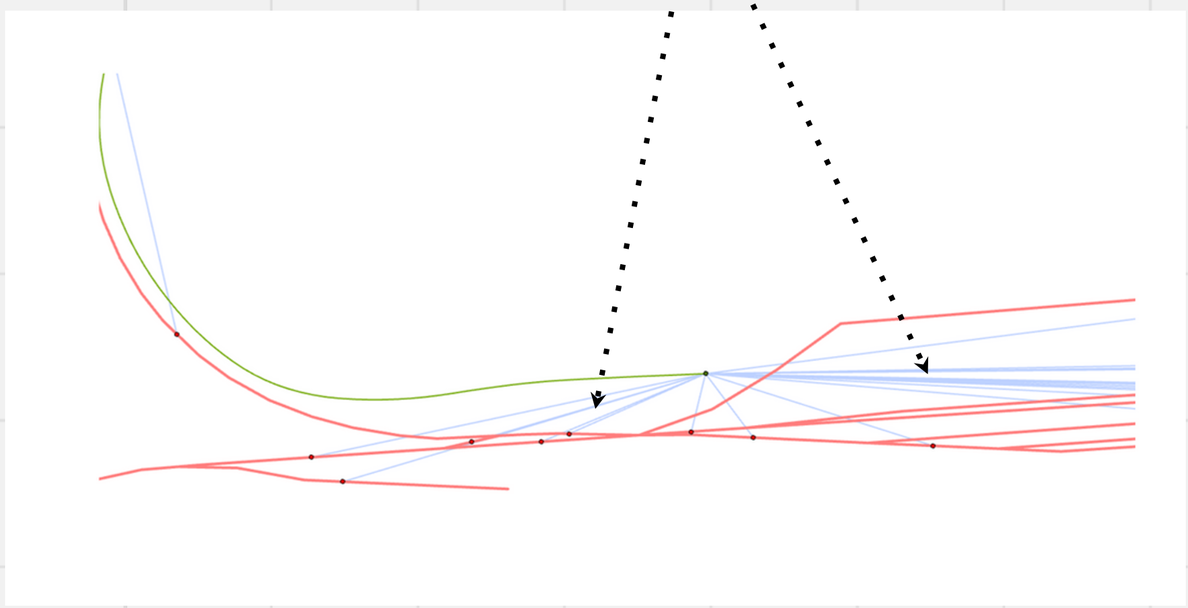
PIPELINES

@STUDY AREA



OUR MULTIMODAL TRANSPORTATION NETWORK WITH CONNECTORS BUILT

TRANSPORTATION NETWORK - CONNECTORS

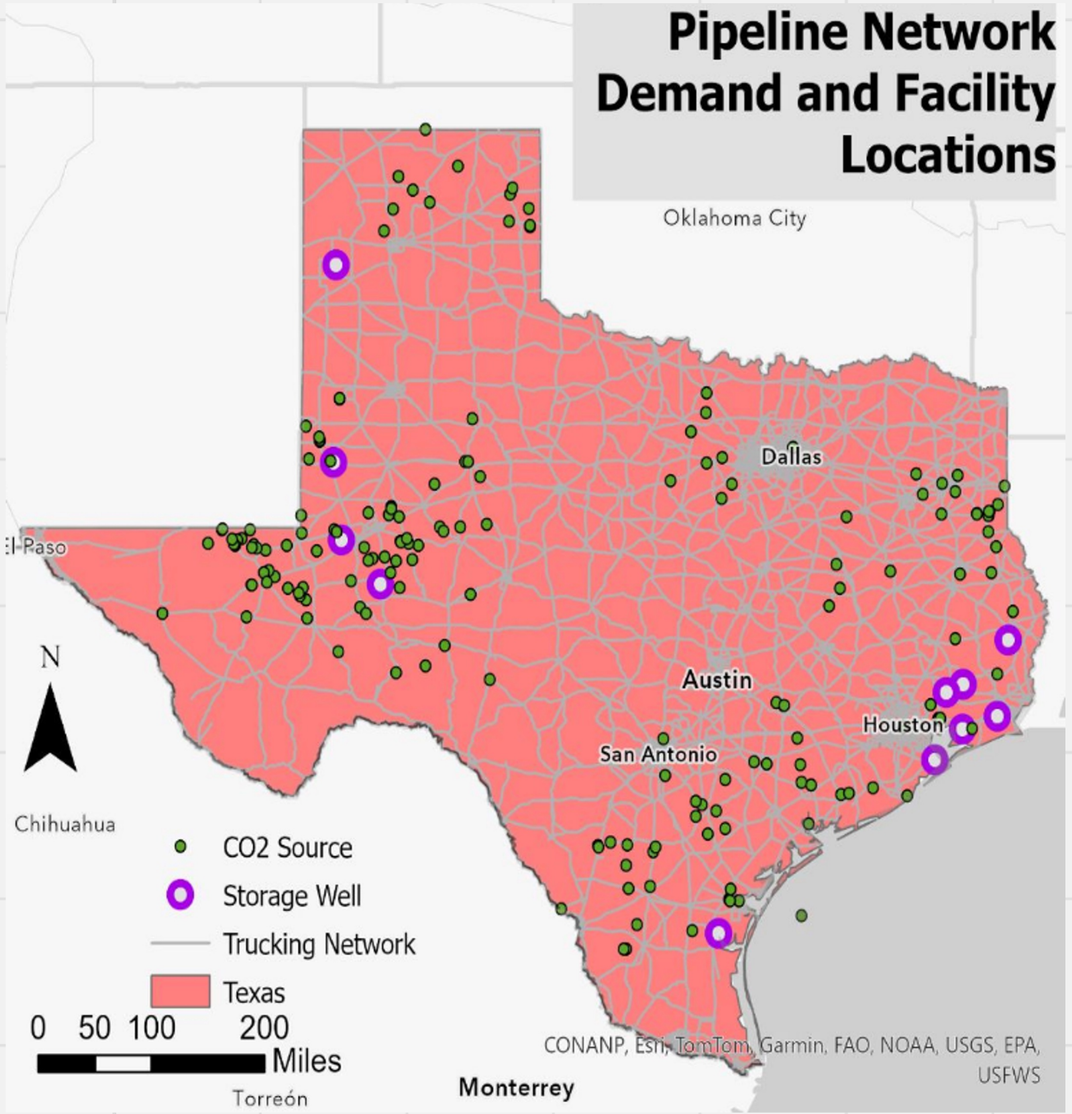


To allow transfer between rail and trucking, connectors between trucking nodes and rail yards are built using the Near Function

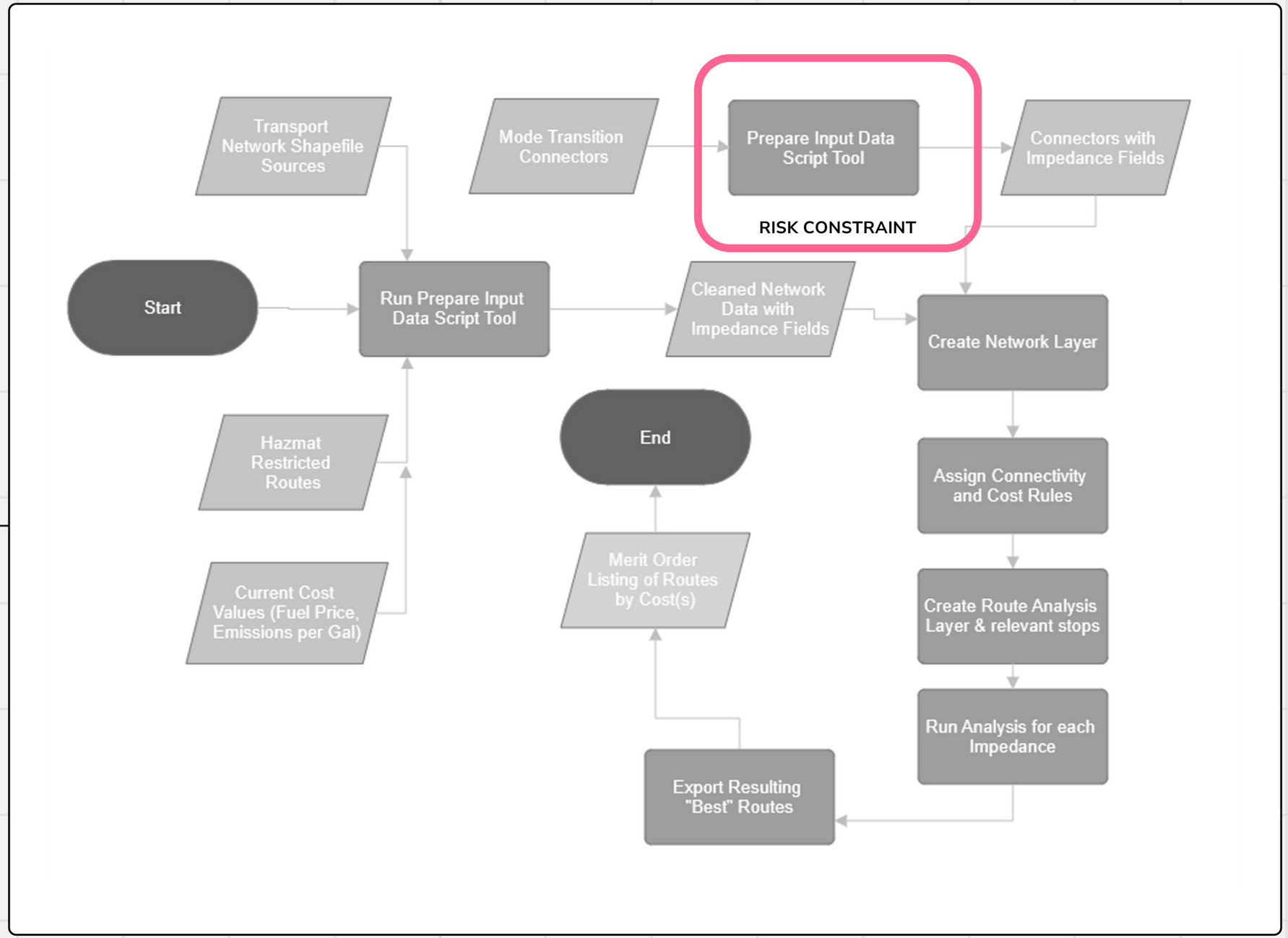
@STUDY AREA

STUDY AREA - CO2 TRANSPORT ORIGINS & DESTINATIONS

CO2 SOURCE (ORIGINS) AND CARBON STORAGE FACILITIES (DESTINATIONS) IN TEXAS



RISK CONSTRAINT



RISK MODEL FOR PIPELINES

X1Pi is the **average amount of fatalities** per incident for accident type Pi

X2Pi is the **average amount of injuries** per incident for accident type Pi

X3Pi is the **distance from the location on the pipeline with the highest probability of an incident to the closest hospital** in meters

X4Pi is the **average amount of CO2 released** per incident for pipeline type Pi

X5Pi is the **total population in a radius of 5 km of the pipeline**

X6Pi is the total number of kilometers **a pipeline traverses vulnerable land**

MODEL

$$
\begin{aligned}
Risk_P &= \sum_{j=1}^{2} Frequency_{P_j} \cdot Severity_{P_j} \\
&= \sum_{i=1}^{2} \sum_{k=1}^{7} f_{P_i} w_{k_{P_i}} X_{k_{P_i}} \\
&= (f_{P_1} w_{1_{P_1}} X_{1_{P_1}} + \ldots + f_{P_1} w_{6_{P_1}} X_{6_{P_1}}) + (f_{P_2} w_{1_{P_2}} X_{1_{P_2}} + \ldots + f_{P_2} w_{6_{P_2}} X_{6_{P_2}}).
\end{aligned}
$$

RISK MODEL FOR MULTIMODAL-APPROVED ROUTES

X1 is the **average amount of fatalities** per incident for a road

X2 is the **average amount of injuries** per incident for a road

X3 is the **distance from the road to the closest hospital** in meters

X4 is the **average amount of CO2 released** per incident

X5 is the **average population density** in a radius of 1 km of the road segment

X6 is the total number of kilometers **a road traverses vulnerable land**.

MODEL

$$\begin{aligned} Risk_T &= Frequency_T \cdot Severity_T \\ &= f\vec{x}\vec{W} \\ &= fw_1X_1 + \ldots, fw_6X_6. \end{aligned}$$

Tool 1 Calculate Total Number of Population Within 5 Kilometer of Pipeline Segments

Input:

Pipeline Shapefile

Population Raster

Output:

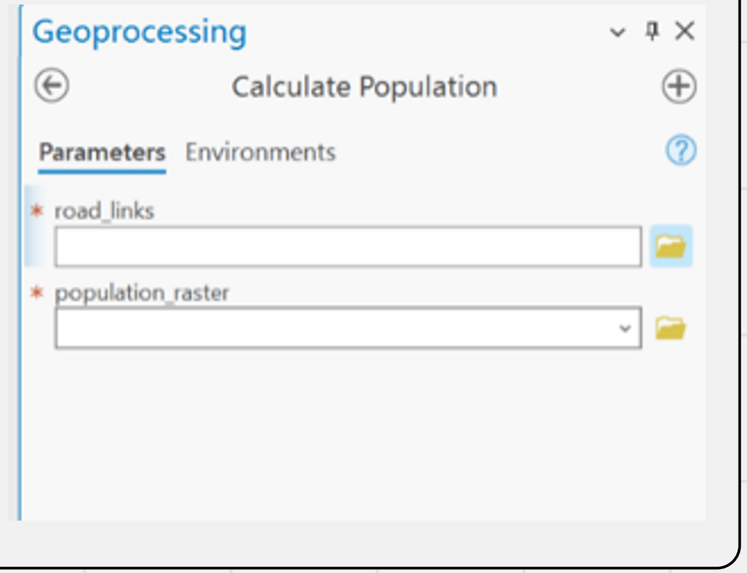
Updated pipeline Shapefile with fields holding:

1. the total number of population within 5 kilometer of the pipelines
2. The 5-kilometer radius buffered area

PYTHON SCRIPT TOOL

```
TRY:
  ROAD_LINKS = ARCPY.GETPARAMETERASTEXT(0)
  POPULATION_RASTER = ARCPY.GETPARAMETERASTEXT(1)
  BUFFER_DISTANCE = "5000 METERS"

  BUFFER_OUTPUT = "IN_MEMORY\\BUFFERED_ROADS"
  ARCPY.BUFFER_ANALYSIS(ROAD_LINKS, BUFFER_OUTPUT, BUFFER_DISTANCE)

  ARCPY.ADDFIELD_MANAGEMENT(BUFFER_OUTPUT, "AREA", "DOUBLE")

  ARCPY.CALCULATEGEOMETRYATTRIBUTES_MANAGEMENT(
    BUFFER_OUTPUT,
    [["AREA", "AREA_GEODESIC"]],
    "SQUARE_METERS"
  )

  CLIPPED_RASTER = ARCPY.SA.EXTRACTBYMASK(POPULATION_RASTER, BUFFER_OUTPUT)

  ZONAL_STATS_TABLE = "IN_MEMORY\\ZONAL_STATS"
  ARCPY.SA.ZONALSTATISTICSASTABLE(BUFFER_OUTPUT, "OBJECTID", CLIPPED_RASTER,
ZONAL_STATS_TABLE, "DATA", "SUM")

  ARCPY.JOINFIELD_MANAGEMENT(BUFFER_OUTPUT, "OBJECTID", ZONAL_STATS_TABLE,
"OBJECTID", ["SUM"])

  ARCPY.ADDFIELD_MANAGEMENT(ROAD_LINKS, "POPULATION_DENSITY", "DOUBLE")

  ARCPY.JOINFIELD_MANAGEMENT(ROAD_LINKS, "OBJECTID", BUFFER_OUTPUT,
"OBJECTID", ["AREA", "SUM"])

  WITH ARCPY.DA.UPDATECURSOR(ROAD_LINKS, ["POPULATION_DENSITY", "SUM",
"AREA"]) AS CURSOR:
    FOR ROW IN CURSOR:
      POPULATION_SUM = ROW[1]
      BUFFER_AREA = ROW[2]
      POPULATION_DENSITY = POPULATION_SUM / BUFFER_AREA IF BUFFER_AREA > 0
ELSE 0
      ROW[0] = POPULATION_DENSITY
      CURSOR.UPDATEROW(ROW)

EXCEPT EXCEPTION AS E:
  ARCPY.ADDERROR(STR(E))
```

HOW THE TOOL LOOK LIKE IN GIS



Buffer_Area	SUM_12_13_14
0.014251	21.947788
0.036445	6349.369962
0.026829	1449.280787
0.037416	1960.007376
0.027525	1927.558027
0.016401	94.027557

OUTPUT

Tool 2 Calculate Total Number of Meters a Pipeline Segment Traverses Vulnerable Land

Input:

Pipeline Shapefile

Vulnerable Area Shapefile

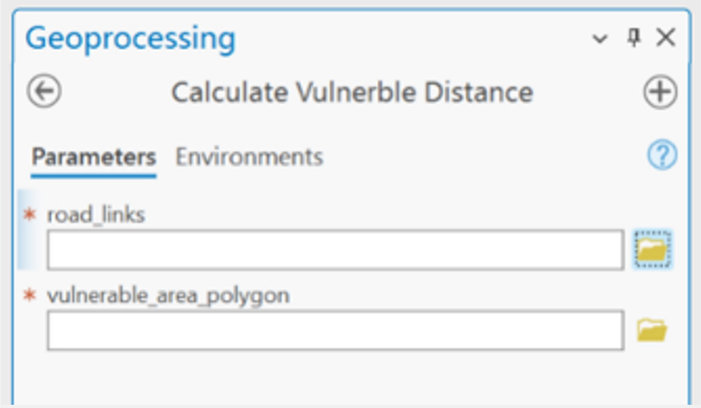
Output:

Updated pipeline shapefile with fields holding the total number of meters a pipeline segment traverses vulnerable land

PYTHON SCRIPT TOOL

```
IMPORT ARCPY

TRY:
  ARCPY.ENV.OVERWRITEOUTPUT = TRUE

  # INPUTS OF TRUCKS, RAILROADS, PIPES
  ROAD_LINKS = ARCPY.GETPARAMETERASTEXT(0)
  VULNERABLE_AREA_POLYGON = ARCPY.GETPARAMETERASTEXT(1)

  VULNERABLE_DISTANCE = "IN_MEMORY/VULNERABLE_DISTANCE"
  ARCPY.INTERSECT_ANALYSIS([ROAD_LINKS,
VULNERABLE_AREA_POLYGON],   VULNERABLE_DISTANCE,   "ALL",   "",
"LINE")

  ARCPY.ADDFIELD_MANAGEMENT(VULNERABLE_DISTANCE,
"SEGMENT_LENGTH", "DOUBLE")

  WITH  ARCPY.DA.UPDATECURSOR(VULNERABLE_DISTANCE,  ["SHAPE@",
"SEGMENT_LENGTH"]) AS CURSOR:
    FOR ROW IN CURSOR:
      ROW[1] = ROW[0].LENGTH
      CURSOR.UPDATEROW(ROW)

  ARCPY.JOINFIELD_MANAGEMENT(ROAD_LINKS,           "OBJECTID",
VULNERABLE_DISTANCE, "OBJECTID", ["SEGMENT_LENGTH"])

  ARCPY.DELETE_MANAGEMENT(VULNERABLE_DISTANCE)

EXCEPT EXCEPTION AS E:
  ARCPY.ADDERROR(STR(E))
```

HOW THE TOOL LOOK LIKE IN GIS



OUTPUT

Segment_Length
27404.400234
24038.755806
23310.475617
22897.028975
20108.874094
16131.885429

Tool 3 Calculate Average Precipitation of the Area where the Segment is Located

Input:

Pipeline Shapefile

Precipitation Shapefile

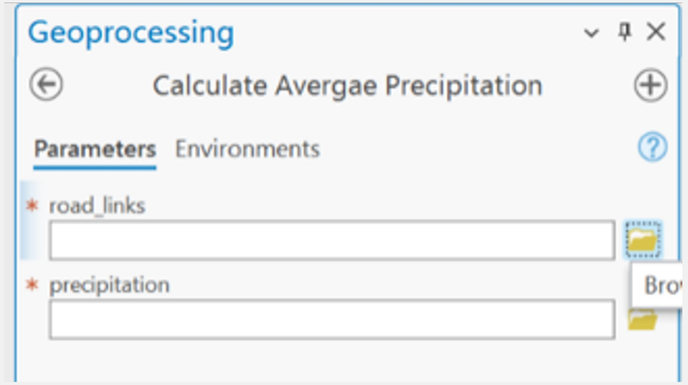
Output:

Updated pipeline shapefile with a field holding the average precipitation of the area where the pipeline segment is located

PYTHON SCRIPT TOOL

```
IMPORT ARCPY
try:
  arcpy.env.overwriteOutput = True

  road_links = arcpy.GetParameterAsText(0)
  precipitation = arcpy.GetParameterAsText(1)

  precipitation_value = "in_memory/precipitation_value"
  arcpy.analysis.Identity(road_links,   precipitation,   precipitation_value,
"ALL", "", "1")

  arcpy.AddField_management(road_links, "precip_avg", "DOUBLE")

  # Store total precipitation and count of overlaps for each road link
  precipitation_totals = {}
  with       arcpy.da.SearchCursor(precipitation_value,       ["FID_Natura",
"LEFT_PrecipInch"]) as cursor:
    for row in cursor:
      road_link_id = row[0]
      precip_value = row[1]
      if road_link_id not in precipitation_totals:
        precipitation_totals[road_link_id] = {'total': 0, 'count': 0}
      precipitation_totals[road_link_id]['total'] += precip_value
      precipitation_totals[road_link_id]['count'] += 1

  with  arcpy.da.UpdateCursor(road_links,  ["FID_Natura",  "precip_avg"])  as
cursor:
    for row in cursor:
      road_link_id = row[0]
      if road_link_id in precipitation_totals:
        total_precip = precipitation_totals[road_link_id]['total']
        count = precipitation_totals[road_link_id]['count']
        row[1] = total_precip / count if count != 0 else 0
      else:
        row[1] = 0
      cursor.updateRow(row)

  arcpy.Delete_management(precipitation_value)

except Exception as e:
  arcpy.AddError(str(e))
```

HOW THE TOOL LOOK LIKE IN GIS



precip_avg
57
56.5
59
12
24
34.333333

OUTPUT

Tool 4 Calculate Distance from the Pipeline Segment to Nearest Hospital

Input:

Pipeline/Link Shapefile

Hospital Points Shapefile

Unit of Measure

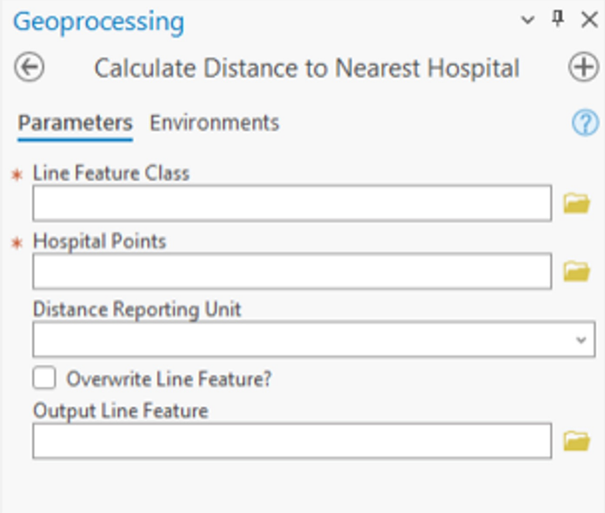
Output:

Updated pipeline shapefile with a field holding calculated distance in the desired units

PYTHON SCRIPT TOOL

```
IMPORT ARCPY
import os
arcpy.env.overwriteOutput = True
DEBUG = True

def getHospDist(lineFC,hosp,dist_unit,overwrite,outpath):

    # set temporary field/feature class locations
    if overwrite:
        finalFC = lineFC
    else:
        finalFC = arcpy.management.CopyFeatures(lineFC,outpath)

    if DEBUG:
        arcpy.AddMessage(f"Distance unit:\n\t{dist_unit}")

    # get near feature coordinates & set field names (saved in pointCpy)
    if dist_unit == "":
        arcpy.analysis.Near(finalFC, hosp,
                            field_names = [["NEAR_DIST","HOSP_DIST"]])
    else:
        arcpy.analysis.Near(finalFC, hosp,
                            location = False,
                            distance_unit = dist_unit,
                            field_names = [["NEAR_DIST","HOSP_DIST"]]
)

    if DEBUG:
        for f in arcpy.ListFields(finalFC):
            arcpy.AddMessage(f.name)
    # rename NEAR_DIST field to HOSP_DIST
    arcpy.management.DeleteField(finalFC,["NEAR_DIST","NEAR_FID"])
    ## might want to add average of length from (1) nearest point,
    ##  (2) closest end of line segment,
    ##  & (3) furthest end of line segment
    ##  for more accurate number

    # clean up arcpy objects & python variables (if necessary


    return f"Added values for distance to nearest hospital from lines."
```

HOW THE TOOL LOOK LIKE IN GIS



OUTPUT

HOSP_DIST
0.39163
0.389357
0.33563
0.411413
0.222993
0.38857

Tool 5 Calculate the Number of Historic Accidents and Fatalities on a Road Link

Input:

Route/Link Shapefile

Accidents Point Shapefile

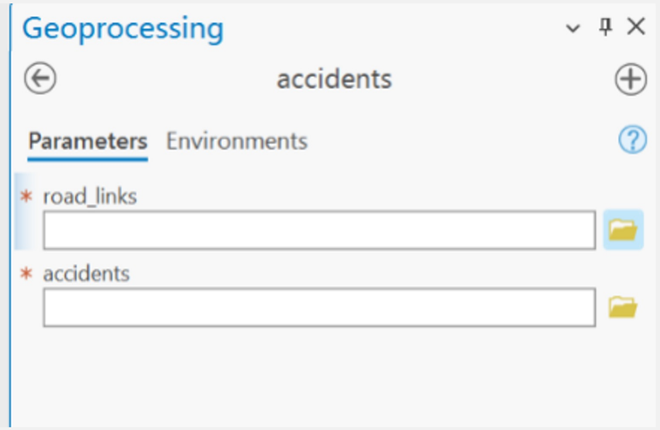
Output:

Updated link shapefile with a field holding the number of accidents

PYTHON SCRIPT TOOL

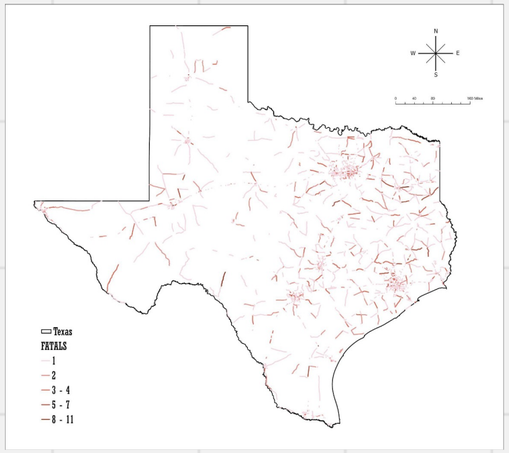
```
IMPORT ARCPY

try:
  road_links = arcpy.GetParameterAsText(0)
  accidents = arcpy.GetParameterAsText(1)
  fatal_field = "Fatals" # Replace 'Fatals' with the actual field name in your accidents dataset that contains fatality counts

  # Create a feature layer from accidents data for spatial analysis
  accidents_layer = arcpy.MakeFeatureLayer_management(accidents, "accidents_layer")

  # Add fields to store the counts of accidents and fatalities
  arcpy.AddField_management(road_links, "Total_Accidents", "DOUBLE")
  arcpy.AddField_management(road_links, "Fatalities", "DOUBLE")

  # Create buffers around road links
  buffer_distance = "50 Meters"
  buffer_output = "in_memory\\buffered_roads"
  arcpy.Buffer_analysis(road_links, buffer_output, buffer_distance)

  # Intersect the buffered road links with accidents
  intersect_output = "in_memory/intersect_output"
  arcpy.Intersect_analysis([buffer_output, accidents_layer], intersect_output)

  # Update accident counts in the intersect_output
  arcpy.AddField_management(intersect_output, "accident_count", "DOUBLE")
  with arcpy.da.UpdateCursor(intersect_output, ["accident_count"]) as cursor:
    for row in cursor:
      row[0] = int(arcpy.GetCount_management(accidents_layer).getOutput(0))
      cursor.updateRow(row)

  # Join the calculated accident counts back to the original road links layer
  arcpy.JoinField_management(road_links, "OBJECTID", intersect_output, "OBJECTID", ["accident_count"])

except Exception as e:
  arcpy.AddError(str(e))
```

HOW THE TOOL LOOK LIKE IN GIS

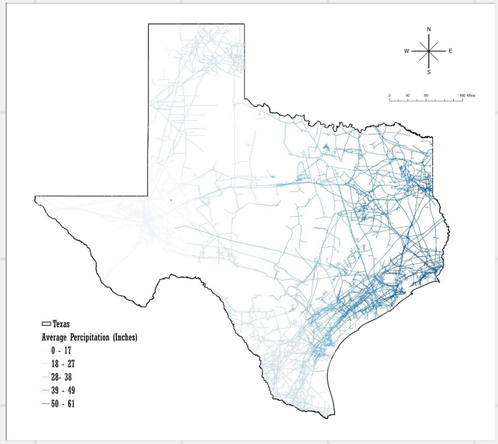


Join_Count	FATALS
7	11
2	9
7	7
7	7
5	6
3	6
1	6

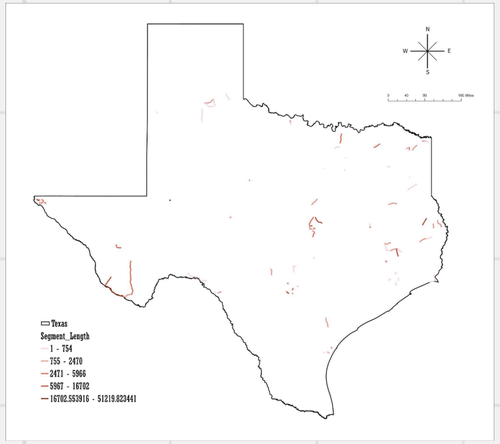
OUTPUT



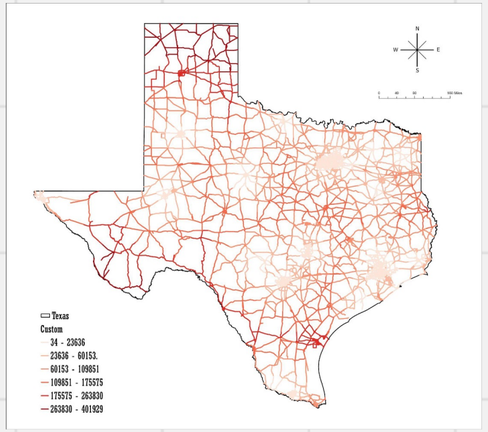
Fatalities



Average Precipitation



Vulnerable Land



Nearest Hospital Distance

Some outputs visualized

Scripts are developed for all modes

Calculated Variables → Risk Model → Risk Score

Segment_Length	Buffer_Area	SUM_12_13_14	precip_avg	HOSP_DIST
27404.400234	0.014251	21.947788	57	0.39163
24038.755806	0.036445	6349.369962	56.5	0.389357
23310.475617	0.026829	1449.280787	59	0.33563
22897.028975	0.037416	1960.007376	12	0.411413
20108.874094	0.027525	1927.558027	24	0.222993
16131.885429	0.016401	94.027557	34.333333	0.38857
11892.22506	0.011319	41.397817	57	0.217826
11821.152321	0.009823	0.929209	41	0.222993
10774.032192	0.007462	10.911347	45	0.383358
10545.7558	0.007435	7.857397	58.5	0.388376
9673.20835	0.009093	58.899377	54.5	0.374687
8902.835737	0.018492	73.242639	12	0.383358
8667.420818	0.052604	23.361253	55	
8615.897459	0.017858	742.279503	32.333333	
8490.677795	0.010705	1.033868	45	

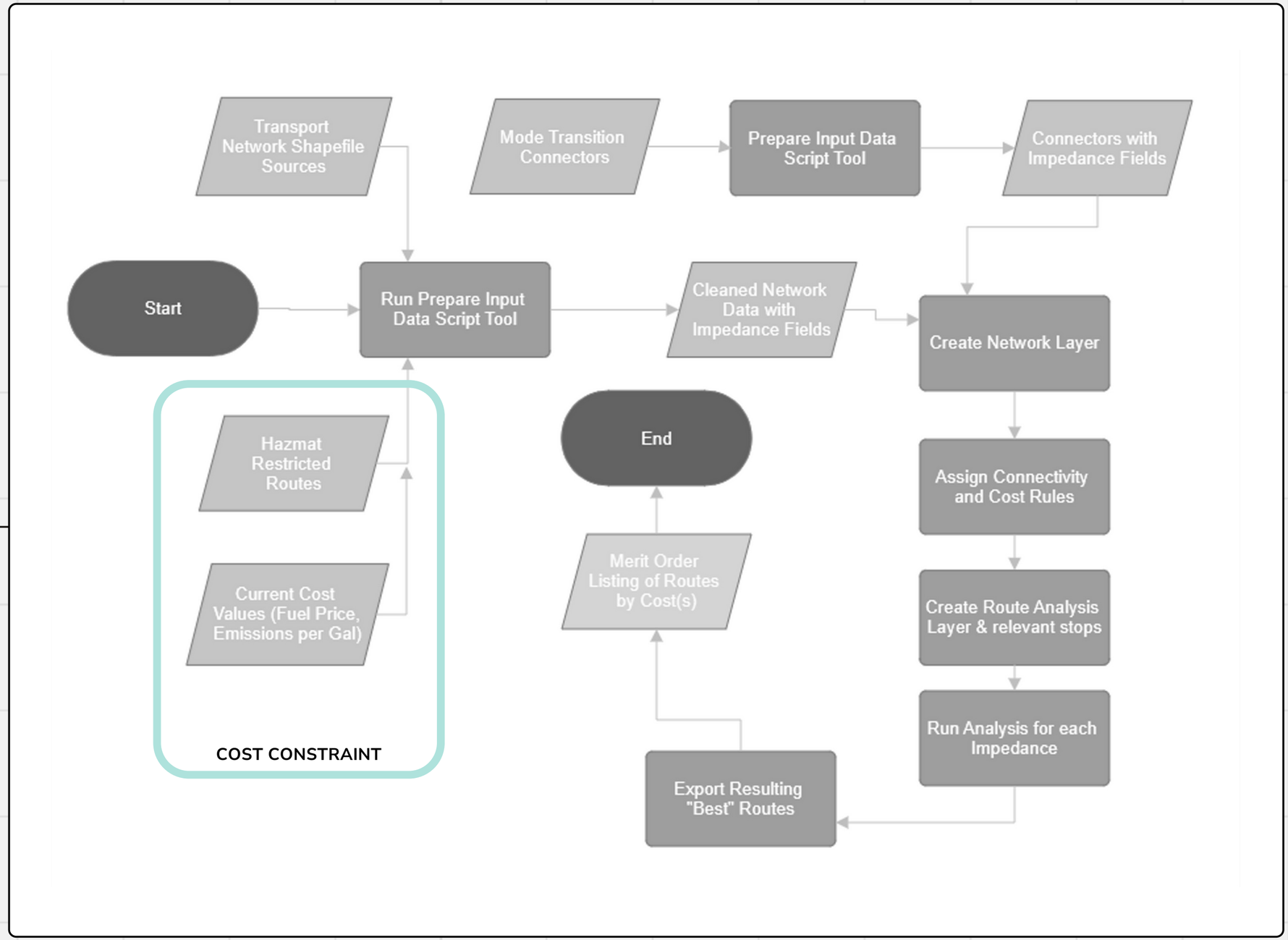
$$
\begin{aligned}
Risk_T &= Frequency_T \cdot Severity_T \\
&= f\vec{x}\vec{W} \\
&= fw_1X_1 + \ldots, fw_6X_6.
\end{aligned}
$$

Risk_Score
0.55013
0.363179
0.365514
0.333914
0.258203
0.310667
0.784629
0.242933

WEIGHTED RISK SCORE CALCULATION

@COST

COST CONSTRAINT



IMPROVED COST MODEL

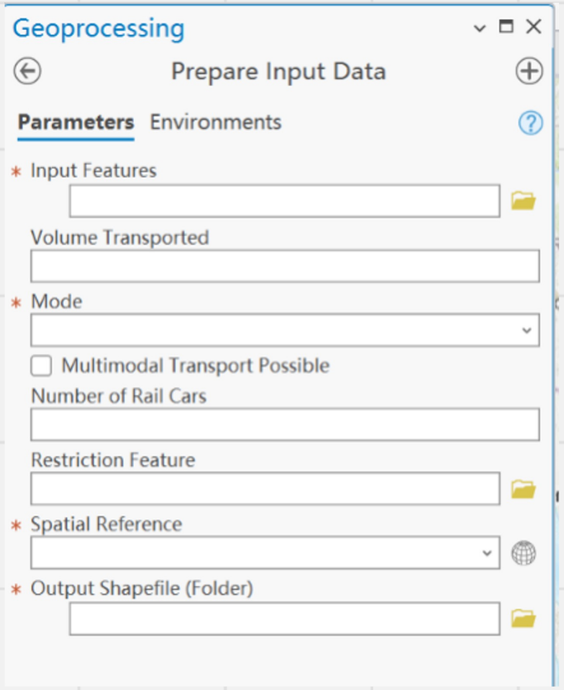
ORIGINAL COST MODEL

Previously used following variables to find Cost in $/mile…

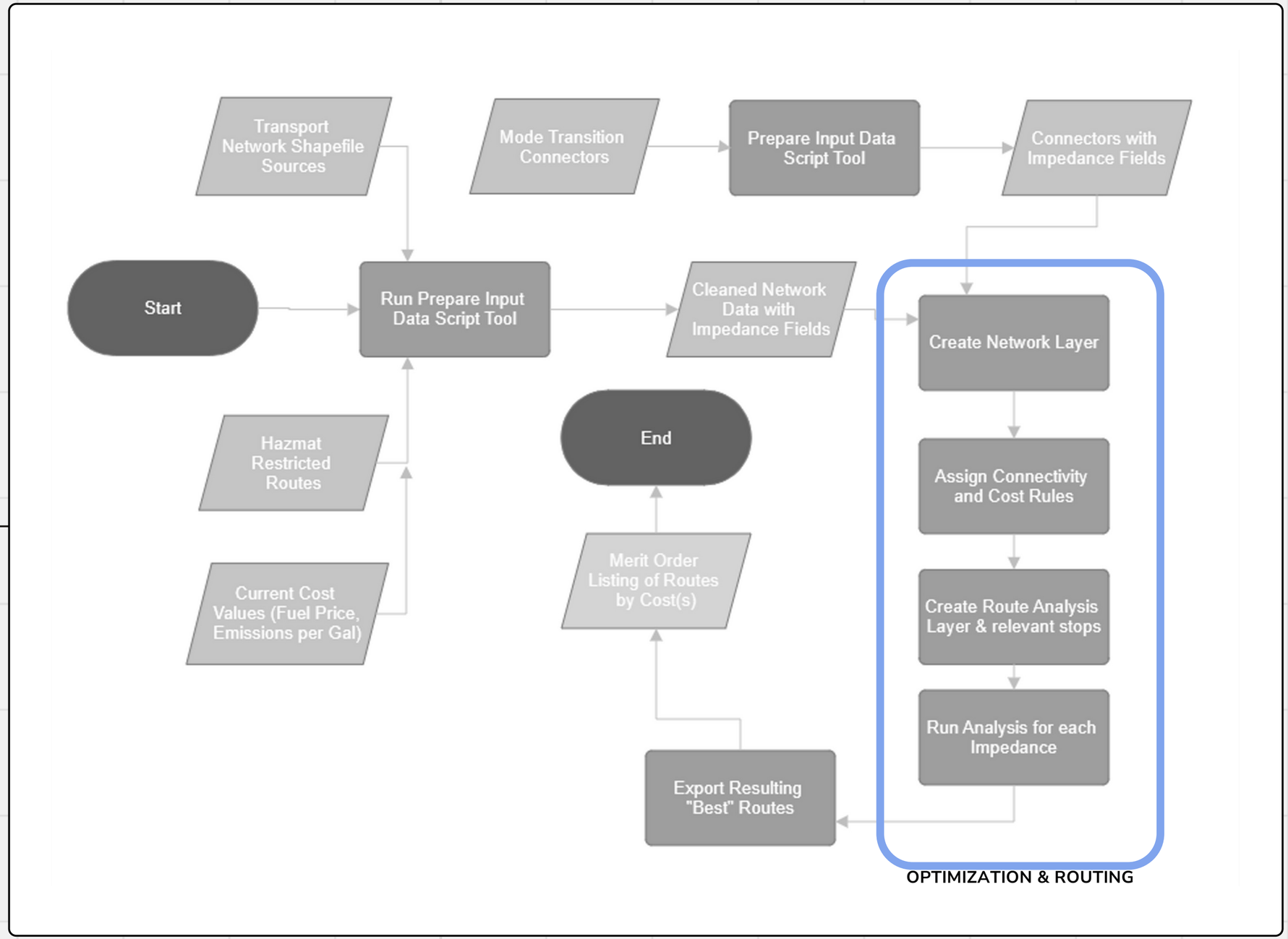
- Fuel cost per unit of mode
- Network edge link (considering single or round trip by mode)
- Relevant mode vehicle capacities

UPDATED COST MODEL

- Additional time (hour) calculations
- Standardize cost to $/mile/tonne
- Implemented
  - Pipeline mode (not multimodal)
  - Different capacities for multimodal rail cars
  - Different impedance specificities with Risk



OPTIMIZATION & ROUTING



Tool 6 Create Network Dataset with the Pipeline, Railroad, Trucking Routes holding the calculated Risk Score/Cost fields

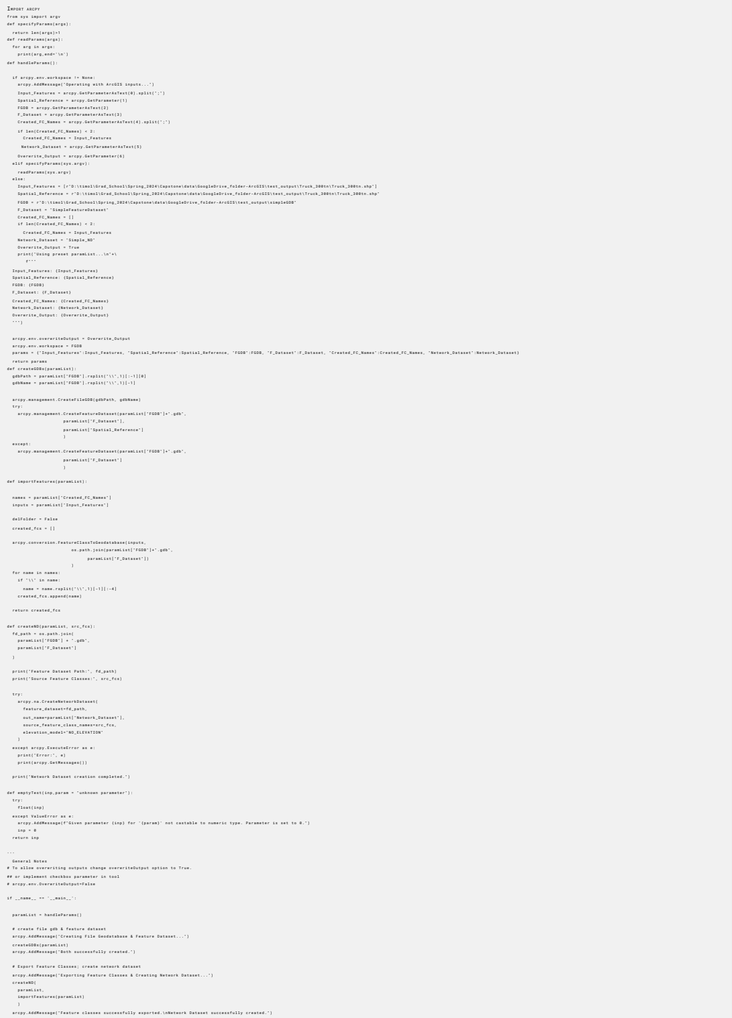
Input:

Pipelines, railroads, trucking routes with calculated fields, connectors, nodes

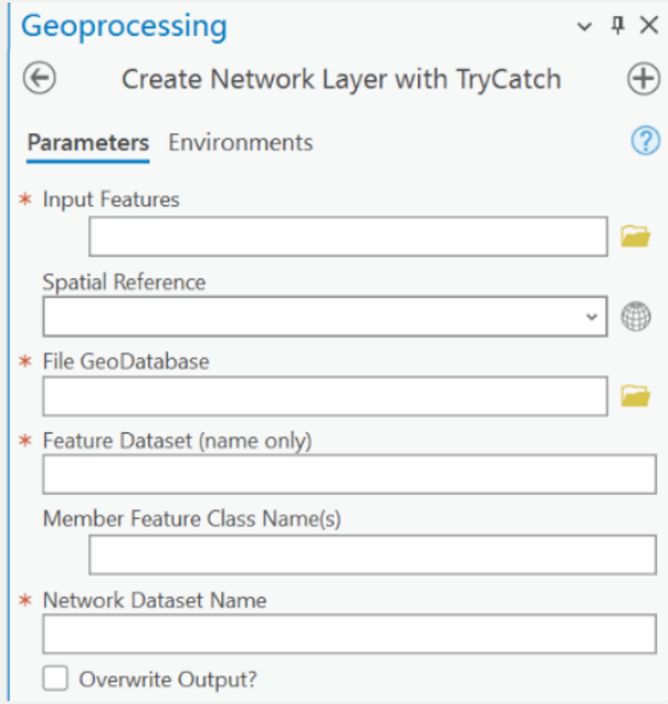
Output:

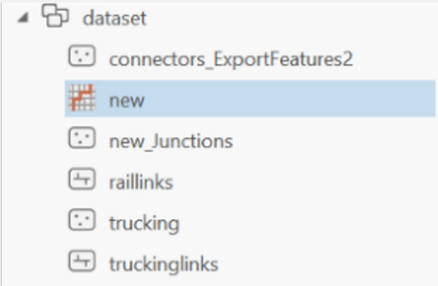
Network dataset & network analysis layer

PYTHON SCRIPT TOOL



HOW THE TOOL LOOK LIKE IN GIS





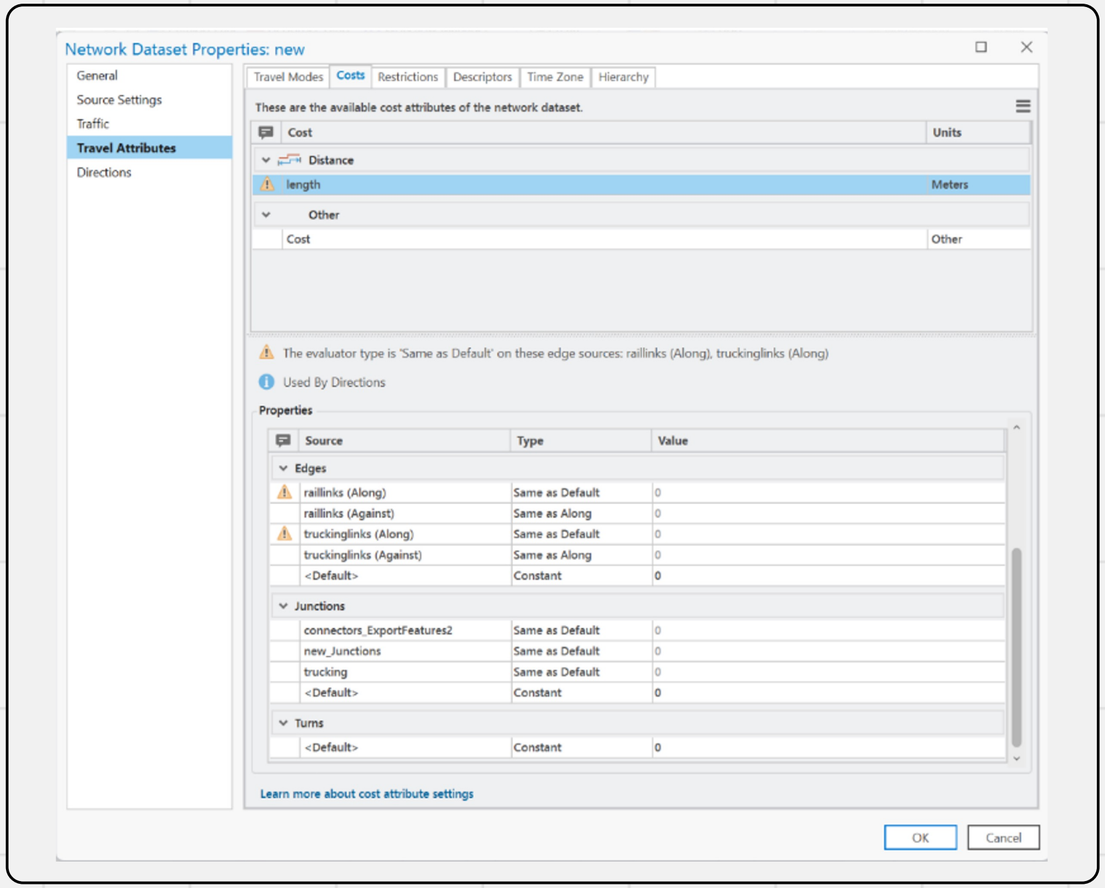
OUTPUT

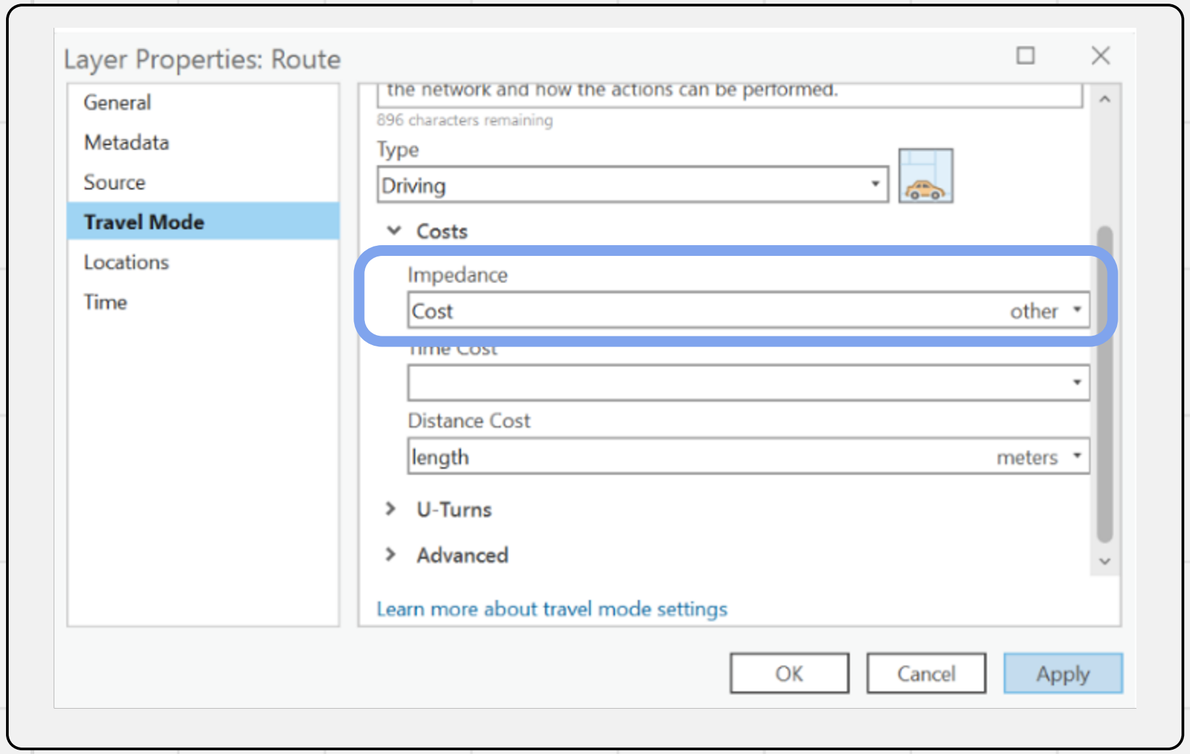
SET UP NETWORK ANALYSIS LAYER (MANUAL STEP)

Set up network analysis properties

Change Impedance:

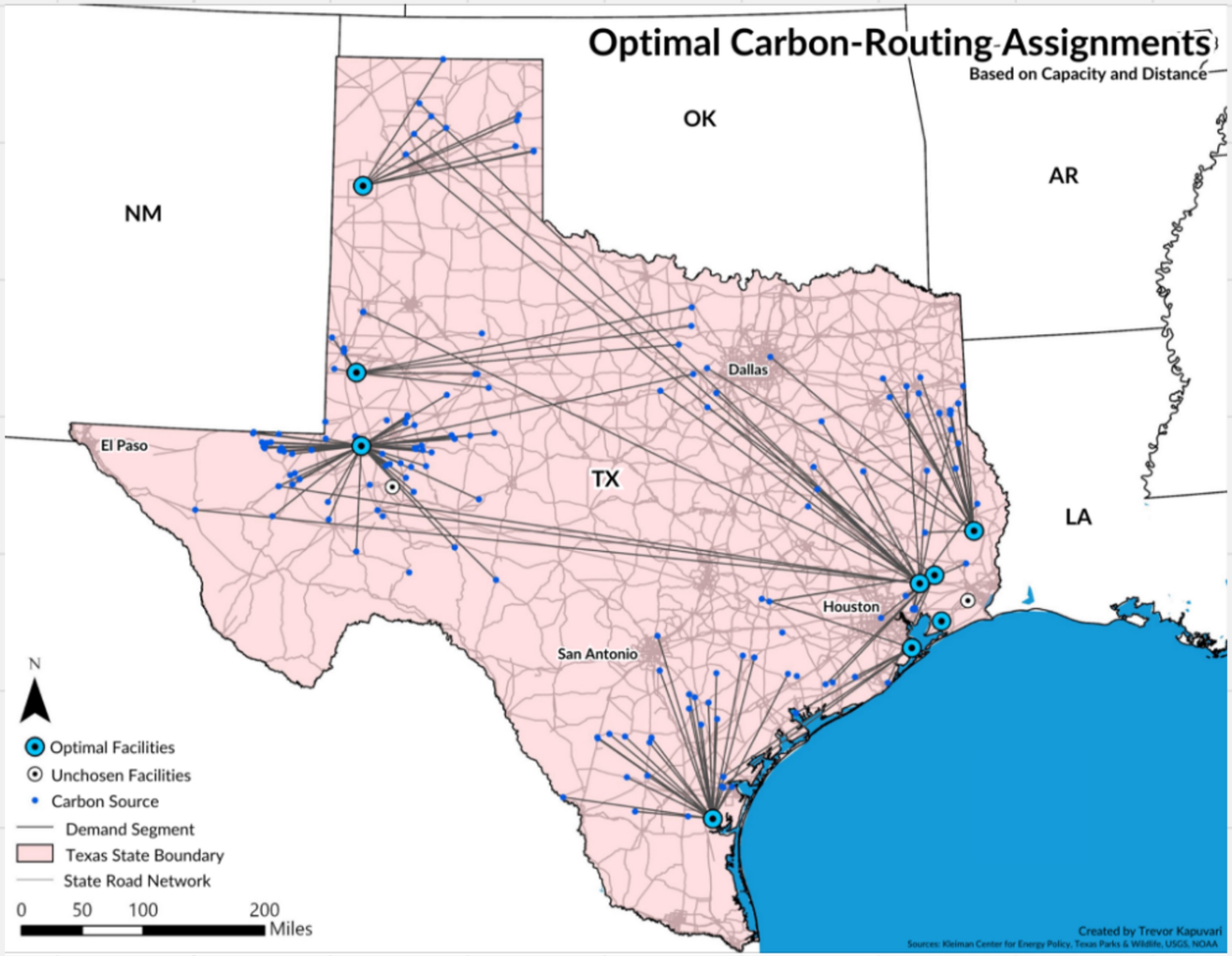
Optimize Cost or Risk_Score





FINDING OPTIMIZED ORIGIN-DESTINATION PAIRS

If the origin (CO2 emission sources) and destination (CO2 storage facilities) data are not already paired, an additional step is necessary to optimize these pairings based on the carbon storage capacity of the facilities. This is essential because origins and destinations must be paired to conduct network routing optimization effectively.



This od-pair optimization is conducted using CPLEX:

Minimize Travel Distance

Tool 7 Input Origin-destination pairs, Solve, Output the Optimized Route and Breakdown Table

Input:

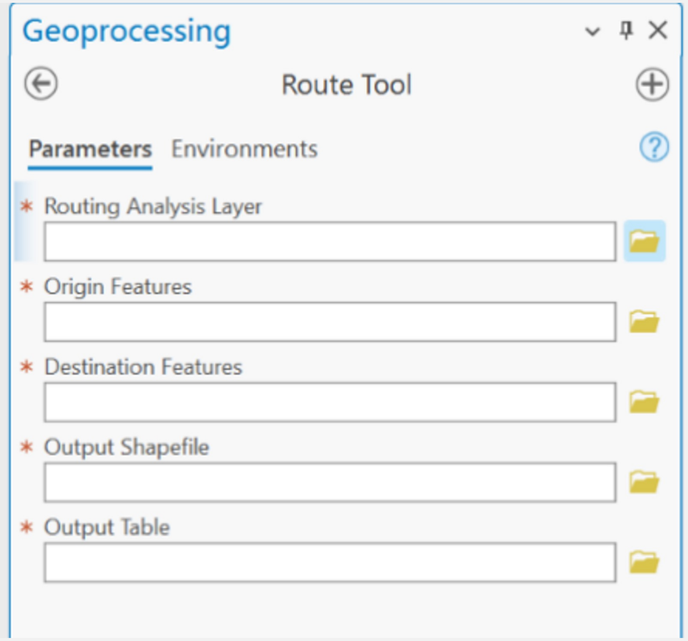
origin-destination pairs network analysis layer

Output:

1. Visualization of the chosen route
2. Cost/risk breakdown table of the chosen routes

PYTHON SCRIPT TOOL

HOW THE TOOL LOOK LIKE IN GIS



@MUSA/CPLN

SPRING 2024

MUSA/CPLN CAPSTONE

RESULTS & OUTPUT

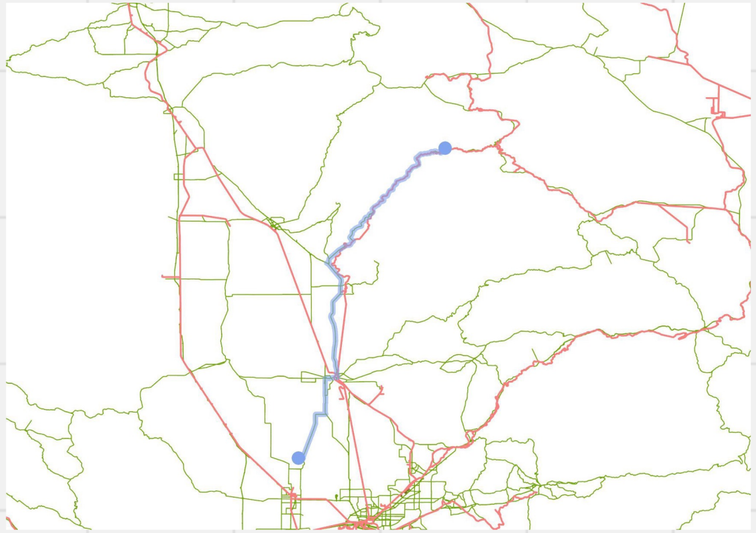
OUTPUT

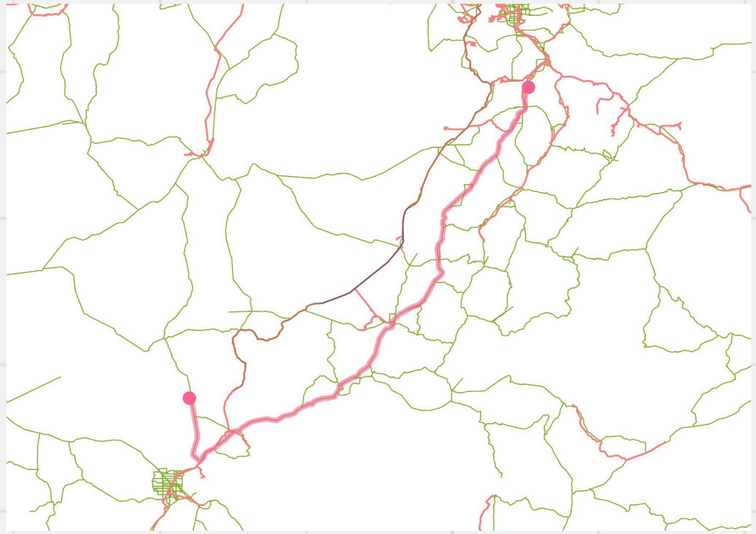
Routing	Origin-Destination Matrix

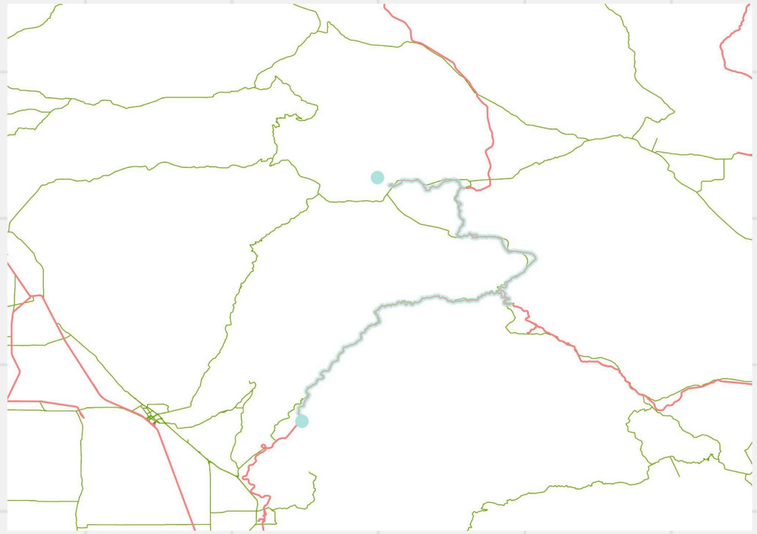
• **Optimal Route for COST and RISK Impedance with…**
  ○ **$Truck_{Risk}$**
  ○ **$Truck_{Cost}$**
  ○ **$Rail_{Risk}$**
  ○ **$Rail_{Cost}$**
  ○ **$Pipeline_{Risk}$**
	• **Impedance values for solution pairs**

OUTPUT Example

Route Shapefile







Cost & Risk Impedance Breakdown

Route ID	Risk	Cost	TruckRisk	TruckCost	RailRisk	RailCost	PipelineRisk	PipelineCost
1								
2								
3								

Reference

NASA. (2024). The Keeling Curve. https://climate.nasa.gov/vital-signs/carbon-dioxide/?intent=121

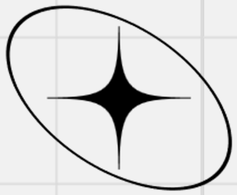
Roads to Removal. (2024). https://roads2removal.org/

Great Plains Institute. (2022). https://betterenergy.org/blog/gpi-carbon-and-hydrogen-hubs-atlas/

Statista. (2023). https://www.statista.com/statistics/726634/large-scale-carbon-capture-and-storage-projects-worldwide-capacity/

Congressional Budget Office. (2023). https://www.cbo.gov/publication/59832

IEA. (2024). https://www.iea.org/energy-system/carbon-capture-utilisation-and-storage



THANK YOU