Identifying Critical Pathways in CO_2 Pipeline Routing using a K-shortest Paths with Limited Overlap Algorithm

Authors:

Talsma, Carl J.[1], Duque, Juan C.[1], Prehn, Jonathon[1], Upchurch, Christopher[1], Lim, Seow[2], Middleton, Richard S.[1*]

Affiliations:

[1]Carbon Solutions LLC, 1041 Grand Ave. Unit #142, Saint Paul, Minnesota, USA

[2]Arizona State University, Tempe, Arizona, USA

*** Corresponding Author:**
richard.middleton@carbonsolutionsllc.com

Abstract:

Public perception towards pipeline development and eminent domain concerns around pipeline routing represent a major obstacle in the implementation of carbon capture and sequestration (CCS) infrastructure. Previously, pipeline routing software and literature has focused on finding the least-cost path for routing pipelines. However, these methods often rely on incomplete information and cannot account for social issues such as negotiations with landowners or the complex cost-benefit analysis when considering alternative routing pathways. To address this, we develop and apply the k-shortest paths with limited overlap (kSPwLO) algorithm ESX to pipeline routing scenarios to provide many alternative paths and assess how these paths can inform practical pipeline development concerns. First, we apply the algorithm to a simple case of routing between one emissions source and one sequestration sink while adjusting the input parameter of the ESX algorithm. Next, we apply the algorithm to a complex network of sources and sinks to determine how alternate pipeline routes might impact the overall network structure, and we compare the costs of the alternate structure to the optimal solution provided by the *SimCCS*^{PRO} CCS infrastructure design software. We find that the ESX algorithm efficiently and effectively finds alternate routing paths at a marginal increase in transportation cost (4.5–6.2%) when compared against the optimal route. We also find that many pipeline segments deploy intermittently across different routing simulations, indicating that, at the given cost uncertainty, many different solutions exist and network structure is often robust to changes in pipeline routing. We demonstrate that the ESX algorithm can be used to develop a cost-benefit analysis for choosing between multiple pipeline routes.

Highlights:

- We implement the k-shortest paths with limited overlap algorithm ESX and apply it to the pipeline routing problem to create alternative routing solutions.
- We show that ESX is able to create many different pipeline routes with limited overlap at a small increase in cost relative to the overall CO_2 transport cost, providing optionality in infrastructure development.
- We show how uncertainty in pipeline routing can lead to changes in which CO_2 emissions sources may be optimal for capture in a given network, and we identify which source might be robust to uncertainty.
- We calculate the Pipeline Importance Index to quantify the overall value of routes output by ESX to the overall pipeline network. We show how this metric can help identify physical and social constraints on pipeline routing.

Introduction

According to the Intergovernmental Panel on Climate Change (IPCC), carbon capture, utlizationm and storage (CCUS) is an important technology for the future reduction of atmospheric CO_2 concentrations [1]. Further, negative emissions technologies typically require CCS and are now a key requirement limiting global warming to 2°C and achieving emissions targets by 2050 [2,3]. Governments have begun investing heavily in CCS technology. In the United States (US), CCS is supported under the 45Q tax credit, which offers $85/tonne of CO_2 sequestered underground in saline reservoirs [4]. The significant investment by governments in CCS will result in billions of dollars of infrastructure, including CO_2 pipelines, supporting the new carbon economy of the future. In the US alone, studies have estimated that transport of CO_2 will require 10,000s km of dedicated pipelines [2,5,6].

Early CCUS projects in the US have been met with pushback from the public and other stakeholders regarding the safety and routing of large-scale pipeline projects, even resulting in the cancellation of large CCS projects in the US Midwest [7–9]. Public approval and adoption of CCS and large-scale pipeline infrastructure will be critical for CCS projects to succeed in the US and abroad. Pipeline routing in the early stages of project development, while not representing the largest cost, is disproportionately important to a project's success.

CostMAP^{PRO} was developed to optimally route CO_2 pipelines using a least cost pathway (LCP) approach that allows for custom pipeline routes determined by project priorities and has been widely used to optimally route CO_2 pipelines to simulate infrastructure development [10,11]. *CostMAP*^{PRO} compiles geospatial layers including land cover, slope, population density, existing pipeline corridors, roads, rivers, and topography, among others, and assigns weights based on how these features will impact pipeline costs and routing. While the need for optimally routed pipelines is obvious for improved efficiency and reduced costs, many factors outside of the geospatial data compiled may affect pipeline routing, such as the ability to negotiate with

landowners and stakeholders in a region. Thus, while cost optimality is desirable, optionality in pipeline routing could be critical to determine the most realistic route given "on-the-ground" knowledge and unforeseen issues.

Here, we expand the use case and methods of *CostMAP*^{PRO} to create a network of K-alternate paths that offers both optionality and optimality to improve pipeline routing and offer a more regional view of pipeline routing possibilities. We wrap the *CostMAP*^{PRO} implementation of Dijkstra's shortest path algorithm wtihin the ESX algorithm, a K-shortest paths with limited overlap (kSPwLO) algorithm that produces alternative paths between nodes [12]. We apply the algorithm to compute K-alternate paths while varying the overlap defined by the user input (theta). This allows the identification of not just a single path, but hundreds of possible paths whose construction may be equally feasible given uncertainties. It also allows identification of chokepoints in pipeline construction, and which corridors may be the most important for infrastructure buildouts given a lack of alternative paths. To our knowledge, pipeline routing is a novel application of the k-shortest paths and k-shortest paths with limited overlap algorithms.

Further, we present how uncertainty in pipeline routing may impact the probability of CCS deployment through randomly sampling pipeline configurations achieved through the ESX algorithm and integrating them into *SimCCS*^{PRO}. Previously applied to uncertainty in the deployment of CO_2 sources and sinks [13,14], these results show us how infrastructure development may be impacted if the single optimal pipeline route cannot be achieved, and which solutions may be more or less robust to pipeline routing uncertainty.

Materials and methods

2.1 The CostMAP^{PRO} *and SimCCS*^{PRO} *framework*

We utilize the *CostMAP*^{PRO} software to route pipelines between nodes and to calculate the cost of constructing pipelines [10,11,15]. *CostMAP*^{PRO} utilizes a library of geospatial layers, including land cover, government lands, slope, existing pipeline right-of-ways, roads, rivers, and railways, among others. Each raster layer is assigned both cost and routing weights to either discrete values or a range of continuous values to determine whether the feature geography is expected to increase or decrease a pipeline's cost or routing affinity. This separation of cost and routing weights allows us to separately determine the routing process and the cost calculation process. By doing so, we can incorporate features into the routing process that have routing considerations beyond just the pipeline cost, such as environmental protected areas, regulated lands, or river crossings. Once weights are assigned, we integrate them to determine the routing and cost network that we use to route the pipeline between nodes using a least-cost pathway approach. *CostMAP*^{PRO} utilizes these weights to create a least-cost pathway network where each raster cell has eight connections (edges) corresponding to the eight queen's kernel directions. Each edge weight between cells consists of the average weighted distance between the two cells, with each cell contributing equally to the weight. Diagonal connections have a higher weighted edge by virtue of connecting a longer distance.

The weighted cost and routing networks created by *CostMAP*^{PRO} are next applied to *SimCCS*^{PRO}, which is used to route pipelines between defined CO_2 sources and sequestration sinks [16–18]. *SimCCS*^{PRO} first connects nodes using a Delaunay triangulation, connecting nodes to form triangles such that their circumcircles do not contain any of the nodes [19]. Next, *SimCCS*^{PRO} uses Dijkstra's algorithm[20] to route pipeline connections between each of the nodes connected through a Delaunay neighboring process. Dijkstra's algorithm provides the shortest possible path between nodes as defined by the routing weights network provided by *CostMAP*^{PRO}. The resulting network is called the candidate network from which *SimCCS*^{PRO} utilizes a linear solver to determine which sources, sinks, and pipeline connections minimize the total infrastructure cost at a given volume of CO_2 to capture or, alternatively, which components maximize the captured CO_2 given a cost.

2.2 The ESX Algorithm

The *CostMAP*^{PRO}-*SimCCS*^{PRO} workflow has been a popular means to model both regional and local CCS infrastructure deployment and to optimize network configuration for CCS projects [21]. However, the pipeline routing component of the workflow cannot account for all the complex realities that exist on the ground when trying to route a CO_2 pipeline. Yen's algorithm has long been a popular means of solving the k-shortest path problem [22]. However, due to the density of routing nodes in the network generated by *CostMAP*^{PRO}, Yen's algorithm requires many iterations of Dijkstra's algorithm, such that the problem is intractable for complex pipeline routing. Further, Yen's algorithm does not limit overlap and instead results in solutions that only deviate slightly from the original path. As an alternative approach, we implement the ESX algorithm, which finds k paths by eliminating nodes that are sorted into a priority queue. ESX is found in the literature as an alternative to Yen's algorithm for finding the *kSPwLO* [23]. *kSPwLO* is a common application in determining multiple routing options for road networks. While ESX can produce k-alternate paths, those paths are not necessarily the k-*shortest* paths and the solution resulting from the ESX algorithm should not be considered optimal. However, while the results of ESX are not considered exact, ESX compares favorably against other *kSPwLO* algorithms with regard to speed and scales well for large networks and large values of k [23]. When applied to pipeline routing in *CostMAP*^{PRO}, where we create a dense network of nodes and branches based on raster data, the ESX algorithm becomes appropriate for quickly computing many alternate paths.

ESX works by eliminating branches from the routing network one at a time, determined by the order of nodes in a heuristically-defined priority queue. The priority queue can be determined in different ways based on the needs and priorities of the application. In this paper, we eliminate the largest weighted edge along the previously selected paths and then run Dijkstra's algorithm again on the routing network with the eliminated branch. We chose the largest weighted edge because the large weight suggests that the edge is located in an area with less affinity for pipelines and thus may benefit from alternative routes. The similarity between two paths is determined by the overlap ratio defined below in equation 1:

$$(1) \qquad Sim(p_1, p_2) = \frac{\sum_{\forall (n_i, n_j) \in p_1 \cap p_2} w(n_i, n_j)}{\min\{l(p_1), l(p_2)\}}$$

, where $0 \leq Sim(p_1, p_2) \leq$. $\forall(n_i, n_j)$ is the set of all nodes from node i to node j, and $p_1 \cap p_2$is the set of edges shared between path 1 and 2. $w(n_i, n_j)$ is the set of weights which define the cost of moving from node i to node j. Thus, the numerator is the sum of weights representing the set of edges common to both paths where travel from node i to node j. $l(p_x)$ is the sum of all weights connecting the source and target node or the weighted length of path x. The denominator is always defined as the shorter length path between the two paths being assessed. Thus, the similarity between the paths can simply be described as the ratio of the sum of shared weights between two paths against the minimum total path length between the two paths.

The ESX algorithm can be described by the following steps given a road network G with a set of nodes, N. Additionally, we define the source, s, a target, t, and a set of untouchable nodes, E. $\theta[0,1]$ is the user-defined threshold for allowable overlap between paths. The algorithm returns a set of k alternate paths contained in P_{LO}.

```
//Initialize solution set A, containing the shortest path from source to target using Dijkstra's algorithm
A ← {Dijkstra(G, s, t)};
// Initialize empty set B
B ← {};
// Initialize set of removed edges
E_removed = {};
for k from 1 to k
        // Loop through each node in previous path
        for i from 0 to length(A_{k-1}) − 2
                // Define the spur node as the ith node in path
                N_SN ← n_i ∈ A_{k-1}
                // Define the root path as the set of nodes from 0 to i
                p_root ← n_{0,i} ∈ A_{k-1}
                for each path, p_j in A
                        if p_root == p_j(0 → i)
                            // Remove the links that are part of the previous shortest paths which //share
                            the same root path.
                             G ← G\e(i → i + 1);
                            // Add edge to removed edges
                            E_remove ← E_removed ∪ {e(i → i + 1)};
                for each node in p_root except N_SN
                        // Remove node from network G
                        G ← G\node;
                //Calculate spur path using Dijkstra's algorithm
                p_spur = Dijkstra(G, N_SN, t);
                // Total path is the combination of the root path and spur path
                p_total = p_root + p_spur
                if p_total ∉ B
                        // Add total path to B
                        B ← B ∪ {p_total};
                //Restore removed edges to network G
                G ← G ∪ E_removed
                // Restore nodes in root path to network G
                G ← G ∪ ∀node_j ∈ p_root
        if B is empty
                Break;
        // Sort B based on length of paths
```

$B \leftarrow \{p_i, length(p_i)\}, \forall p_i \in B;$
// Add shortest path in B to solutions set A
$A \leftarrow A \cup \{B_0\};$
// Remove shortest path in B
$B \leftarrow B \backslash B_B;$
return A;

The ESX algorithm starts with the shortest path and then removes the highest-ranked edge in the priority queue. The algorithm then recalculates the shortest path using the chosen shortest path algorithm (i.e., Dijkstra's). If a path cannot be found after the edge is removed, the edge is added to a special set of edges flagged as not removable and the algorithm moves on to removing the next edge in the priority queue, and repeats the process. The new path is compared against the previous path for similarity. If the similarity is above the user-defined value of θ, then the algorithm again removes the next edge in the priority queue and repeats the process until it has found a suitable path.

2.3 ESX Applied to a simple Pipeline Routing Problem

First, we apply the ESX algorithm to find a set of paths between two points, varying the values of k and theta. In our case study, the point chosen as the source corresponds to the Labadie coal plant, west of St Louis, Missouri and the sequestration site resides in the Mt. Simon Sandstone formation in Illinois and shown in Figure 1. While these points are meant to represent a realistic routing scenario, it is somewhat unlikely that Labadie Coal Plant would be a candidate for CCS due to its age and potential for retirement in the near future. However, the route does present interesting obstacles, including population centers, rivers, and established pipeline corridors. Outside of these key routing obstacles, the surrounding region is largely agricultural and topographically flat. A large pipeline (42-inch diameter) also exists running East-West near the Mt. Simon sequestration site, along with a smaller pipeline (26-inch diameter) that spurs off to the South toward St. Louis. These existing pipelines would likely be appealing corridors for routing a new pipeline in the region.These pipelines and other major pipeline right-of-ways are also shown in Figure 1.

We run the ESX algorithm to route multiple paths between the source and sink while adjusting the values of k (the number of routes produced) and the value of theta (the amount of overlap allowed for an alternate route ([0,1]). We assess the quality of the solutions for the application of pipeline routing and identify where major corridors and chokepoints exist across solutions.

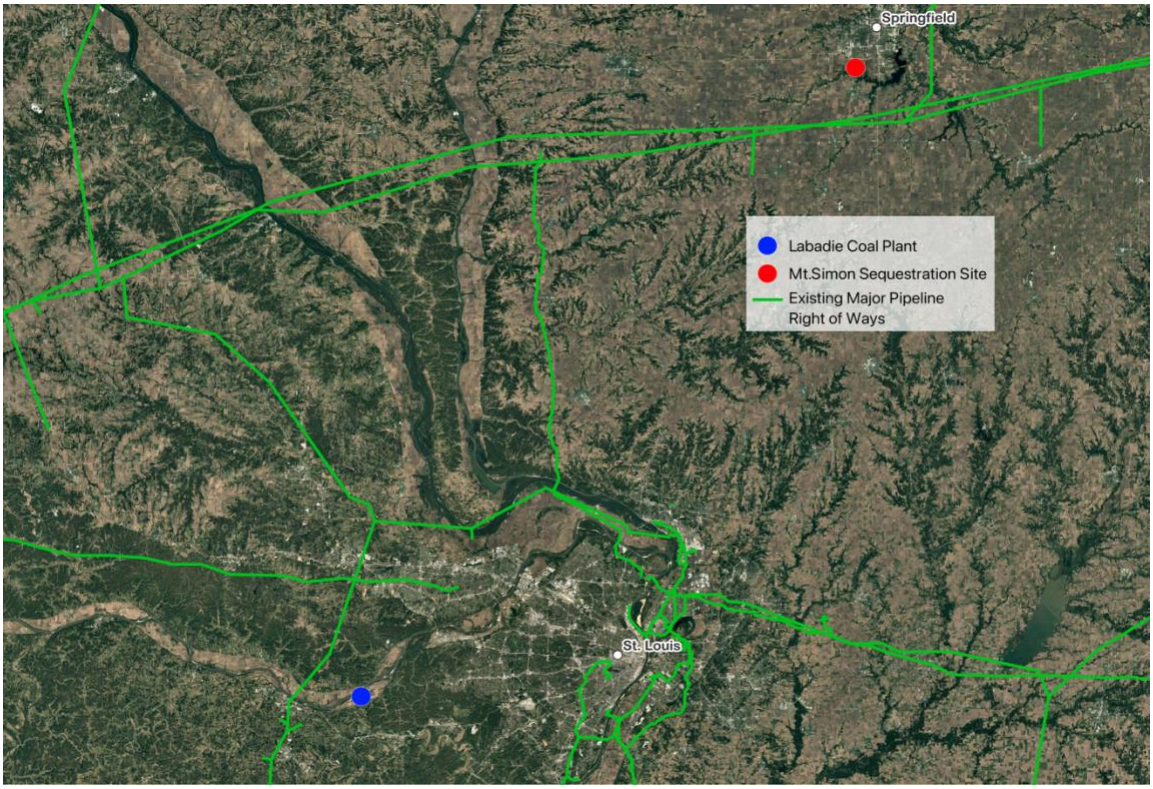

Figure 1: The Labadie coal plant site and Mt. Simon Sandstone sequestration site. Major existing pipelines are shown in green.

2.4 ESX in CCS Infrastructure Design

To test the ESX algorithm's ability to construct a pipeline network consisting of multiple sources and sinks, we construct a set of CO_2 sources and sinks based on the recently-abandoned Heartland Greenway project, which aimed to capture CO_2 from ethanol manufacturing facilities from Iowa and surrounding states and transport CO_2 via pipeline to southern Illinois. The project was recently cancelled due to challenges from local landowners and state governments and an inability to secure routing permits [24,25]. By applying the ESX algorithm to the proposed pipeline network, we can produce a set of alternative paths that could be selected for permitting. Further, the project has a high cost of transport due to the large distances covered and relatively small volumes of CO_2.

The network is composed of 33 CO_2 emissions sources comprised mostly of Ethanol processing plants (except two fertilizer manufacturing plants) across Illinois, Iowa, South Dakota, Nebraska, and Minnesota, and a single sequestration site in Southwest Illinois. According to data from the

Renewable Fuels Association, these plants produce approximately 10.77 millon tonnes of CO_2 annually (10.77 $MtCO_2$/yr). Each ethanol source is given the same capture cost of \$19.12/tCO_2, and the fertilizer sources are given a capture cost of \$16.07/tCO_2. We vary the routes used by the network and determine if uncertainties in the pipeline path result in differences in the optimal sources chosen. Due to the relatively high cost of routing for this network, we expect the optimal solutions to have a greater sensitivity to uncertainties in the pipeline cost than other CCS projects.

First, we construct a Delaunay network to generate pairs of connections between sources and sinks. Then, we run the ESX algorithm between each resulting Delaunay pair and output 10 paths between each pair using a theta value of 0.5 (allowing 50% overlap between paths). To constrain the uncertainty of the paths chosen, we limit the resulting paths to only those that are within a total routing weight of 30% of the shortest possible path. Next, we randomly select a path between each node, resulting in a pipeline network composed of a single selected path between each node. We repeat this step 10 times, resulting in 10 randomly sampled networks. Finally, we run *SimCCS*^{PRO} for each network at varying capture amounts to determine if differences in the randomly selected pipeline routes will result in differences in the optimal sources to capture from. *SimCCS*^{PRO}, when given a set of sources and sinks with capture, transport, and storage costs [17]. By modulating the pipeline networks available to *SimCCS*^{PRO}, we can determine which sources are more or less robust to changes in pipeline routing and we can determine which pipeline routes and chokepoints are most important to the cost effectiveness of the network.

To analyze how important certain pipeline corridors are to the routes output by ESX and the propensity for ESX to use these corridors, we developed the Pipeline Importance Index. First, we determined what routing corridors ESX tended to utilize across multiple routes by calculating the pipeline density (an integer from 0–10 for k=10). However, because we use the highest routing weights to eliminate nodes in the priority queue, areas of high pipeline density are often in areas of low routing weights and thus lower routing importance or consequence. To counteract this, we define the Pipeline Importance Index as the product of pipeline density and the average routing weight for each raster cell, normalized so that pipeline density and routing weights are equally scaled. See equation 2 for calculating the Routing Importance for cell i below, where k is the number of paths determined by ESX:

$$(2) \qquad Pipeline\ Importance\ Index_i = pipeline\ density_i \times \frac{\overline{Routing\ Weight_i} \times k}{Routing\ Weight_{max}}$$

Results

We present the results of running the ESX algorithm at various values of k and theta using the Labadie Coal Plant and the Mt. Simon Sandstone formation as a test case. Figure 2 shows the results of the ESX algorithm for 25 paths at varying theta values from 0.1–0.9. As expected, higher theta values result in smaller deviations from the original shortest path. As the theta value is decreased, a larger variety of paths emerge, including paths of greater length than the original path. Notably, a theta value of 0.2 resulted in longer paths than at a theta value of 0.1. At a theta

value of 0.1, parallel paths tend to develop along the shortest pathway, and the longer paths to the Northwest are no longer present. This is likely because the low theta value is limiting the algorithm from utilizing the shared corridor that travels directly west from the Mt. Simon sequestration site along the identified pipeline route

Figure 3 shows the result of the ESX algorithm at both 100 paths and 25 paths at various values of theta. By increasing the number of paths, we can achieve a similar effect to decreasing the value of theta in that the results show paths with greater geographical variation to the optimal path. Longer paths emerge at higher k values, as well as more local variations in paths when compared to the results of smaller theta values. Comparatively, at 25 paths, we see very little variation between theta values of 0.4 and 0.8. While using a small theta value or large k values can result in similar routes, both options increase the computational time needed for processing. Because of the ESX algorithm formulation, lowering the theta parameter may result in many more iterations of Dijkstra's shortest path algorithm as the results may not meet the stricter overlap threshold imposed. Increasing the k value scales more linearly with computational time and will require the removal of fewer network nodes. However, it is important to note the differences in the paths achieved by both lower theta and increasing k. Increasing k will result in more local deviations from previous routes found, while lowering theta will result in paths with less overlap.

From this analysis, we can identify major chokepoints and points of deviation for the pipeline network. For instance, when routing through the larger St. Louis metropolitan area, the pipeline networks seem to prefer three distinct pathways. One pathway routes along the Missouri River to route between major development and population centers, a second path routes far to the west avoiding any major development, and a third pathway again routes west but finds a route through the development. Even at 100 iterations of k (theta=0.8), these three major pathways persist.

Figure 4 shows the geographic results of running *SimCCS*^{PRO} using different pipeline routes resulting from the ESX algorithm and *CostMAP*^{PRO}. Each panel in the figures represents the output of 10 separate *SimCCS*^{PRO} simulations using randomly selected pipeline routes output from the ESX algorithm using k=10 shown in green. The optimal pipeline routing found using *CostMAP*^{PRO} and *SimCCS*^{PRO} is shown in pink. We will refer to these groups as scenarios, while the 10 iterations of *SimCCS*^{PRO} within each scenario will be referred to as simulations. The sources are represented by pie charts showing the mean capture percentage for that CO_2 source across all of the 10 separate simulations.

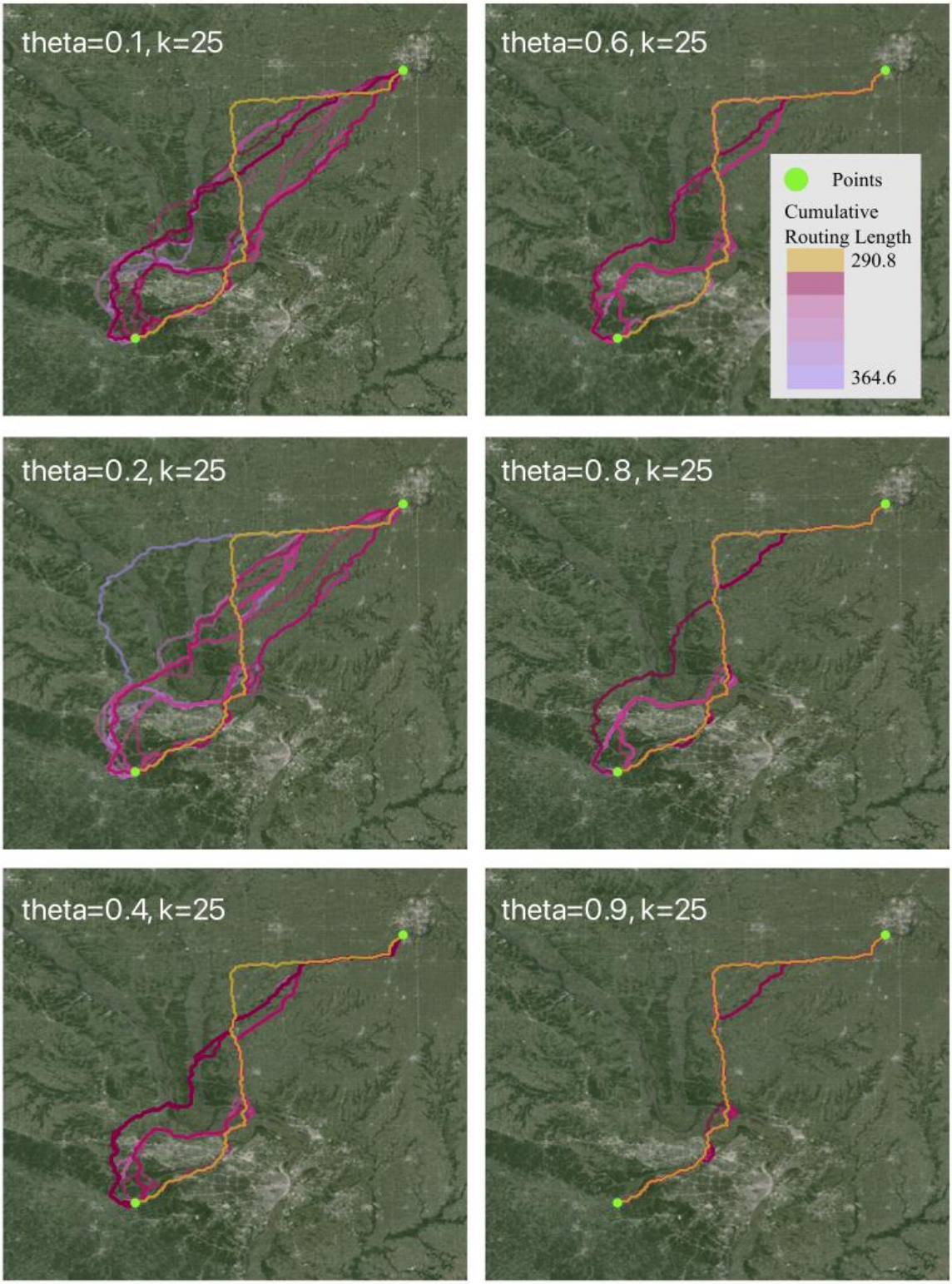

Figure 2: Routing paths using the ESX algorithm at various theta values for k=25 paths. The color of the path corresponds to the cumulative routing length with more purple corresponding to larger routing weights and more red corresponding to smaller weights. The optimal path calculated by Dijkstra's algorithm is shown in gold.

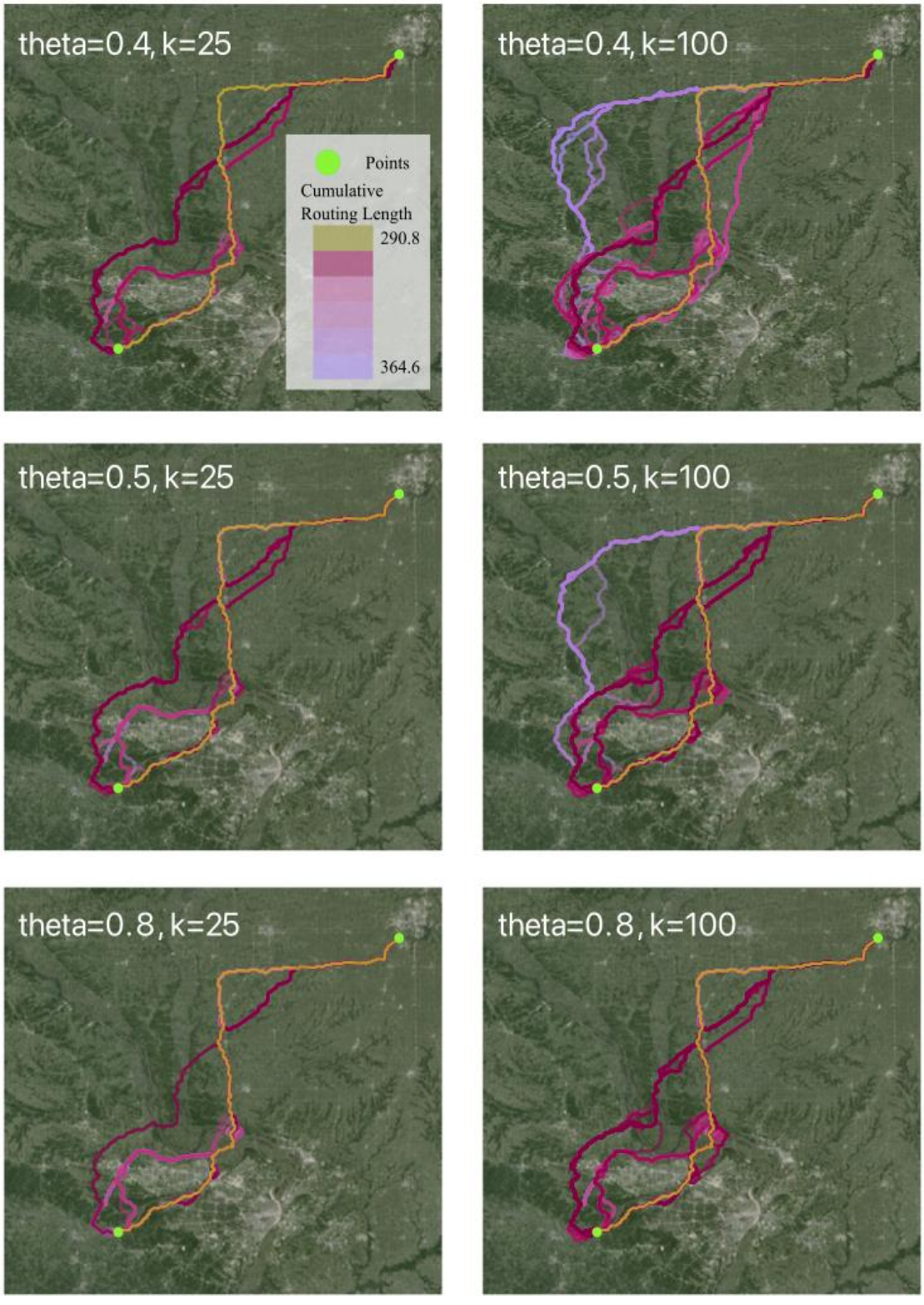

Figure 3: Routing paths at k=100 and k=25 paths at theta values of 0.4, 0.5, and 0.6. The color of the path corresponds to the cumulative routing length with more purple corresponding to larger routing weights and more red corresponding to smaller weights. The optimal path is shown in gold.

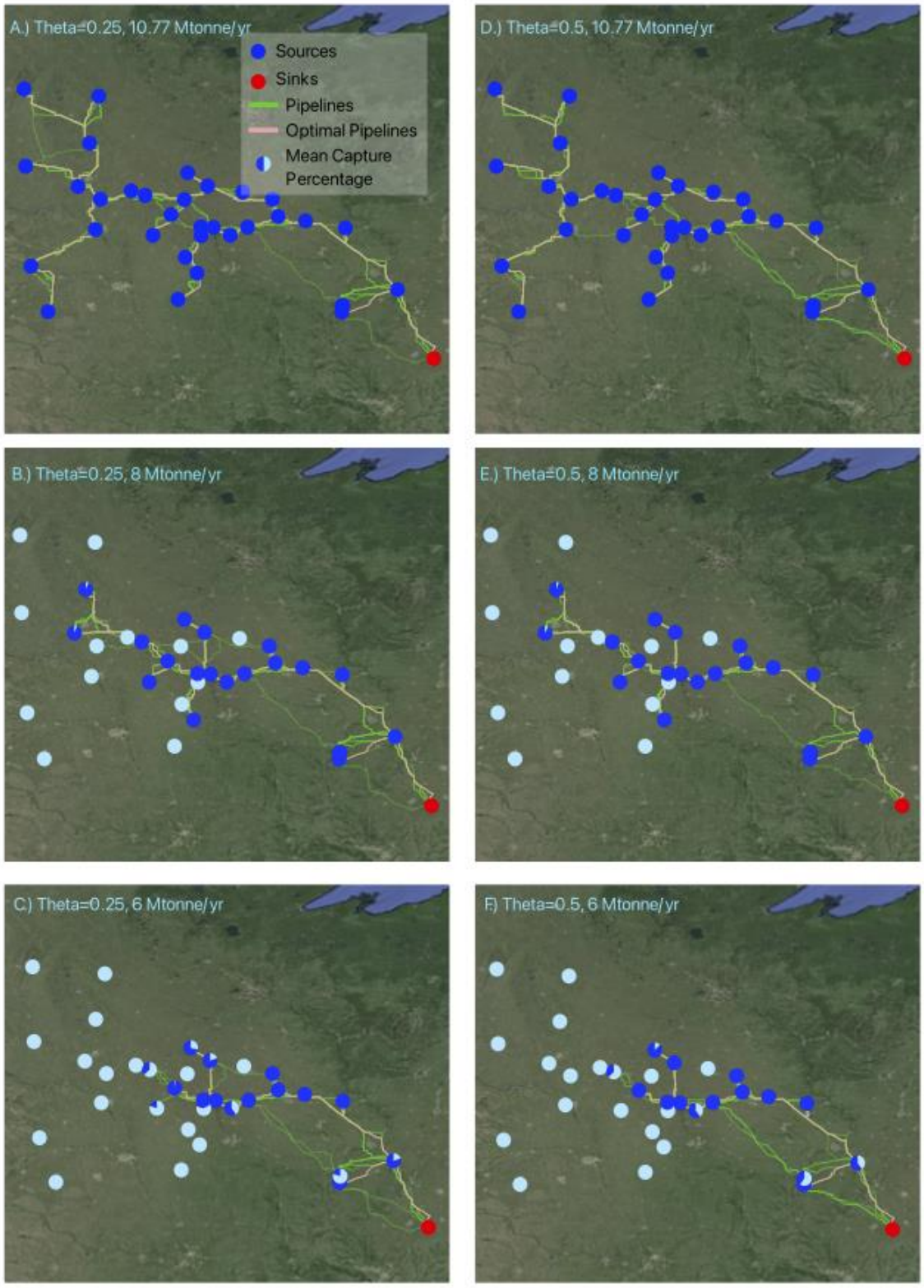

Figure 4: Results of *SimCCS*^{PRO} using the ESX algorithm at various theta values and capture volumes. Each panel shows 10 simulations using pipeline routing output by the ESX algorithm (green) along with the optimal solution routing (pink). The emissions sources are shown as pie charts displaying the mean capture percentage at that source across the 10 simulations for each scenario.

We define a source as having deployed if any of the CO_2 from the source is captured regardless of the actual volume captured. Only the source whose deployment changes between simulations within a given scenario are shown and we omit sources that are always deployed or sources that never deploy. Notably. at a theta value of 0.25 and 6 $MtCO_2$/yr, capture, 8 individual sources experience a change in deployment based on the pipeline routes chosen and 10 sources deploy across every simulation in the scenario. At the same capture volume and using the higher theta value of 0.5, only 4 sources show variable deployment. At higher capture volumes, while the percent of CO_2 captured at a given source may change, the same sources deploy across each simulation within a scenario. This is evident in panel C of Figure 4, where we can see the optimal solution does not extend to the emissions sources furthest from the sink.

	Theta 0.25, 6 Mtonne/year		Theta=0.5, 6 Mtonne/year	
Operator	**Mean Capture %**	**Deployment %**	**Mean Capture %**	**Deployment %**
Big River Resources Galva LLC	80.0	80.0	60.0	60.0
Flint Hills Resources, LLC	20.0	20.0	100.0	100.0
Green Plains Inc.	80.0	80.0	100.0	100.0
POET Biorefining - Gowrie	20.0	20.0	100.0	100.0
POET Biorefining - Jewell	60.0	60.0	60.0	60.0
Valero Renewable Fuels - Hartley	36.6	40.0	37.2	40.0
Valero Renewable Fuels - Welcome	71.3	80.0	100.0	100.0
Big River Resources West Burlington LLC	20.0	20.0	40.0	40.0

Table 1 (above): The mean capture percentage and deployment percentage for CO_2 sources whose deployment was intermittent across simulations at a capture volume of 6 Mtonne/year.

The pie charts shown in Figure 5 represent the share of pipelines within a scenario that deploy at various percentages, excluding pipeline links that never deploy. For example, at a capture volume of 8 $MtCO_2$/yr and theta value of 0.25, a large share of links (~45%) deploy only 10% of the time, while about 25% of links deploy in every simulation. Each scenario shown exhibits a considerable amount of deployment variability, with only a small proportion of pipeline segments deploying across all simulations in each scenario (~15–45% of segments, depending on scenario).

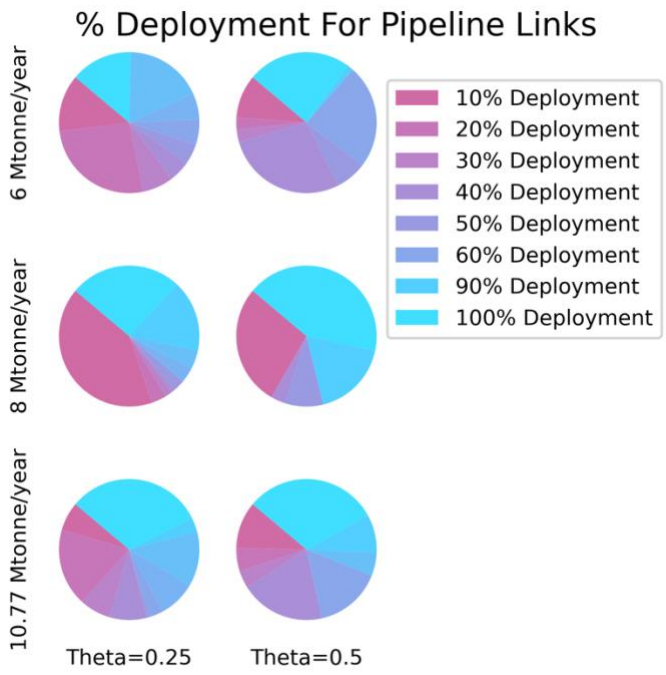

Figure 5: The deployment percentage for pipeline links between nodes excluding links that never deployed for each scenario. 100% deployment signifies that a pipeline was built between the two nodes in every simulation within a scenario, while a 10% deployment signifies that a pipeline was constructed between nodes for only 1 of the 10 simulations within a scenario.

The *SimCCS*^{PRO} results are further described by Figure 6, which show boxplots describing the distribution of cost and pipeline length at various capture volumes and values of theta. Each box within the figures again represents the 10 separate *SimCCS*^{PRO} simulations performed at the given theta value and capture volume, along with the optimal cost and pipeline length found at the same capture volume. The optimal cost and pipeline length found by *SimCCS*^{PRO} is indicated by the pink star in each subplot for comparison. These results show that a higher theta value (i.e., more overlap with the optimal route) resulted in slightly cheaper results overall. Overall pipeline length was roughly similar between differing theta values, with slightly greater variance in pipeline length with a smaller theta value. Both the transport unit cost and the total length of pipeline exhibited relatively small variance. The standard deviation in transport cost for each of the six scenarios ranged from \$0.31-\$0.72/tCO$_2$ (0.84–1.96% of the mean cost), an average of \$1.43–2.15/tCO$_2$ (4.5–6.2%) more than the optimal transportation cost solution, depending on scenario. The standard deviation of pipeline length across each scenario was 2.24–3.00% of the mean pipeline length and, on average, 5.3–6.7% longer than the optimal route.

To determine on a more granular level how the routes output by ESX are constrained by the underlying routing network provided by *CostMAP*^{PRO}, we calculate the Pipeline Importance Index for the 10.77 MtCO$_2$/yr scenario at a theta value of 0.5. The index is able to tell us where ESX is forced to route through an area with a high routing cost due to obstacles in the surface. Figure 7 shows a map of the calculated Pipeline Importance Index for the northernmost portion of the scenario. Two distinct sections of pipeline are shown to have a high Importance Index and inspecting the underlying satellite imagery quickly reveals why these corridors are utilized by all 10 of the routes output by ESX. One of the sections of pipeline is clearly constrained to both the south and north by many lakes. It is likely that few viable routes exist further north because of these lakes, and alternative routes to the south would likely be long enough to make them significantly more costly. The second stretch of pipeline indicated on the map is constrained in different ways. To the north, the pipeline is constrained by higher-density population, and we can see mining operations to the southeast of the pipeline route. Further, the pipeline runs directly to

the south of a highway, which has likely been determined as a corridor by the *CostMAP*^{PRO} routing network.

1 Discussion

Overall, the ESX algorithm provided realistic alternative routes with the capability of varying the level of overlap between routes. By varying the theta value and the k value, we were able to

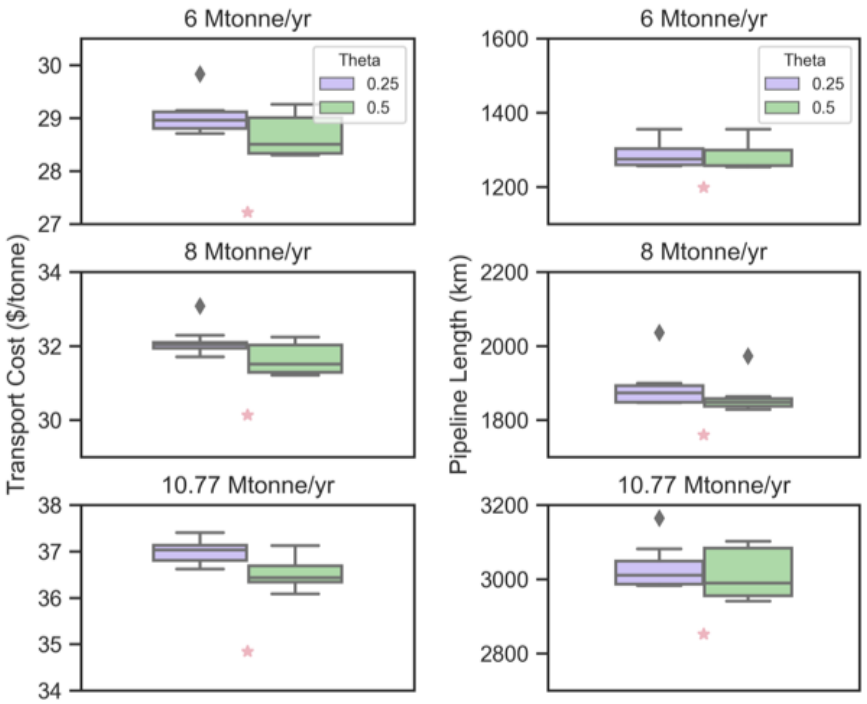

Figure 6: The boxplots show the distribution of Transport Cost ($/tonne) and Pipeline Length (km) across each scenario along with the optimal values (pink stars) for both transport cost and pipeline length.

provide alternative routing at varying scales (local to regional). It should be noted that for the results shown, running Yen's algorithm would have required weeks to run on a Desktop computer. *CostMAP*^{PRO} utilizes a much denser network of nodes when compared to road routing problems and requires a more computationally practical approach. The application of pipeline routing is not as constrained geographically as routing along an existing road network. For this reason, the computational efficiency of the ESX algorithm is preferable when compared to more optimal solutions provided by Yen's algorithm.

The algorithm was able to determine distinct routing pathways through densely developed areas, as in the Labadie-Mt. Simon case. The three routes that emerged through the larger St. Louis suburbs illustrate an important instance of chokepoints and why pipeline routing optionality is important. The shortest cumulative routing length routes along the Missouri River between populated areas. While this option is the optimal route, environmental concerns may force developers to opt for an alternative route. A second option routes entirely around the development but requires a longer pipeline at a greater cost to build. A third option routes through the developed area at an intermediate cost along an existing pipeline right-of-way, but

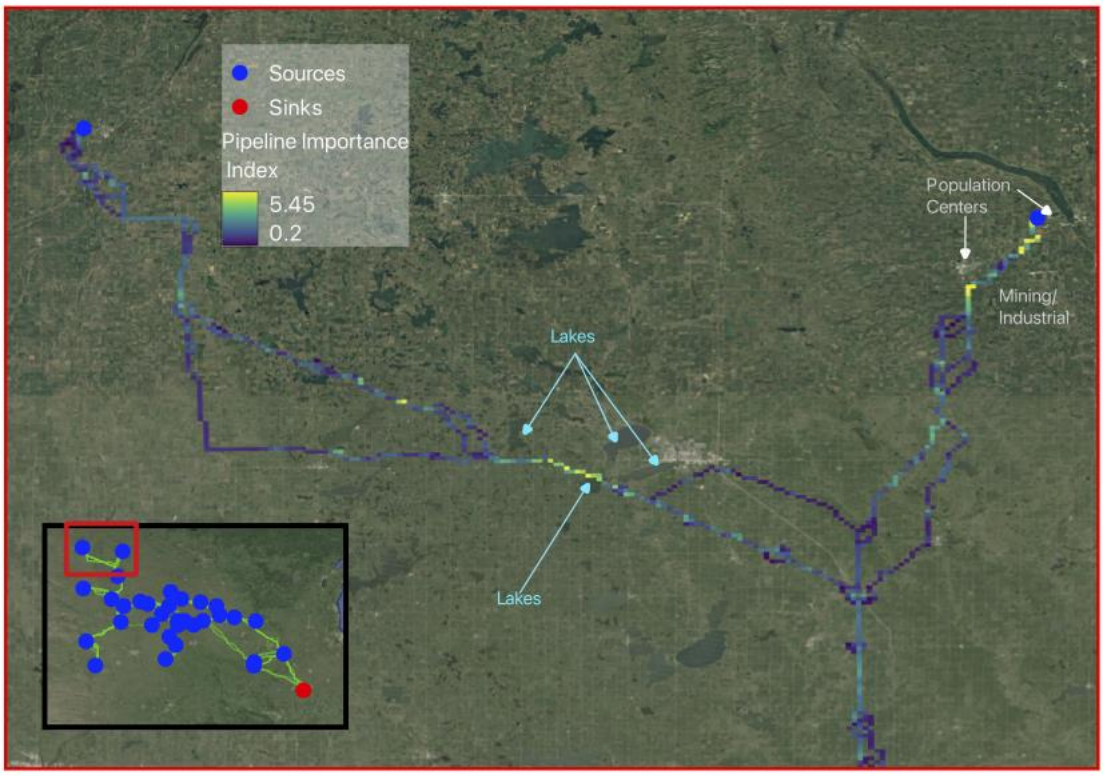

Figure 7: The map shows the calculated Pipeline Importance Index (Equation 2) for the 10.77 Mtonne/yr scenario at a theta value of 0.5. The map shows the sources furthest north within the red inset of the submap shown. Key features along areas of high Importance Index are indicated to show the outing constraints on these areas.

with increased risk to public safety and construction in a densely populated area. Each of the three routing options present unique challenges and benefits and, while only one is deemed optimal, the added optionality provided by the ESX algorithm adds context to build out a cost-benefit framework through which a developer can make more informed decisions.

Eminent domain disputes were a major factor in the cancellation of the Heartland Greenway CO_2 pipeline project. Our results show that extensive alternative routes exist at a similar transport cost. We were able to create alternative pipeline networks with substantial variability in which pipeline connections were deployed and, in some cases, which emissions sources were captured. It should be noted that the high cost of CO_2 transport for this project likely allowed for greater variation in the pipeline network. CCS projects where emissions sources are better co-located with storage typically have lower transport costs (<\$10/tCO_2) and higher capture costs. For these projects, prioritizing emissions sources with low costs is likely more impactful to the cost of the project and is likely a major factor in the final pipeline network. Additionally, pipeline routing in more complex topography, denser development, or a more varied landscape would likely constrain the pipeline routing process more, offering fewer routing alternatives at similar transport costs.

Even the most complex CCS infrastructure design projects benefit from optionality in pipeline routing, as the routing process must be robust to factors such as land-owner negotiations, local rules and regulations, and public opinion, which can change quickly as in the case of the

Heartland Greenway pipeline project. Our results allow us to see early in a project where securing right-of-way might be most important to a project or where pipeline routing alternatives exist. Of particular use in our results is the ability to determine which pipeline connections deploy across every simulation. When planning and negotiating with landowners for large-scale CCS and pipeline projects, these links can be prioritized over those that see intermittent deployment.

Using the ESX algorithm in the context of pipeline routing allows for more informed and localized routing of pipelines, more robust infrastructure networks that are less dependent on securing right-of-ways for pipelines in a given location, and a framework for exploring alternative routes and their relationship to the optimal solution. We are further able to leverage the paths output by ESX by calculating the Pipeline Importance Index to immediately identify areas where the pipeline routes are constrained into choosing high routing weight paths with regularity. Thus, we are able to automate the detection of major chokepoints in pipeline routing between nodes. With this information, CCS infrastructure planning can again prioritize these pathways when negotiating with landowners and increase the chances of success for the project.

Conclusion:

Using the ESX algorithm, we were able to extend the *CostMAP*^{PRO} and *SimCCS*^{PRO} modeling framework to create alternative pipeline networks at similar costs to the optimal solution. Because of the dense network of nodes used in *CostMAP*^{PRO} and the need for quick processing times, the ESX algorithm presents a natural solution for solving the kSPwLO problem when applied to pipeline routing. By manipulating the k and theta values, we can achieve alternative routes on either a more local or regional scale, as shown in our Labadie Coal Plant – Mt. Simon results. and we show key areas where chokepoints and major alternative routes exist.

Our *SimCCS*^{PRO} modeling results show that the kSPwLO algorithms can be a powerful tool for determining pipeline links critical to infrastructure buildout while simultaneously presenting an array of routes. We were able to identify where pipeline routes may be more or less critical to the overall solution provided by *SimCCS*^{PRO}. Further, by calculating many routes early on in a project, we are able to find solutions that may be more robust to the challenges of landowner negotiation and siting. Additionally, we developed the Pipeline Importance Index to identify critical pathways where routing is constrained to using high-cost pathways. Using the Pipeline Importance Index, we are able to automate the detection of major chokepoints that can be considered early on in the pipeline routing process.

Future work is needed to determine how changing the formulation of the priority queue within the ESX algorithm affects the routes found by the algorithm. Here, we chose to use the highest routing weights in the queue because of their disproportionate impact on the routing solution. A more complex formulation for the priority queue could use the input data from *CostMAP*^{PRO}, such as population density to find alternative routes based on more specific routing concerns. Land-ownership, being a large impediment to development, could also be incorporated into the algorithm for finer spatial resolution routing. Future work may focus on less homogeneous areas where more physical constraints to routing may exist, such as more mountainous areas or areas of high-density development.

Acknowledgements

The authors would like to recognize Dr. Sean Yaw and Dr. Brendan Hoover who worked on the original implementation of *CostMAP* and its successive iterations.

Data Availability

The data that support these finding of this study are available on request to the corresponding author, RSM.

References

[1] Intergovernmental Panel on Climate Change (IPCC), Technical Summary, in: Climate Change 2022 – Impacts, Adaptation and Vulnerability, 2023. https://doi.org/10.1017/9781009325844.002.

[2] E. Larson, C. Greig, J. Jenkins, E. Mayfield, A. Pascale, C. Zhang, J. Drossman, R. Williams, S. Pacala, R. Socolow, E. Baik, R. Birdsey, R. Duke, R. Jones, B. Haley, E. Leslie, K. Paustian, A. Swan, Net-Zero America: Potential Pathways, Infrastructure, and Impacts Report, Princeton University (2020).

[3] R.W.J. Edwards, M.A. Celia, Infrastructure to enable deployment of carbon capture, utilization, and storage in the United States, Proc. Natl. Acad. Sci. U. S. A. 115 (2018). https://doi.org/10.1073/pnas.1806504115.

[4] R.A. Esposito, V.A. Kuuskraa, C.G. Rossman, M.M. Corser, Reconsidering CCS in the US fossil-fuel fired electricity industry under section 45Q tax credits, Greenhouse Gases: Science and Technology 9 (2019). https://doi.org/10.1002/ghg.1925.

[5] R. Middleton, J. Bennett, K. Ellett, M. Ford, P. Johnson, E. Middleton, J. Ogland-Hand, C. Talsma, Reaching Zero: Pathways to Decarbonize the US Electricity System with CCS, SSRN Electronic Journal (2022). https://doi.org/10.2139/SSRN.4274085.

[6] E. Middleton, M. Ford, M. Miranda, J. Eidbo, J. Rey Bennett, K. Cox, J.C. Duque, J. Ogland-Hand, C. Talsma, S. Henao, S. Grant, National Industrial Sector Decarbonization, 2024. https://www.carbonsolutionsllc.com/national-industrial-sector-decarbonization-2-2/ (accessed March 17, 2026).

[7] D. Eller, What we know about three carbon capture pipelines proposed in Iowa, Des Moines Register (2021).

[8] D. Eller, Iowans at Capitol pus for stronger restrictions on eminent domain for carbon capture pipeline, Des Moines Register2 (2022).

[9] J. Strong, Carbon pipeline opponents decry sham process, Iowa Capital Dispatch (2022).

[10] C. Talsma, E. Middleton, R. Middleton, Costmappro: Addressing the Massive-Scale Co2 Pipeline Challenge, SSRN Electronic Journal (2022). https://doi.org/10.2139/SSRN.4273192.

[11] B. Hoover, S. Yaw, R. Middleton, CostMAP: an open-source software package for developing cost surfaces using a multi-scale search kernel, International Journal of Geographical Information Science 34 (2020). https://doi.org/10.1080/13658816.2019.1675885.

[12] T. Chondrogiannis, P. Bouros, J. Gamper, U. Leser, D.B. Blumenthal, Finding k-shortest paths with limited overlap, VLDB Journal 29 (2020). https://doi.org/10.1007/s00778-020-00604-x.

[13] R.S. Middleton, S. Yaw, The cost of getting CCS wrong: Uncertainty, infrastructure design, and stranded CO2, International Journal of Greenhouse Gas Control 70 (2018). https://doi.org/10.1016/j.ijggc.2017.12.011.

[14] M.W. Miranda, J.D. Ogland-Hand, J.M. Bielicki, R.G. Moghanloo, J. DaneshFar, R.S. Middleton, Developing a roadmap for carbon capture, and storage in Oklahoma by assessing the viability of stacked storage, Greenhouse Gases: Science and Technology 13 (2023). https://doi.org/10.1002/ghg.2244.

[15] R.S. Middleton, M.J. Kuby, J.M. Bielicki, Generating candidate networks for optimization: The CO 2 capture and storage optimization problem, Comput. Environ. Urban Syst. 36 (2012). https://doi.org/10.1016/j.compenvurbsys.2011.08.002.

[16] R.S. Middleton, J.M. Bielicki, A scalable infrastructure model for carbon capture and storage: SimCCS, Energy Policy 37 (2009). https://doi.org/10.1016/j.enpol.2008.09.049.

[17] R.S. Middleton, S.P. Yaw, B.A. Hoover, K.M. Ellett, SimCCS: An open-source tool for optimizing CO2 capture, transport, and storage infrastructure, Environmental Modelling and Software 124 (2020). https://doi.org/10.1016/j.envsoft.2019.104560.

[18] R.S. Middleton, M.J. Kuby, J.M. Bielicki, Generating candidate networks for optimization: The CO 2 capture and storage optimization problem, Comput. Environ. Urban Syst. 36 (2012). https://doi.org/10.1016/j.compenvurbsys.2011.08.002.

[19] S. Fortune, Voronoi Diagrams and Delaunay Triangulations, (1995) 225–265. https://doi.org/10.1142/9789812831699_0007.

[20] M.A. Javaid, Understanding Dijkstra's Algorithm, SSRN Electronic Journal (2013). https://doi.org/10.2139/SSRN.2340905.

[21] R.S. Middleton, S.P. Yaw, B.A. Hoover, K.M. Ellett, SimCCS: An open-source tool for optimizing CO2 capture, transport, and storage infrastructure, Environmental Modelling and Software 124 (2020). https://doi.org/10.1016/j.envsoft.2019.104560.

[22] A.W. Brander, M.C. Sinclair, A Comparative Study of k-Shortest Path Algorithms, in: Performance Engineering of Computer and Telecommunications Systems, 1996. https://doi.org/10.1007/978-1-4471-1007-1_25.

[23] T. Chondrogiannis, J. Gamper, P. Bouros, U. Leser, Exact and approximate algorithms for finding k-Shortest paths with limited overlap, in: Advances in Database Technology - EDBT, 2017. https://doi.org/10.5441/002/edbt.2017.37.

[24] J. Strong, Navigator CO2 cancels its multistate pipeline project • Iowa Capital Dispatch, (2023). https://iowacapitaldispatch.com/2023/10/20/navigator-co2-cancels-its-multi-state-pipeline-project/ (accessed July 11, 2024).

[25] I. Richardson, Iowa would delay eminent domain process for carbon pipelines until next year, under bill passed by House, Des Moines Register (2022).