Modeling CO_2 Storage Pipeline Routes in the United States

by

Kevin Fritze

Dr. Lincoln Pratson, Advisor
2009

Master's Project submitted in partial fulfillment of the
requirements for the Master of Environmental Management degree
in the Nicholas School of the Environment of Duke University

April 2009

Abstract

Carbon capture technology offers the potential for the United States to continue the widespread use of coal to generate electricity in a carbon constrained economy. This study examines potential configurations of a nationwide carbon capture and storage network by integrating work on the capture, transportation, and storage aspects of such a network. Modeling conducted with the Department of Energy's National Energy Modeling System (NEMS) model provided data on which plants will retrofit or be built with carbon capture technology in the future. A cost surface and several models were developed using geographic information system (GIS) software to site new plants and connect all carbon capture plants to geologic storage basins throughout the United States.

Several network scenarios with different levels of foresight and government pipeline construction assistance are examined. Building a trunkline increases the overall length of the pipeline network. Assuming all pipelines are the same diameter, initial cost estimates indicate that building a trunkline network is more expensive than allowing plants to connect to storage sites on their own. More detailed information on pipeline diameter is needed to develop accurate cost estimates of different pipeline network configurations to further inform decision makers.

Table of Contents

List of Figures

Introduction:

In the past few years public and political interest in the United States in taking action to reduce human impacts on the climate has increased significantly. Emissions of greenhouse gasses (GHG's) from the electric power sector make up 40 percent of total carbon dioxide (CO_2) emissions in the United States.[1] Coal-fired power plants account for 50 percent of total electricity generation in the U.S., 27 percent of the country's total GHG emissions, and 83 percent of electric sector CO_2 emissions.[2] The country's vast coal reserves and current reliance on coal-fired power plants for baseload generation have prompted the development of technologies and policies that will allow coal to remain a significant component of electricity generation in a carbon constrained economy.

Carbon capture and storage (CCS) technology has the potential to allow fossil fuel power plants to operate while emitting significantly less carbon dioxide than they currently do. In 2006 the Department of Energy's National Energy Technology Laboratory (NETL) studied the costs of retrofitting AEP's Conesville Unit #5 with carbon capture equipment to determine the technical and economic feasibility of the technology. The cost of capturing CO_2 varied with the amount captured. The levelized cost of electricity (LCOE) increased between 2.31 to 6.92 cents per kilowatt-hour (kWh) and the cost per ton of CO_2 captured decreased from $113 to $89 as the capture percentage increased from 30 to 90 percent. These values demonstrated that carbon

[1] Energy Information Administration, "Greenhouse Gases, Climate Change & Energy", May 2008, available at <http://www.eia.doe.gov/bookshelf/brochures/greenhouse/greenhouse.pdf>

[2] Ibid.,
Pew Center on Global Climate Change, "Coal and Climate Change Facts", 2009, available at <http://www.pewclimate.org/global-warming-basics/coalfacts.cfm>

capture technology is economically feasible. The report also found that there are no major technical barriers preventing the installation of carbon capture retrofit technology. [3]

The technology to inject CO_2 into geologic reservoirs is already mature and in use today in enhanced oil recovery (EOR) operations, where CO_2 is pumped into a reservoir to increase pressure and drive out more oil and gas. While EOR can provide some opportunities for storage, particularly in the near-term, deep saline aquifers offer the greatest potential for large-scale, long-term storage. In their special report on carbon capture and storage the Intergovernmental Panel on Climate Change (IPCC) found that the potential storage in saline formations is at least as much as in existing oil and gas reservoirs, and could be several orders of magnitude greater.[4] Research conducted at Duke University to identify potential deep saline storage basins in the United States is used to provide CO_2 storage site locations in this study.

Given the technical and economic feasibility of carbon capture and storage technologies, and a sufficiently high financial cost to emit carbon dioxide, the widespread adoption of CCS technology is likely following the adoption of climate legislation in the United States. The question that then arises is, what would a nation-wide CCS network look like? Which plants would retrofit with or be built with carbon capture technology, where would the storage sites be located, and where would the pipelines be built to transport the captured carbon to be stored?

Some work has been done to examine the routing of CO_2 pipelines in a CCS network. Researchers at Oak Ridge National Laboratory and Harvard's Kennedy School developed a scalable model to minimize the cost of storing a given amount of CO_2 annually. The model determines how much to store in which reservoirs and builds the least costly pipelines to

[3] National Energy Technology Laboratory, "Carbon Dioxide Capture from Existing Coal-Fired Power Plants", November 2007, available from <http://www.netl.doe.gov/energy-analyses/pubs/CO2%20Retrofit%20From%20Existing%20Plants%20Revised%20November%202007.pdf>

[4] Intergovernmental Panel on Climate Change, "Special Report on Carbon Dioxide Capture and Storage", 2005, available at <http://www.ipcc.ch/ipccreports/srccs.htm>

construct to facilitate that storage. They demonstrated the model by examining a potential CCS network on California using 37 sources of CO_2 and 14 storage reservoirs in the state.[5] A group of companies have joined together to create the Integrated CO_2 Network (ICO_2N) in Canada, a CCS network designed to lay the foundation for a nationwide network. The plan is to build the network in the Western provinces of British Columbia, Alberta, and Saskatchewan first, and eventually expand eastward.

What is still needed is an examination of a nationwide CCS network. Modeling a nationwide network will facilitate understanding of the scale and cost of transporting captured CO_2 on such a large scale. In addition, it will allow for the examination of potential government assistance in the construction and operation of a pipeline network. A government sponsored trunkline connecting several major storage basins can provide the storage network with redundancy in case of problems at some basins. It could also provide broad cost benefits to CO_2 sources by the length of the pipeline needed to connect sources to the storage network.

Approach:

This project developed the methods for examining a nationwide CCS network under different policy scenarios. Geographic information system (GIS) software is used to link sources of CO_2 to storage basins. The routes are optimized based on cost so that the lowest cost route from each source to the storage network is used to build the network in each scenario. Different pipeline scenarios are explored with and without a trunkline between some storage basins. The timing of CO_2 sources joining the storage network is varied as well, with sources being phased in yearly or all starting to capture their emissions in the same year.

[5] Richard Middleton and Jeffrey Bielicki, "A scalable infrastructure model for carbon capture and storage: *SimCCS*" Energy Policy 37 (2009) 1052–1060

Data Inputs:

CO_2 Sources:

The Climate Change Policy Partnership (CCPP) at Duke University used the Energy Information Agency's (EIA) National Energy Modeling System (NEMS) 2008 model to determine which plants are likely to retrofit or be built with carbon capture and storage technology in the future under different scenarios.[6] The default EIA NEMS model does not have an option to estimate carbon capture retrofits to existing plants, so the National Energy Technology Laboratory developed code to allow the model to do so.[7]

The code developed by NETL can only model one type of carbon capture retrofit technology per scenario. There are also no restrictions on the installation of CCS retrofits, if the model determines that a plant can retrofit economically then it will do so. Real world barriers to installation, such as constraints on materials, engineering, and construction; issues related to regulation, permitting, and siting; and the state of CO_2 storage infrastructure, are not taken into account.

The CCPP gathered cost information for both conventional and carbon capture technologies, including amine and chilled ammonia based retrofit technology. Cost data for advanced monoethanolamine (MEA) technology was obtained from the NETL report describing its carbon capture retrofit modeling in NEMS.[8] Cost estimates for chilled ammonia were obtained from a study by the Electric Power Research Institute (EPRI).[9] This information was

[6] Eric Williams and Munish Chandel, "Modeling the adoption of carbon capture technologies", in development
[7] Rodney Geisbrecht, "Retrofitting Coal-Fired Power Plants for Carbon Dioxide Capture and Sequestration - Exploratory Testing of NEMS for Integrated Assessments", National Energy Technology Laboratory, Jan 18, 2008, available at <http://www.netl.doe.gov/energy-analyses/pubs/v%203%20-%20FINAL%20-%20retrofit_NEMS_exploratory.pdf>
[8] Ibid.
[9] Richard Rhudy and Sean Black, "Chilled Ammonia Process Update", presentation at CO_2 Capture Network in Lyon, France, May 24, 2007, available at <http://www.co2captureandstorage.info/docs/capture/10th%20cap%20network%20web%20files/K%20-%20Rhudy%20-%20Chilled%20Ammonia%20as%20solvent.pdf>

used to update the cost assumptions in NEMS to reflect recent changes in costs that were not accounted for in EIA's default input assumptions.

NEMS scenarios were developed based on different technologies, levels of carbon capture, and carbon price. The CCS network configurations described in this analysis are based on the NEMS results of using MEA technology to capture 90% of carbon emissions with a carbon price of $20 per ton beginning in 2012 and increasing at 8% per year.

CO_2 Source Locations:

The NEMS model output includes plant codes that can be matched to EPA's eGRID database to determine the plant's latitude and longitude coordinates. This provides the location of existing plants that retrofit with carbon capture technology under each scenario. To meet future demand NEMS creates new plants as needed which do not have latitude or longitude information, and the location of these plants must be estimated as accurately as possible to determine the resulting pipeline network in each scenario.

Census and NERC Regions

NEMS specifies the census and North American Electric Reliability Council (NERC) region that a new plant is located within. Using these overlapping regions narrows the possible area within which a new plant can be located.

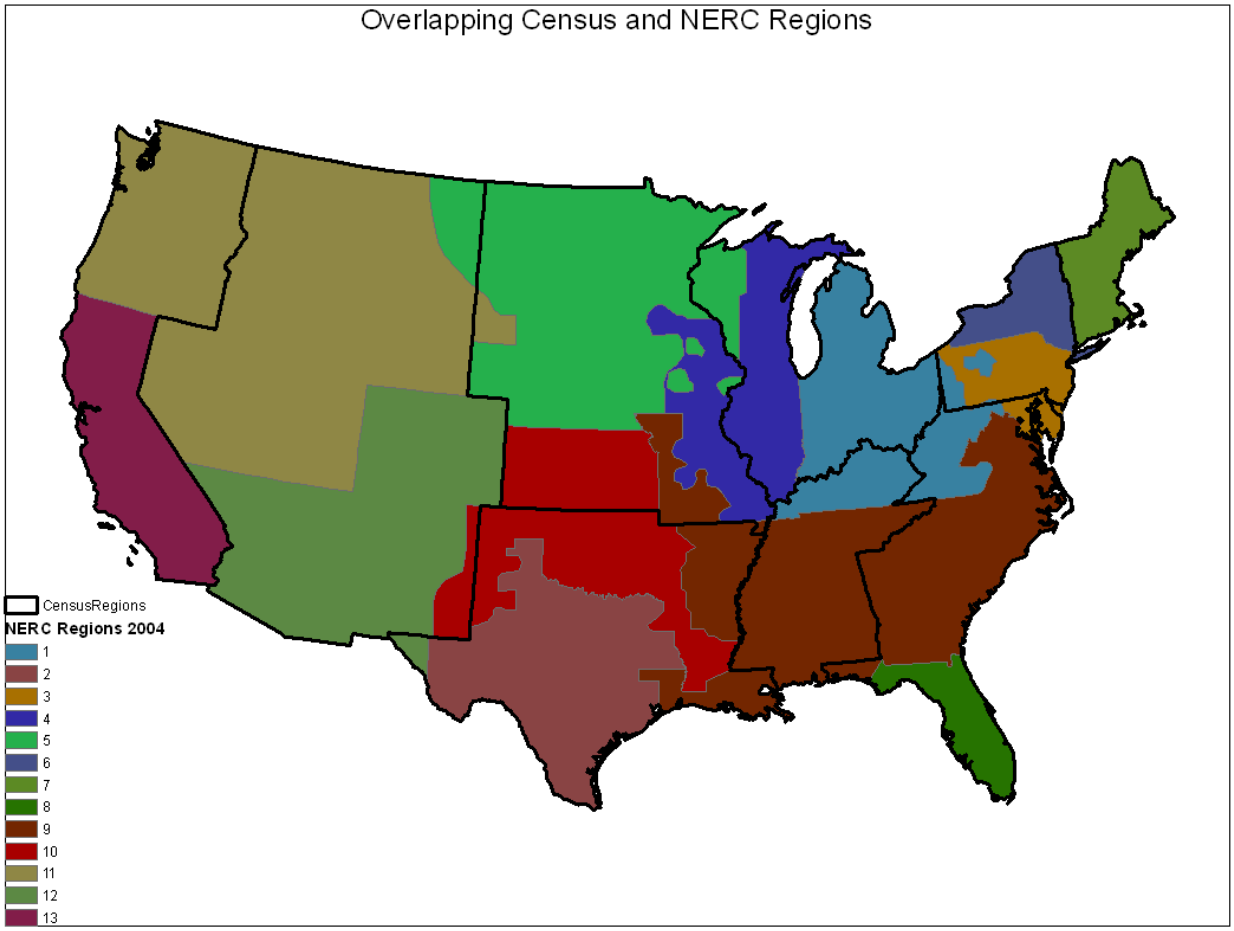

Figure 1: Overlapping census and national electricity reliability council regions.

As can be seen from figure 1 the census regions cut across many of the NERC regions. This allows the placement of new plants to be more refined as it can eliminate some sections of the NERC region. In order to further refine the placement of new plants the location of existing plants was studied with regard to their proximity to existing relevant infrastructure, including rivers, railroads, pipelines, and transmission lines.

Infrastructure

Almost all coal plants within EPA's eGRID database are located within one mile of transmission lines and either a railroad or major river. This provided another parameter to inform the placement of new plants created by NEMS. To identify suitable areas to build new plants transmission lines, rivers, and railroads were buffered by 1 mile on either side and the

intersection of buffered transmission lines and buffered railroad or river area were selected out. Areas that were within medium and heavy urban development were removed to reflect the difficulty and undesirability of building a new power plant in such an environment. The remaining intersection areas are the potential areas within which a new plant can be placed.

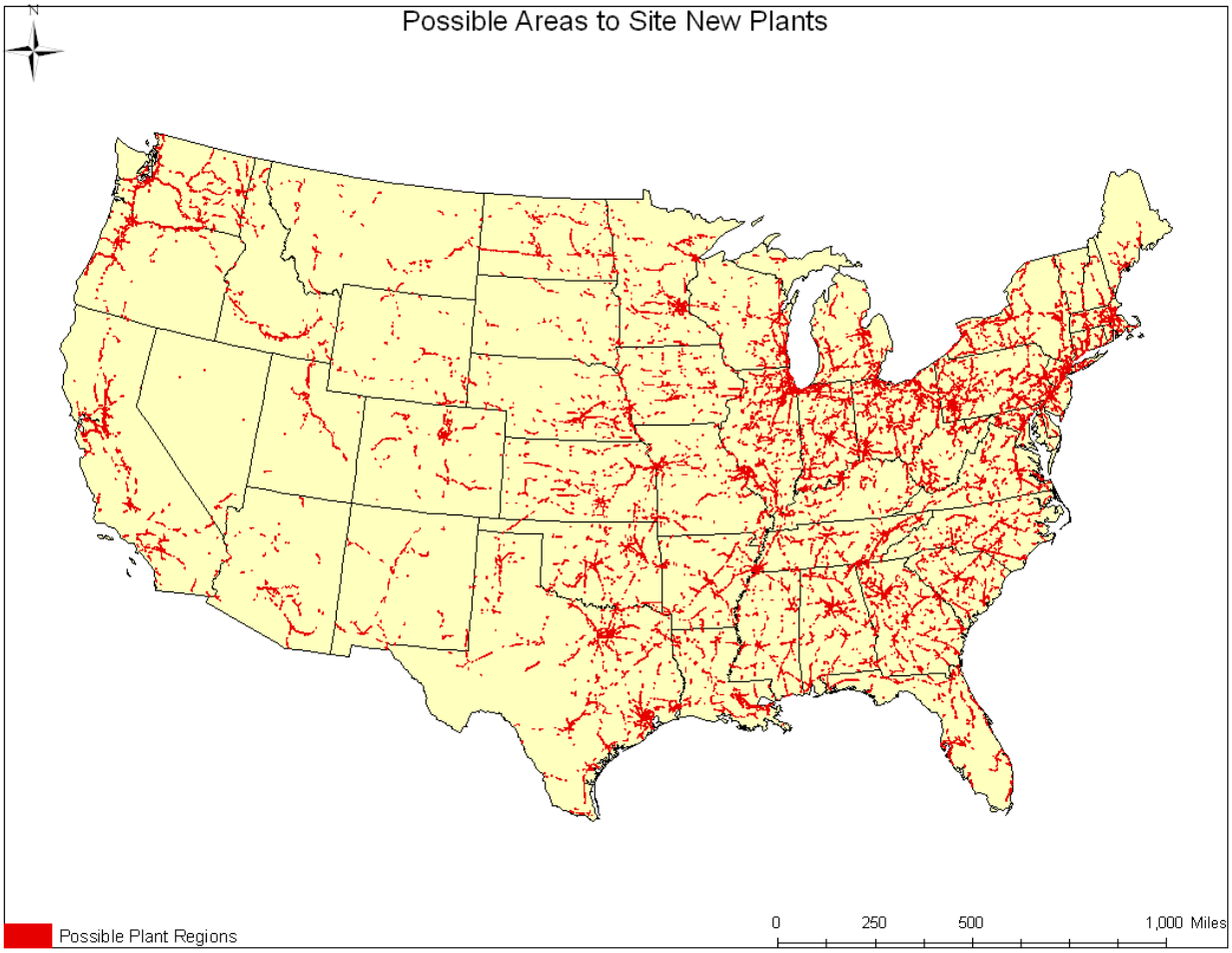

Figure 2: Areas where new plants are allowed to be cited.

Carbonsheds

While the construction of new plants within the intersection of 1-mile buffered transmission lines with buffered railroads or rivers minimizes the cost of fuel transportation and electricity transmission, a consideration in a carbon constrained economy will also be the cost of transporting captured CO_2. The cost to transport CO_2 from a location to each of the storage sites was determined through the development of 'carbonsheds'.

The cost to transport CO_2 to each storage site from a given location was calculated using a fixed pipeline construction cost multiplied by a cost surface and added to the cost of storing carbon dioxide in the sequestration site. The cost surface, described in greater detail below, indicates the cost to build a pipeline across any given cell. The cost distance from each storage site extends to the point at which it would cost the same for a plant to send its CO_2 to another storage site, creating carbonshed boundaries. The boundaries represent the area within which a plant will send its CO_2 to the carbonshed's storage site because it is the least costly storage option available to it.[10]

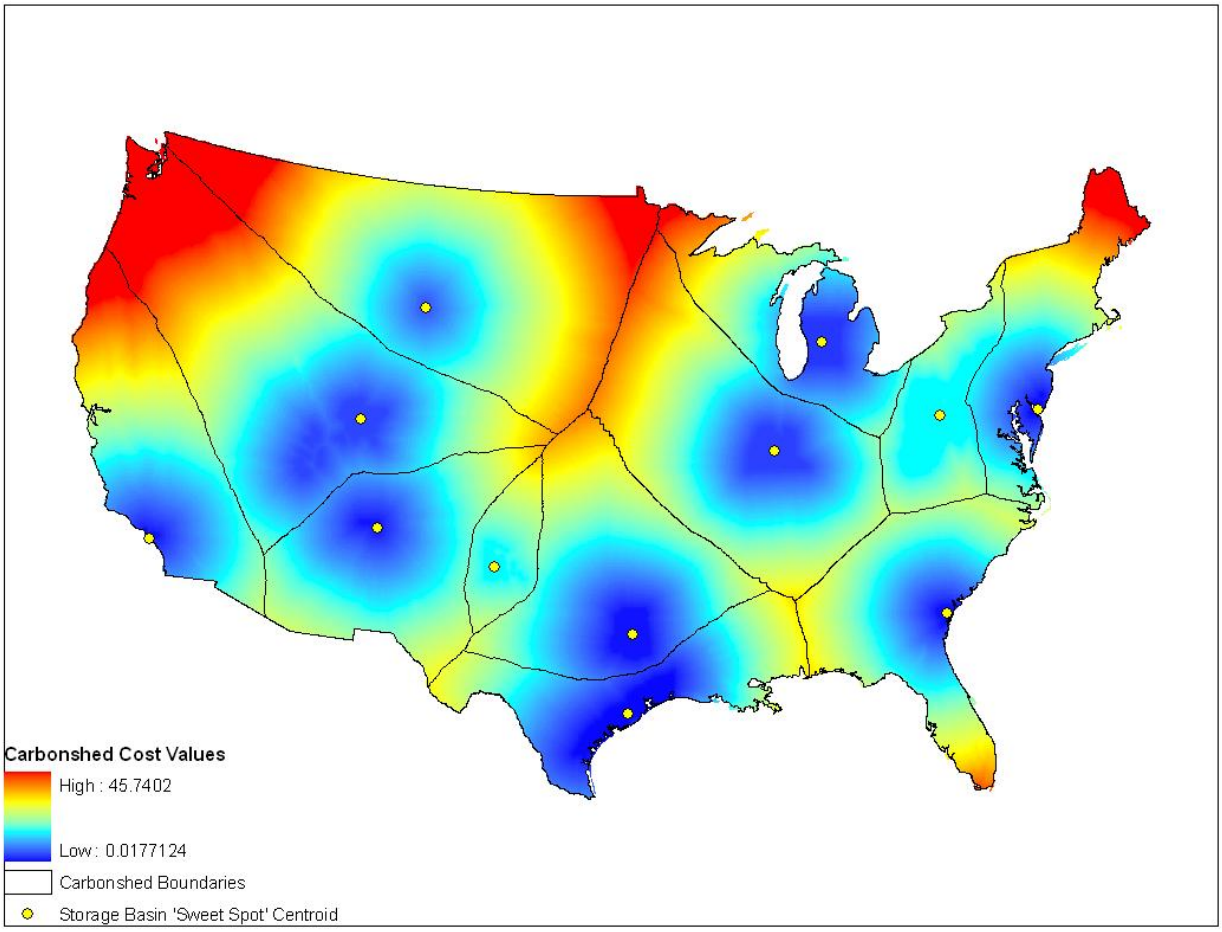

Figure 3: Carbonsheds

[10] Jordan Eccles, "Modeling CO_2 Transportation and Storage Costs", in preparation

Plant Placement Model

An ArcGIS model was created to locate the new plants created by NEMS in each scenario. The model uses the carbonsheds to minimize costs related to CO_2 transportation and storage, and the 1 mile buffered intersection areas described above to minimize infrastructure costs for transporting fuel and electricity.

To place each new plant in the region specified by NEMS the 1 mile buffered areas that are within each combined NERC/census region were selected out to be the allowable locations to build new plants. The highest priority location within those areas is determined by selecting the carbonshed cell(s) with the lowest cost value. The model then creates a point within that cell to represent the location of the new plant.

To ensure that new plants are not build on top of each other each new plant is buffered by a distance of 30 miles and all potential new plant areas within that buffered area are removed. The buffer distance was chosen based on the average distance of existing coal power plants in EPA's eGRID database from each other. The model repeats, finding the lowest cost carbonshed cell within the remaining potential new plant areas and placing a plant there, until locations for all new plants have been generated.

Storage Site Locations:

The carbon sequestration sites used in this analysis consist of deep saline aquifers within sandstone formations for which data was available to analyze the cost of CO_2 storage. Eccles, et. al. found that there is large storage potential available throughout the United States with significant variability in the capacity and cost of storage within individual storage basins.[11] The

[11] Eccles, et al, "Physical and Economic Potential of Geological CO2 Storage in Saline Aquifers", Environmental Science & Technology, 2009, 43 (6), 1962-1969

analysis of storage costs in that paper provides the basis for the point locations used in the pipeline modeling.

To represent the storage basins in this analysis single point locations were used. The points represent the centroids of each basin's storage 'sweet spot'. The 'sweet spot' area consists of the 500 meter cells in the basin within which the cost of storage is less than the average cost of storage within the entire basin. Out of 15 basins examined, 3 were too costly to send CO_2 to compared with shipping the CO_2 to another basin for storage, leaving 12 basins for use in constructing a storage network.[12]

Cost Surface:

In order to calculate pipeline routes a cost surface was constructed which represents the relative difficulty (and thus cost) of constructing a pipeline through various types of terrain. The cost surface was constructed as a raster layer of the continental United States with 500 meter-squared cells. The cell values were designed on a relative scale, with 100 being the cost of building a standard pipeline without an existing right-of-way on ideal terrain. Based on data from the Oil and Gas Journal[13] and verified by pipeline experts, the fixed costs associated with pipeline construction represent 90 percent of the cost of building a pipeline. Each cell in the cost surface is given a base value of 90 to represent this fixed cost, and the value of each cell is changed from that base value depending on the terrain features of that cell.

Existing Infrastructure

The three primary variables that influence the cost of building a pipeline through a given cell are landcover, right of way, and slope. The presence of existing infrastructure was determined using a composite data layer which included the location of major pipelines,

[12] Jordan Eccles, "Modeling CO_2 Transportation and Storage Costs", in preparation
[13] Warren True, "Pipeline Economics: Profitable 2000, higher demand push US natural gas construction plans", Oil & Gas Journal, volume 99, issue 36, Sept. 3, 2001

transmission lines, railroads, and roads.[14] If a cell had an existing right of way in it, a value of 1 was added to the cell. If no existing right of way was present the cell was given a value of 10. This reflects the additional costs that would be associated with obtaining a new right of way for pipeline construction relative to building along an existing right of way.

Slope

In the case of the slope of a cell, the average slope of the terrain within a cell was calculated and based on that a cost value was assigned to that cell. The slope data layer was developed from a 500 meter resolution digital elevation model (DEM) from the U.S. Geological Survey.[15] If the cell had an average slope of less than 10 degrees the cell was given a value of 0. If the average slope was greater than 10 degrees but less than 20 degrees the cell was given a value of 5. Cells with an average slope greater than 20 degrees but less than 30 degrees were given a value of 10, and any cells with an average slope greater than 30 degrees were given a value of 15.

Landcover

The landcover variable had the greatest potential to significantly influence the cost of pipeline construction through a given cell. National landcover data for 2001 was obtained as 30 meter resolution Landsat 7 imagery from the Multi-Resolution Land Characteristics Consortium. This data was aggregated to 500 meter resolution by classifying each cell as the majority landcover represented within it.[16]

Barren terrain and grassland provide ideal conditions for pipeline construction, and thus any cells classified as barren or grassland were given values of 0 for landcover. Cells dominated

[14] Anna Frankel, "A Geospatial Analysis of Pathways for Carbon Sequestration" April 25, 2008, p. 5, available at <http://hdl.handle.net/10161/534>
[15] Ibid. p. 4
[16] Ibid. p. 5-6

by agricultural areas were given a value of 20, reflecting the additional difficulties of navigating the pipeline around farming infrastructure and property. Forested areas require additional preparation work which increases the cost of pipeline construction approximately 30 percent, so cells dominated by forest were given a value of 30. Cells dominated by wetlands were given a value of 100 to reflect the even greater preparation work necessary in these sensitive areas. Cells that were predominantly water were also given a value of 100 due to the additional costs required to cross a body of water that large.

Lightly developed urban areas were given a value of 20, representing similar concerns with navigating the pipeline route around existing infrastructure as with agricultural areas. More densely developed urban areas were given a value of 200, representing the extreme costs that would be necessary to construct a pipeline through dense urban environments, and to discourage (though not prohibit) routing through these cells based on discussions with pipeline experts.

Once all of the variables had been assigned the values for each cell were added to the base value to give each cell their final cost surface value. The final value represents the relative cost, in percentage terms, of constructing a pipeline through that cell. The following chart illustrates the construction of the cost surface cell values.

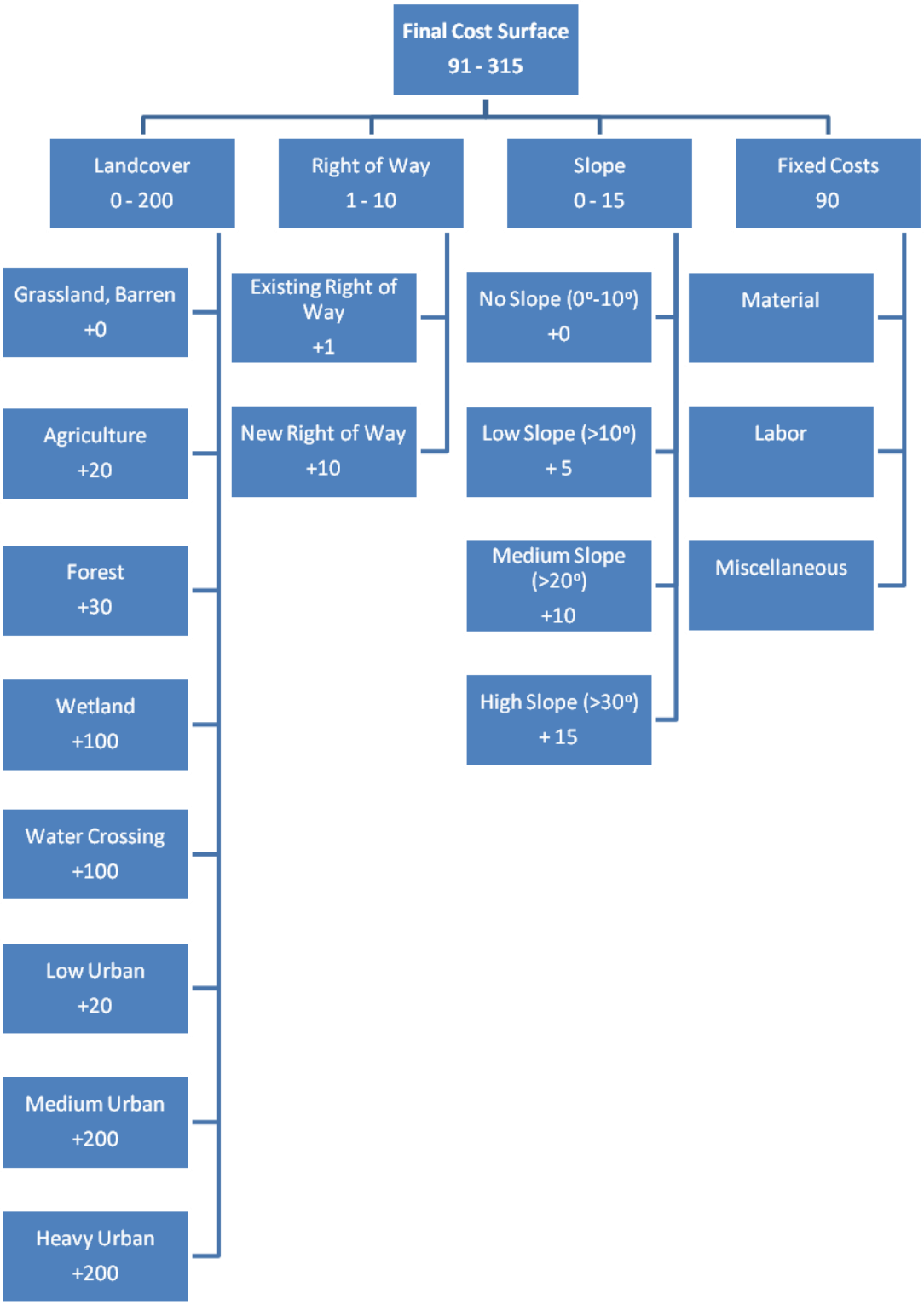

Figure 4: Cost Surface Components

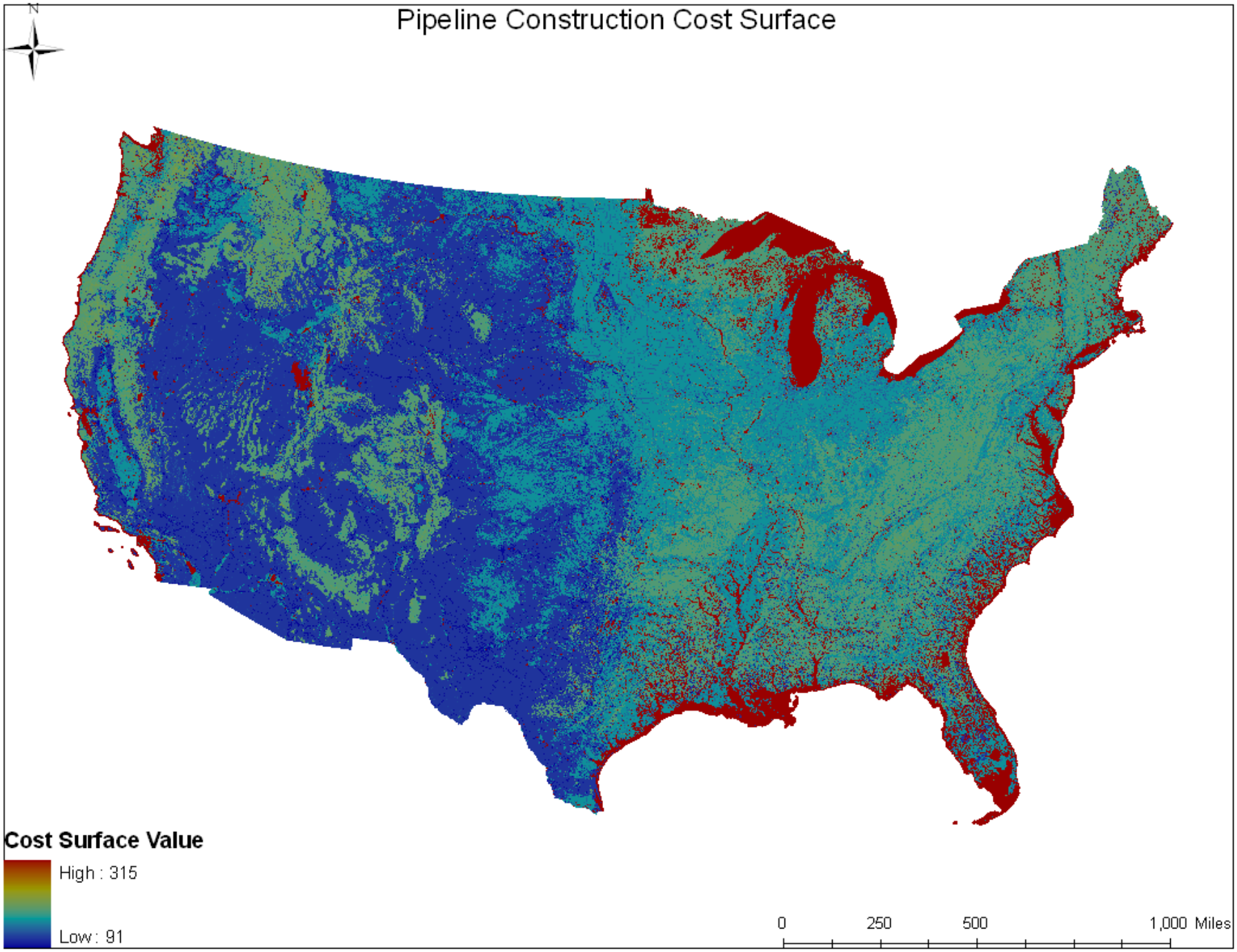

Figure 5: Cost Surface

Pipeline Routing Model:

To link the carbon capture and storage plants to long-term carbon sequestration sites a model was developed in ArcGIS by John Fay at Duke University. The model uses the cost surface described above to connect each plant to a storage site or existing pipeline.

The first step in the pipeline model is to connect some or all of the storage sites to create a trunkline system. The pipelines created in this step do not have directionality since CO_2 can theoretically travel either way along the network to the best storage site, not just the closest. This allows all sources of CO_2 to connect to the trunkline system in addition to the storage sites themselves. This step can be bypassed to examine scenarios without a trunkline system.

The model connects sources of CO_2 to the storage network using a "least cost path" (LCP) analysis. The LCP analysis uses the cost surface described above to find the pathway with the lowest cumulative cost to connect each source to the storage network. The storage network consists of the storage sites themselves, the trunkline, and any other pipelines previously created to connect CO_2 sources to the network. Each CO_2 source is sequentially added to the network through LCP analysis.

Connecting each source in sequence is important because it minimizes the length of the final pipeline network. For example, the cost to link two sites to a storage site might be 100 cost units for each site. If both sites connect directly to the storage site the total cost would be 200 units. However, the cost to connect the two sites is only 20 units, so if the sites coordinate their efforts and connect to the storage site together it will be cheaper. Connecting the two sites together and then connecting to the storage site reduces the total cost to 120 units. As this demonstrates, the method of sequentially adding each CO_2 source to the storage network minimizes the overall cost of the final pipeline network.

Methods Demonstration:

To construct each storage network the first task is to establish the location of each storage site and CO_2 source. Existing plants which retrofit with CCS technology can be directly loaded into the GIS through a spreadsheet. New plants are located using the plant location model, described above, and added to the existing plants to create the final source layer.

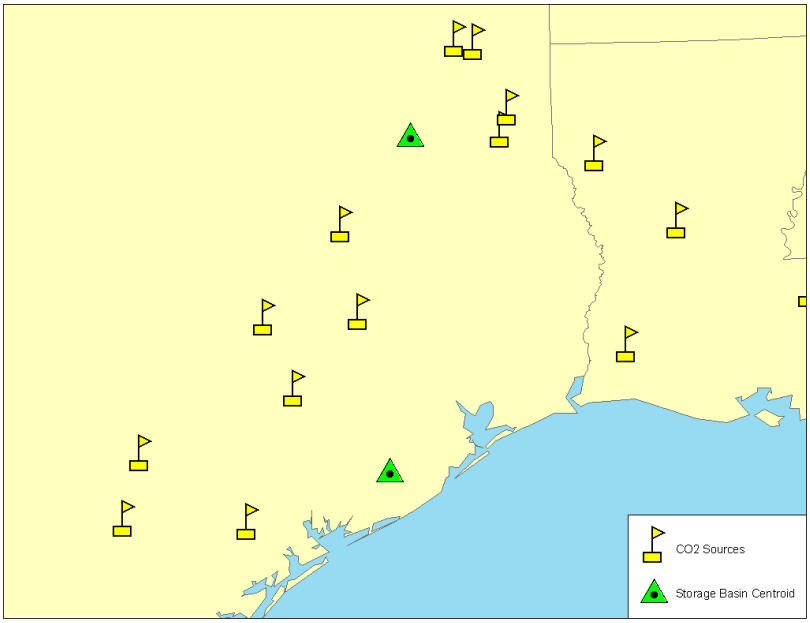

Figure 6: Demonstration Plant and Storage Site Locations

Once the sources and storage sites are located in the GIS the pipeline routing model can be run to connect sources to the storage sites using the cost surface. The pipeline routing model finds the lowest cost path between the closest storage site and the storage network, including previously built pipelines from other plants to storage sites or the trunkline. Figure 7 shows the pipeline network overlaid on a 5 kilometer resolution cost surface, created from the 500 meter cost surface described earlier. The 5 kilometer cost surface produces straighter pipeline routes, which are less costly to construct and may be more reflective of how actual pipelines would be routed.

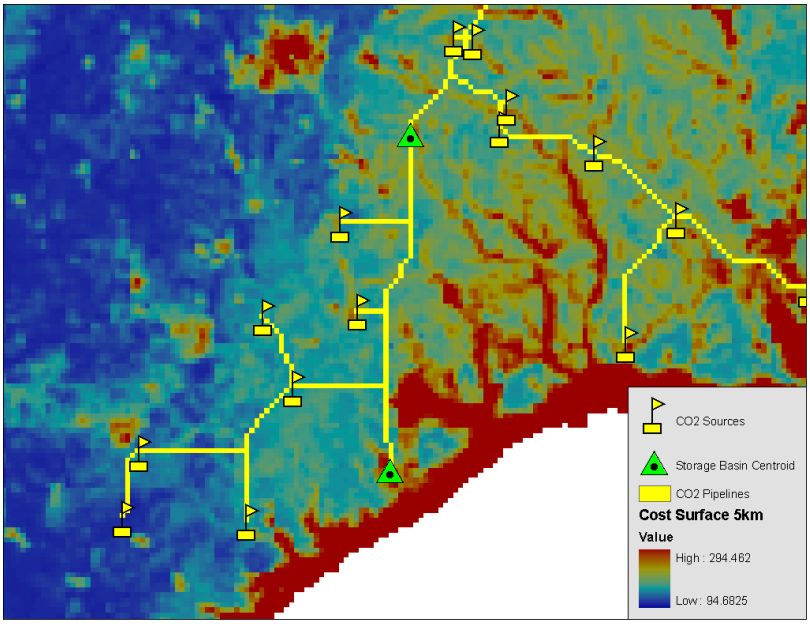

Figure 7: Demonstration Cost Surface and Pipelines

Results:

The CO_2 sources used to create the networks described below are based on the NEMS results of using MEA technology to capture 90% of carbon emissions with a carbon price of $20 per ton beginning in 2012 and increasing at 8% per year. A total of 824 power plant units retrofit with carbon capture technology in this scenario, capturing a total of 165,645,223 tons of CO_2 per year in 2030.[17] Six scenarios are examined, including pipeline networks with no trunkline, a trunkline between storage basins in Texas and Michigan, and a trunkline between storage basins in Texas, Michigan, and New Jersey.

[17] Eric Williams and Munish Chandel, "Modeling the adoption of carbon capture technologies", in development

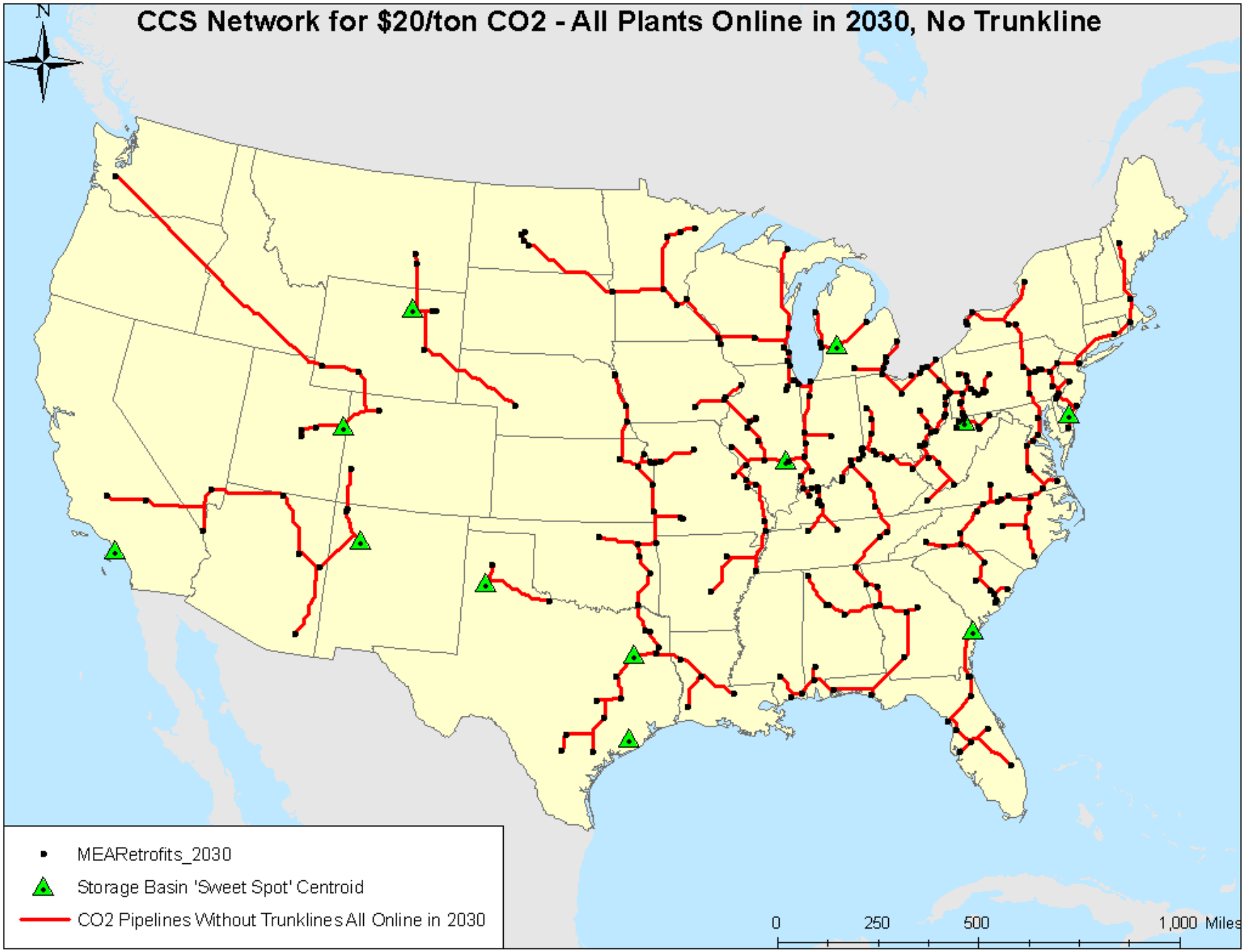

Figure 8: Pipeline Network - No Trunkline, All Plants Online in 2030

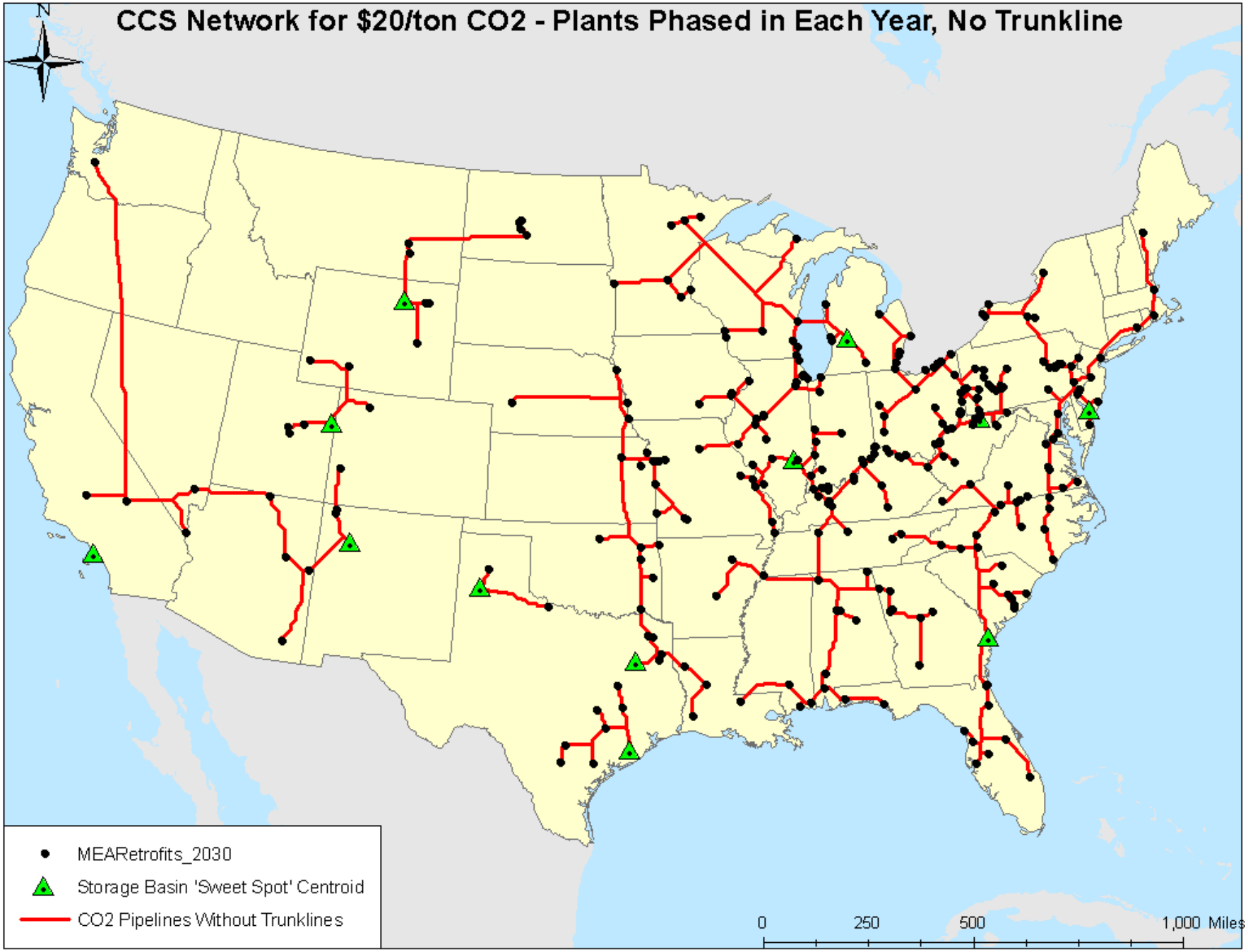

Figure 9: Pipeline Network - No Trunkline, All Plants Phased In

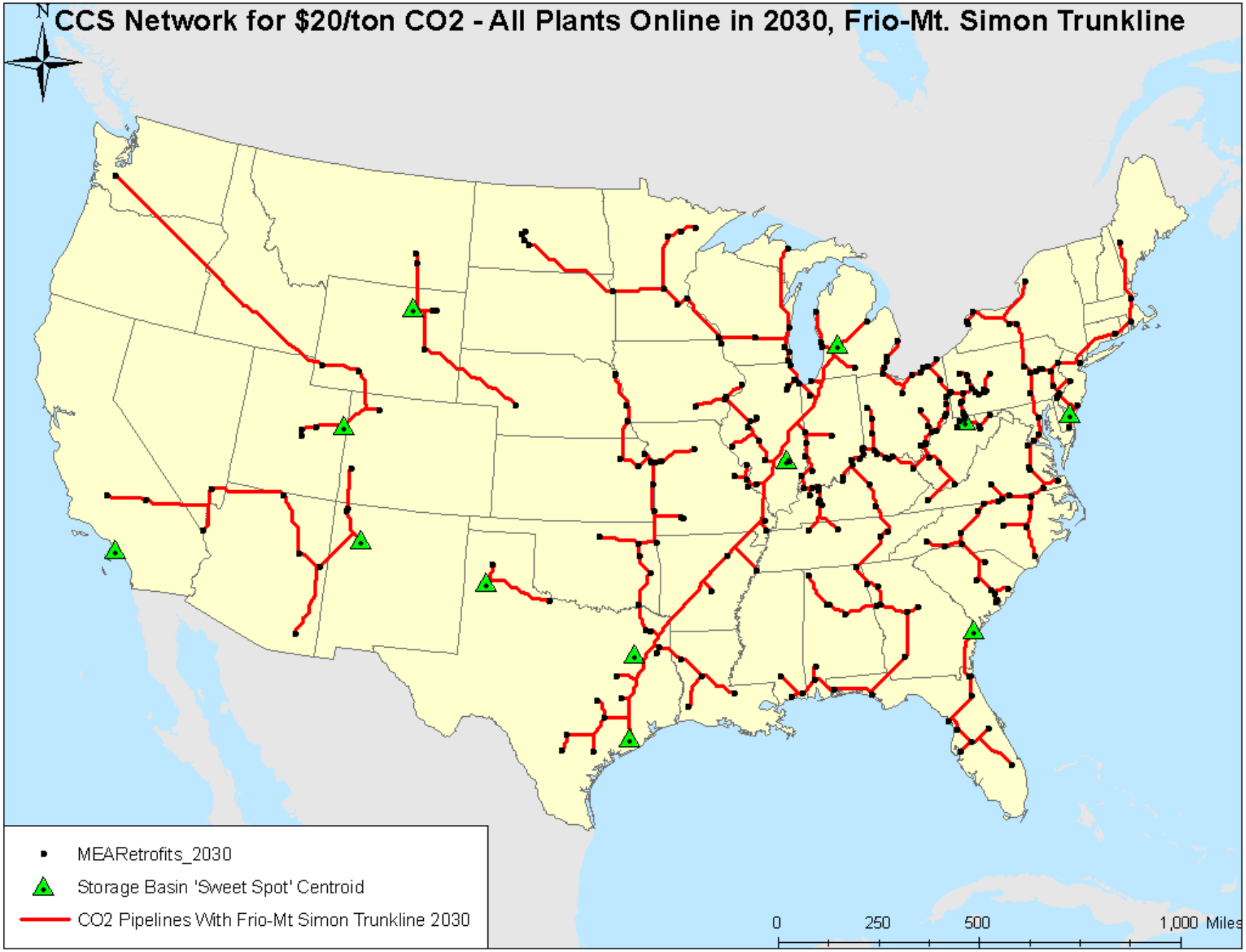

Figure 10: Pipeline Network - Frio-Mt. Simon Trunkline, All Plants Online in 2030

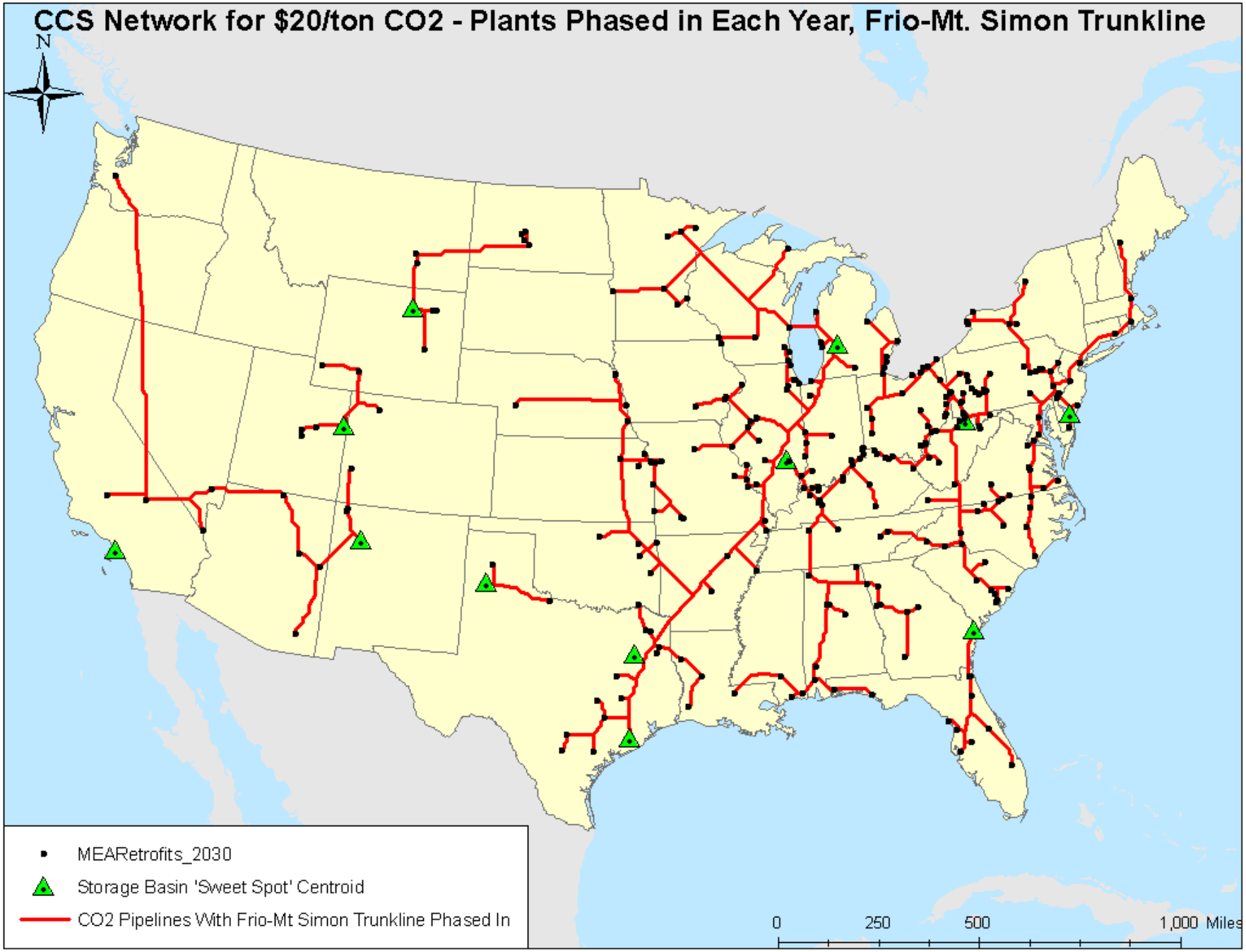

Figure 11: Pipeline Network - Frio-Mt. Simon Trunkline, All Plants Phased In

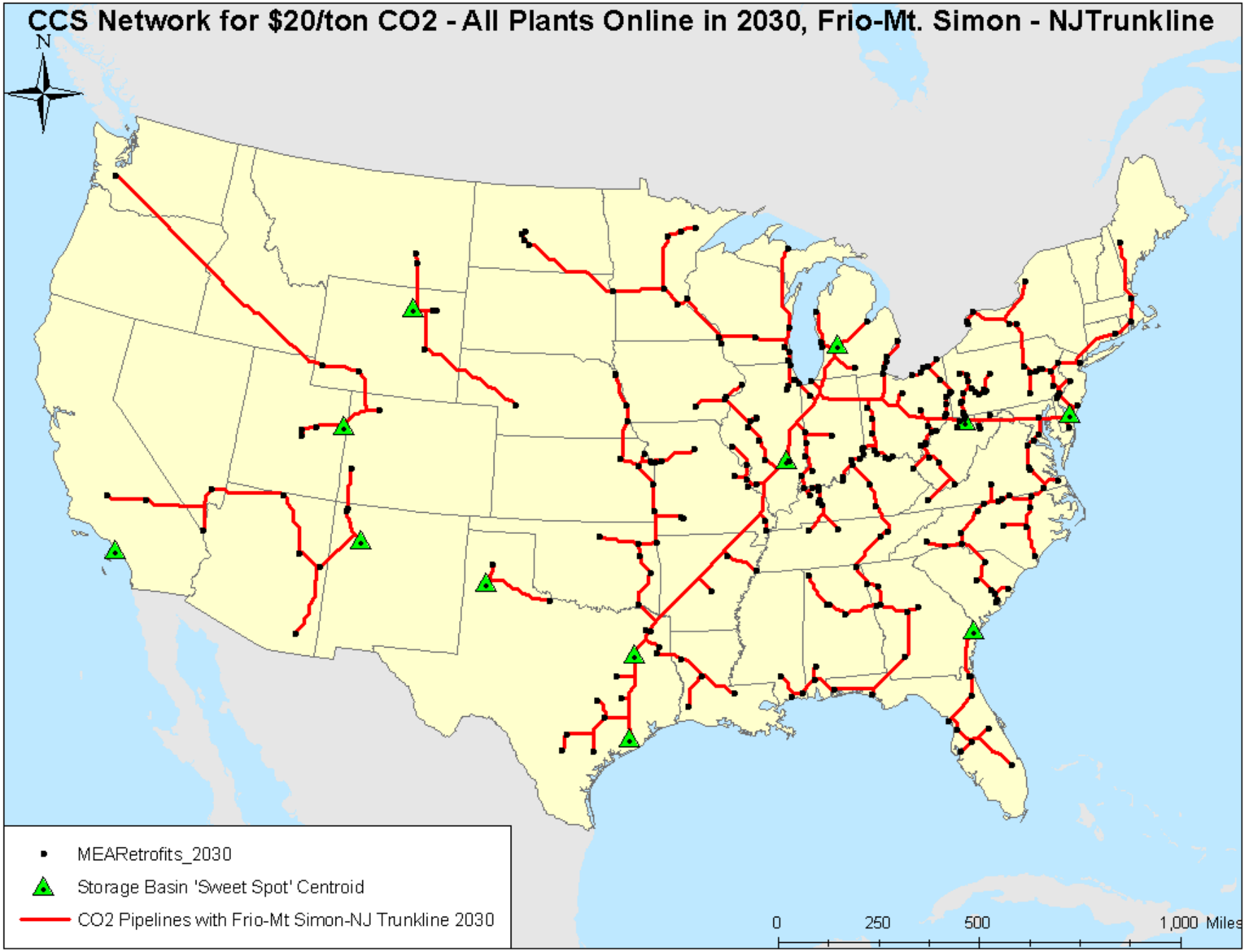

Figure 12: Pipeline Network - Frio-Mt. Simon-NJ Trunkline, All Plants Online in 2030

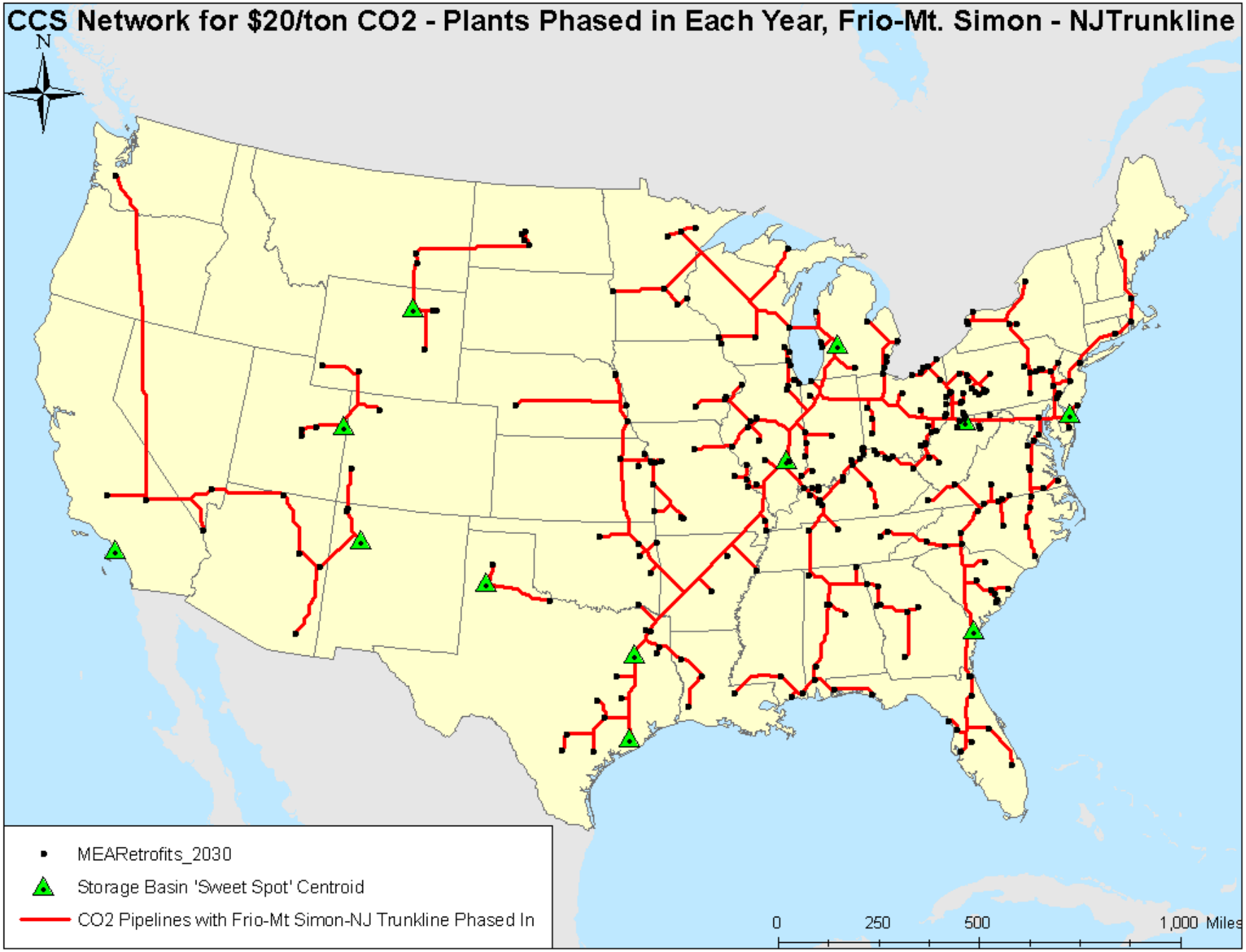

Figure 13: Pipeline Network - Frio-Mt. Simon-NJ Trunkline, All Plants Phased In

Figures 8-13 show that there are many similarities in the networks, especially in the eastern half of the country. There are also some significant differences in the routing of some pipelines. The pipeline routes vary significantly in the western half of the country when the plants are phased in. Pipelines in the southeast also vary considerably when the plants are phased in, and the pipeline between Wisconsin and Michigan is notable when it crosses the great lakes. The presence of trunklines has a large effect on the overall network because many plants connect to them instead of directly to a storage site when trunklines are available.

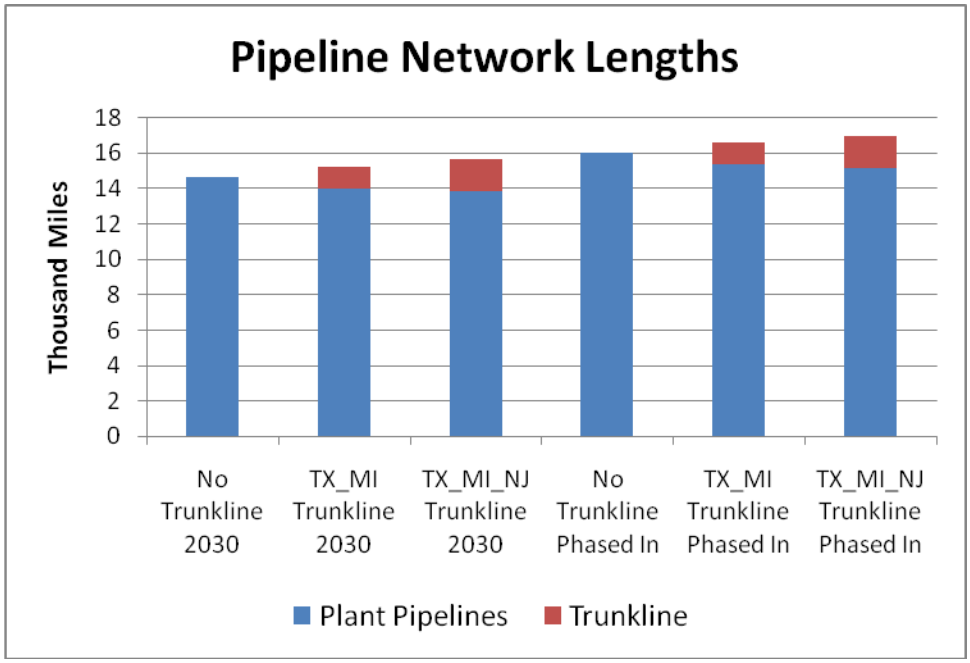

Figure 14: Pipeline network lengths

The length of each pipeline network varies as well, as illustrated in figure 14. The overall pipeline network is longer when a trunkline is constructed compared to when no trunkline is constructed. The network length also increases when the plants are phased in each year. This is as expected since the pipeline routing model is designed to find the most optimal network, and minimizing pipeline length will help minimize construction costs. A significant observation from the data is that the length of pipelines which individual plants are responsible for decreases when

a trunkline is constructed. This raises the question of the cost of constructing the trunkline versus connecting each plant directly to a storage site.

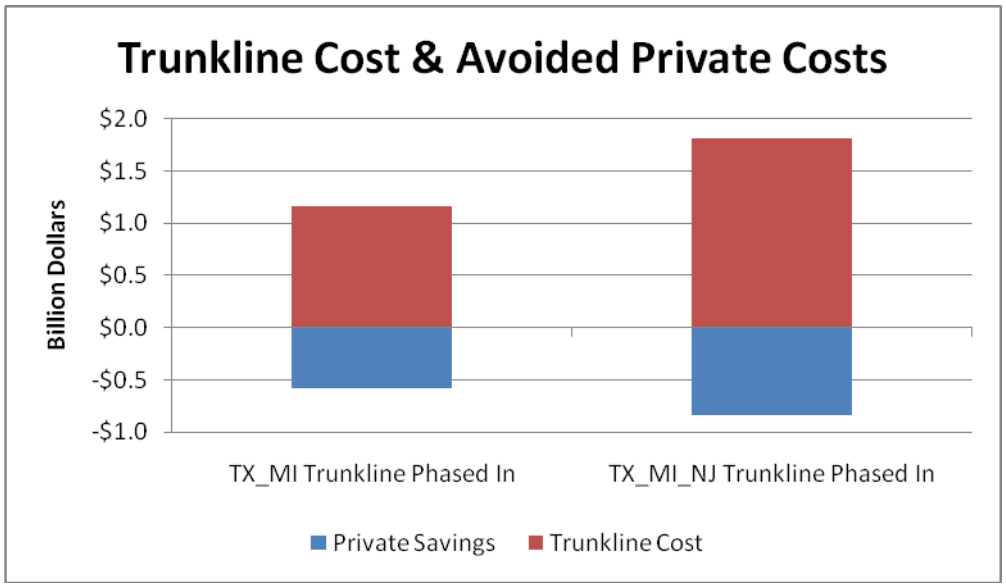

Figure 15: Trunkline costs versus private savings

Figure 15 shows the cost of building the trunkline compared to the avoided costs of individual plants that don't have to connect all the way to a storage site themselves. The cost estimates assume that all of the pipelines are the same diameter. Figure 15 was calculated by multiplying the lengths of the trunklines and the amount of pipeline not built by individual plants by $991,561 per mile, the estimated cost to build a 20 inch diameter pipeline.[18]

As figure 15 indicates, if the cost per mile of pipeline is the same, building a trunkline does reduce the cost to individual plants of connecting to the storage network. However, more money would be spent building the trunkline than would be saved by individual plants. The trunkline's additional benefits besides cost savings to individual plants might justify this expenditure though. Network flexibility and redundancy could provide cost savings in case of an emergency or if a storage basin reaches capacity and can't accept new CO_2 sources.

[18] Eric Williams and Munish Chandel, "Modeling the adoption of carbon capture technologies", in development

The cost of the trunkline, assuming all pipelines are the same diameter, accounts for approximately 10% of the total cost of each pipeline network. The shortest network, if totally constructed of 20 inch diameter pipe, would cost approximately $14,489,718,900 to build. This is a very large infrastructure investment, and all of the other pipeline networks examined in this analysis are more costly than this network. This figure is likely a low estimate because some pipeline segments will have larger diameter pipes to accommodate high flow volumes and thus be more costly to build.

Conclusions:

The pipeline modeling results indicate that building a trunkline increases the overall length of the network but decreases the length of pipelines that need to be built by individual plants. The costs of building a trunkline seems to outweigh the cost savings for individual plants, but the costs are difficult to accurately compare because all the pipelines are assumed to be the same diameter, which is not likely to be the case in reality. The trunkline may need to be multiple parallel pipes of large diameter, and many of the connections from individual plants to the trunkline would likely be multiple pipes or larger diameter to accommodate the captured emissions from upstream plants. Information about the diameter of individual pipeline segments is needed to obtain more accurate cost estimates.

Regardless of whether a trunkline is built or not, a nationwide CO_2 pipeline network will be a huge undertaking. A very large amount of CO_2 will be captured and stored, and the investment needed to construct the transportation network for that carbon will be enormous. The cost estimates presented in this analysis are most likely a significant underestimate of the actual cost of such a network. Policy makers need to appreciate the potential economic impacts of building such a large network when they are debating carbon and CCS related legislation.

Appendices:

Plant Placement Model Script:

```
# New Plant Location Loop
# Description: Uses a search cursor to extract the number of new plants to place in each NERC/Census region.
Places
# the plants in the lowest-cost areas of the 'carbonshed' surface that are within the allowable areas for new plants
# to be built (within 1-mile of the intersection of existing transmission lines and either a railroad or large river).
#
# Created Dec 10, 2008
# Author: Kevin Fritze

import sys, string, os, arcgisscripting
gp = arcgisscripting.create()
gp.overwriteoutput = 1
gp.CheckOutExtension("Spatial")

# Load required toolboxes
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Data Management Tools.tbx")
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Analysis Tools.tbx")

# Input variables
Workspace = sys.argv[1]
ScratchWS = sys.argv[2]
carbonsheds = sys.argv[3]
spatialreffile = sys.argv[4]
plantfile = os.path.join(Workspace, "NEWPLANT.dbf")

# Script Variables
plantsbuff = os.path.join(ScratchWS, "plantsbuff.shp")
boundtemp = os.path.join(ScratchWS, "boundtemp")
boundarearst = os.path.join(ScratchWS, "boundarearst")
boundareaparts = os.path.join(ScratchWS, "boundareaparts.shp")
plantcells = os.path.join(ScratchWS, "plantcells")
plantcellsint = os.path.join(ScratchWS, "plantcellsint")
plantcells1k = os.path.join(ScratchWS, "plantcells1k")
allnewplants = os.path.join(ScratchWS, "allnewplants.shp")

# Prepare output messages
def msg(msgTxt):
   print time.ctime() + " " + msgTxt
   gp.AddMessage(time.ctime() + " " + msgTxt)
   return

# Set spatial reference for new plant files (note: the spatialreffile may need to be altered if ArcGIS is not installed in
this location)
spatialRef = gp.CreateObject("spatialreference")
spatialRef.CreateFromFile(spatialreffile)

# Create file for all new plants
gp.CreateFeatureClass_management(ScratchWS, "allnewplants.shp", "POINT", "", "", "", spatialRef)
```

```
# Open a search cursor
rows = gp.searchcursor(plantfile)
row = rows.next()
region = ""

# Go through each row in the input database file and create the number of plants within the region and time period
specified by the file
while row:
    region = row.Region
    plants = int(row.Plants)
    period = int(row.Period)

    # Loops based on the number of new plants that have to be created, as given by the input dbf file
    for i in range(0, plants):
        gp.AddWarning("Looping through region %s" % (region) + " period %s" % (period) + " plant %s" % (i + 1) +
" of %s" % (plants))
        newplants = os.path.join(ScratchWS, "NewPlants" + region + "_" + "%s" % (period) + ".shp")
        newplantstemp = os.path.join(ScratchWS, "Newplants" + region + "_temp.shp")
        boundarea = os.path.join(Workspace, "PlantRegions.mdb\PlantRegions" + region)
        boundareatemp = os.path.join(ScratchWS, "boundareatemp" + region + ".shp")

        # Uses existing plant layer if already created for that region, and creates a new plant layer if none exists for that
region and time period
        if gp.Exists(newplants):

            # Buffers existing plants (to avoid stacking them on top of each other)
            msg("Buffering plants")
            gp.buffer_analysis(allnewplants, plantsbuff, "30 Miles", "FULL", "ROUND", "ALL")

            # Eliminates the buffered areas from the possible places a new plant can be located
            msg("Erasing buffered plant zones")
            gp.erase(boundarea, plantsbuff, boundareatemp)

            # Extracts the raster cells from the carbonshed cost raster that correspond with areas plants can be placed
            msg("Extracting plant cells")
            gp.ExtractByMask_sa(carbonsheds, boundareatemp, plantcells)

            # Finds the lowest carbonshed cost value
            msg("Finding minimum carbonshed cell value")
            gp.SingleOutputMapAlgebra_sa(plantcells + " * 1000", plantcells1k)
            gp.Int_sa(plantcells1k, plantcellsint)
            mincell = gp.GetRasterProperties_management(plantcellsint, "MINIMUM")
            msg("Minimum cell value = %s" % (mincell)) ##print mincell

            # Extracts the lowest cost value carbonshed cells
            msg("Extracting lowest cost plant areas")
            gp.ExtractByAttributes_sa(plantcellsint, "value <= %s" % (mincell), boundarearst)

            # Converts the carbonshed cells to a polygon to use as the constraining feature class when creating a new
plant
            msg("Converting raster to polygon")
            gp.RasterToPolygon_conversion(boundarearst, boundareaparts)
            msg("Dissolving polygon features into single feature")
            gp.Dissolve_management(boundareaparts, boundareatemp)

            # Creates a new plant location within the lowest cost areas of the carbonshed as defined in the steps above
```

```
        msg("Creating new plant")
        gp.CreateRandomPoints_management(ScratchWS, "NewPlants" + region + "_temp.shp", boundareatemp, "",
1)

        # Appends the new plant to existing plants
        msg("Appending new plants to previously created plants")
        gp.Append_management(newplantstemp, newplants, "NO_TEST")
        gp.Append_management(newplantstemp, allnewplants, "NO_TEST")

    else:

        # Buffers existing plants (to avoid stacking them on top of each other)
        msg("Buffering plants")
        gp.buffer_analysis(allnewplants, plantsbuff, "30 Miles", "FULL", "ROUND", "ALL")

        # Eliminates the buffered areas from the possible places a new plant can be located
        msg("Erasing buffered plant zones")
        gp.erase(boundarea, plantsbuff, boundareatemp)

        # Creates a new feature class to hold the plant points for the region and time period
        msg("Creating new plant file for region %s" % (region))
        gp.CreateFeatureClass_management(ScratchWS, "NewPlants" + region + "_" + "%s" % (period), "POINT",
"", "", "", spatialRef)

        # Extracts the raster cells from the carbonshed cost raster that correspond with areas plants can be placed
        msg("Extracting plant cells")
        gp.ExtractByMask_sa(carbonsheds, boundareatemp, plantcells)

        # Finds the lowest carbonshed cost value
        msg("Finding minimum carbonshed cell value")
        gp.SingleOutputMapAlgebra_sa(plantcells + " * 1000", plantcells1k)
        gp.Int_sa(plantcells1k, plantcellsint)
        mincell = gp.GetRasterProperties_management(plantcellsint, "MINIMUM")
        msg("Minimum cell value = %s" % (mincell)) ##print mincell

        # Finds the lowest carbonshed cost value
        msg("Extracting lowest cost plant areas")
        gp.ExtractByAttributes_sa(plantcellsint, "value <= %s" % (mincell), boundarearst)

        # Converts the carbonshed cells to a polygon to use as the constraining feature class when creating a new
plant
        msg("Converting raster to polygon")
        gp.RasterToPolygon_conversion(boundarearst, boundareaparts)
        msg("Dissolving polygon features into single feature")
        gp.Dissolve_management(boundareaparts, boundareatemp)

        # Creates a new plant location within the lowest cost areas of the carbonshed as defined in the steps above
        msg("Creating new plant")
        gp.CreateRandomPoints_management(ScratchWS, "NewPlants" + region + "_temp.shp", boundareatemp, "",
1)

        # Appends the new plant to existing plants
        msg("Appending new plants to previously created plants")
        gp.Append_management(newplantstemp, newplants, "NO_TEST")
        gp.Append_management(newplantstemp, allnewplants, "NO_TEST")
```

```
    row = rows.next()
```

Pipeline Routing Model Script:

```
## Create Backbone.py
##
## Description: Creates a least cost pathway similating a pipeline from
##              two selected points in a point feature class
##
## Usage: Create Backbone <Sites Feature class> <Inital Backbone> <Cost Surface> <Output Backbone>
##
## Created Oct 2, 2008
## Author: John Fay

import sys, os, arcgisscripting, time, math
gp = arcgisscripting.create()
gp.overwriteoutput = 1
gp.CheckOutExtension("Spatial")

## Input Variables
SitesFC = sys.argv[1]       # Sites to extend pipes to; must have CostDist field
CostFld = sys.argv[2]       # Field to hold marginal cost values
idFld = sys.argv[3]         # Field containing unique IDs
initBB = sys.argv[4]        # Starting backbone pipeline
CostSurf = sys.argv[5]      # Cost surface
outPipeRstr = sys.argv[6]   # Complete pipeline raster datatset
outPipeFC = sys.argv[7]     # Complete pipeline feature class
resetSites = sys.argv[8]    # Boolean to reset CostDist values to zero

## Environment variables
gp.Extent = "MAXOF"
gp.CellSize = "MAXOF"
scratchWS = gp.scratchworkspace
if not scratchWS:
    scratchWS = gp.GetSystemEnvironment("TEMP")


## Script Variables
WorkingBB = os.path.join(scratchWS,'BBTemp')                    # Working backbone file
RemainingSites = os.path.join(scratchWS,'RemainingSites.shp')   # Sites remaining to add
CostDist = os.path.join(scratchWS,'CostDist')                   # Cost distance raster
MaxDist = "                                                     # Cost distance threshold
CostBack = os.path.join(scratchWS,'CostBack')                   # Cost back link raster
CostPts = os.path.join(scratchWS,'CostPoints.shp')              # Sites with cost values
CostPath = os.path.join(scratchWS,'CostPath')                   # Cost path raster
BBUpdate = os.path.join(scratchWS,'BBTemp2')                    # Updated backbone

## Script Functions
def msg(msgTxt):
    print time.ctime() + " " + msgTxt
    gp.AddMessage(time.ctime() + " " + msgTxt)
    return

def wipe_scratch(WS, version = 9.2):
    tmpWS = gp.Workspace
    gp.Workspace = WS
```

```
    if version != 9.3:
        files = []
        tmpFCs = gp.ListFeatureClasses("shape*")
        tmpFC = tmpFCs.next()
        while tmpFC:
            files.append(tmpFC)
            tmpFC = tmpFCs.next()
    else:
        files = gp.ListFeatureClasses("Shape")
    for file in files:
        gp.delete(file)
    gp.workspace = tmpWS
    return 0

def countRecs(fc, version = 9.2):
    if version == 9.2:
        siteCount = int(gp.GetCount(RemainingSites))
    else:
        siteCount = int(gp.GetCount(RemainingSites).getOutput(0))
    return siteCount


try:
    ## Reset cost field, if option is checked
    if resetSites == 'true':
        msg("Zeroing all cost distances in %s" %SitesFC)
        gp.MakeFeatureLayer_management(SitesFC,'lyr','"%s" >= 0' %CostFld)
        gp.CalculateField('lyr',CostFld,"0")
        gp.delete('lyr')

    ## Duplicate initial pipeline
    msg("Copying initial pipeline to %s" %WorkingBB)
    gp.CopyRaster(initBB, WorkingBB)

    ## Extract unprocessed sites from main Site FC
    msg("Extracting unprocessed sites")
    gp.Select_analysis(SitesFC,RemainingSites,'"%s" = 0' %CostFld)
    siteCount = countRecs(SitesFC)
    msg("---------- %d sites remaining to process----------" %siteCount)

    while siteCount > 0:

        ## Create cost distance from existing pipeline
        msg("Starting CostDistance calculation")
        gp.CostDistance_sa(WorkingBB,CostSurf,CostDist,",CostBack)

        ## Find closest site
        msg("Finding closest site to existing pipeline")
        gp.ExtractValuesToPoints_sa(RemainingSites,CostDist,CostPts)
        recs = gp.SearchCursor(CostPts, 'RASTERVALU > 0 AND COSTDIST = 0',",",'RASTERVALU A')
        rec = recs.Next()
        if rec:
            nearest = rec.RASTERVALU          # Distance to nearest record
            siteID = rec.GetValue(idFld)
            msg("Closest point (%d) is %d cost units away" %(siteID, int(nearest)))
            del rec, recs
```

```
        else:
            msg("----All remaining points were found along existing pipelines----")
            del recs
            siteCount = 0
            continue
            #raise "no features"

        ## Create cost path to closest record
        msg("Creating pathway to nearest record")
        exp = '"RASTERVALU" > 0 AND "RASTERVALU" <= %s' %(math.ceil(nearest))
        #msg(">>>Expression is %s" %exp)
        gp.MakeFeatureLayer_management(CostPts,'lyr',exp)
        if int(gp.GetCount('lyr')) == 0: raise "selection error"
        gp.CostPath_sa('lyr',CostDist,CostBack,CostPath,"BEST_SINGLE")
        gp.delete('lyr')

        ## Update cost distance value of selected record
        gp.MakeFeatureLayer_management(SitesFC,'lyr','"%s" = %d' %(idFld, siteID))
        gp.CalculateField('lyr', CostFld, nearest)
        gp.Delete('lyr')

        ## Append new least cost path to existing least cost path
        msg("Appending new pathway to existing pathways")
        exp = 'SetNull((IsNull(%s) and IsNull(%s)), 1)' %(WorkingBB,CostPath)
        gp.SingleOutputMapAlgebra(exp,BBUpdate)
        gp.delete(WorkingBB)
        gp.Rename_management(BBUpdate, WorkingBB)

        ## Update inputs
        gp.Select_analysis(SitesFC,RemainingSites,'"CostDist" = 0')
        siteCount = countRecs(SitesFC)
        if siteCount > 0:
            msg("---------- %d sites remaining to process----------" %siteCount)
        else:
            msg("---------- ALL SITES ADDED ----------")

    ## Copy final working pipeline to saved filename
    msg("Copying pipeline to %s" %outPipeRstr)
    gp.CopyRaster(WorkingBB, outPipeRstr)

    ## Convert final pipeline to FC, if name supplied
    msg("Converting pipeline raster to polyline")
    if outPipeFC != "#":
        gp.RasterToPolyline_conversion(WorkingBB, outPipeFC,'NODATA')

    ## Cleanup
    msg("Cleaning up temporary files")
    tmpData = [WorkingBB,RemainingSites,CostPts,CostDist,CostBack,CostPath]
    for tmp in tmpData:
        if gp.exists(tmp): gp.delete(tmp)
    msg("Deleting scratch files")
    #wipe_scratch(scratchWS)

#---------------------------------------------------------------------------#
except "no features":
    ## Usually happens when remaining points fall on pipeline
```

```
    msg("No sites within the proximity of the pipeline")
    print gp.getmessages()

    ## Copy final working pipeline to saved filename
    msg("Copying pipeline to %s" %outPipeRstr)
    gp.CopyRaster(WorkingBB, outPipeRstr)

    ## Convert final pipeline to FC, if name supplied
    msg("Converting pipeline raster to polyline")
    if outPipeFC != "#":
       gp.RasterToPolyline_conversion(WorkingBB, outPipeFC,'NODATA')

    ## Cleanup
    msg("Cleaning up temporary files")
    tmpData = [WorkingBB,RemainingSites,CostPts,CostDist,CostBack,CostPath]
    for tmp in tmpData:
       if gp.exists(tmp): gp.delete(tmp)

except "selection error":
    msg("Error selecting closest site")
    msg(gp.getmessages())

except:
    gp.AddError("Unhandled exception occurred")
    msg(gp.getmessages())
```