A Genetic Algorithm for Multi-Capacity Fixed-Charge Flow Network Design

Caleb Eardley
School of Computing, Montana State University, Bozeman, MT, USA

Dalton Gomez
School of Computing, Montana State University, Bozeman, MT, USA

Ryan Dupuis
School of Computing, Montana State University, Bozeman, MT, USA

Michael Papadopoulos
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA

Sean Yaw
School of Computing, Montana State University, Bozeman, MT, USA

Abstract

The Multi-Capacity Fixed-Charge Network Flow (MC-FCNF) problem, a generalization of the Fixed-Charge Network Flow problem, aims to assign capacities to edges in a flow network such that a target amount of flow can be hosted at minimum cost. The cost model for both problems dictates that the fixed cost of an edge is incurred for any non-zero amount of flow hosted by that edge. This problem naturally arises in many areas including infrastructure design, transportation, telecommunications, and supply chain management. The MC-FCNF problem is NP-Hard, so solving large instances using exact techniques is impractical. This paper presents a genetic algorithm designed to quickly find high-quality flow solutions to the MC-FCNF problem. The genetic algorithm uses a novel solution representation scheme that eliminates the need to repair invalid flow solutions, which is an issue common to many other genetic algorithms for the MC-FCNF problem. The genetic algorithm's utility is demonstrated with an evaluation using real-world CO_2 capture, transportation, and storage infrastructure design data. The evaluation results highlight the genetic algorithm's potential for solving large-scale network design problems.

2012 ACM Subject Classification Applied computing → Transportation; Computing methodologies → Planning and scheduling; Theory of computation → Network flows

Keywords and phrases Fixed-Charge Network Flow, Genetic Algorithm, Matheuristic, Infrastructure Design

Digital Object Identifier 10.4230/OASIcs.ATMOS.2025.10

Related Version *Previous Version*: https://arxiv.org/abs/2411.05798

Funding This research was funded by the U.S. Department of Energy's Fossil Energy Office through the Carbon Utilization and Storage Partnership (CUSP) for the Western USA (Award No. DE-FE0031837) as well as by the U.S. National Science Foundation through the Research Experience for Undergraduates program (Award No. 2243010).

1 Introduction

The Multi-Capacity Fixed-Charge Network Flow (MC-FCNF) problem is a well-studied optimization problem encountered in many domains including infrastructure design, transportation, telecommunications, and supply chain management [16, 26, 27]. In the MC-FCNF problem, each edge in the network has multiple capacities available to it, with each capacity having its own fixed construction and variable utilization costs. The objective of the MC-FCNF problem is to assign capacities to edges in the network such that a target flow

25th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2025).
Editors: Jonas Sauer and Marie Schmidt; Article No. 10; pp. 10:1–10:14

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

amount can be hosted at minimal cost. The MC-FCNF problem is a generalization of the Fixed-Charge Network Flow (FCNF) problem, which has a single capacity (and fixed and variable costs) available per edge. The MC-FCNF problem is NP-Hard to approximate within the natural logarithm of the number of vertices in the graph [27]. As such, finding optimal solutions to large instances is often computationally infeasible.

Significant work has already been done on solving the MC-FCNF and FCNF problems using many techniques including mathematical programming, branch and bound, and exact optimization approaches [14, 15, 21, 10, 8]. Multi-capacity edge networks are often referred to as buy-at-bulk network design problems, and are often framed as facility location problems, which is similar to the MC-FCNF problem but with added demand constraints on sinks [13, 4, 1]. Genetic algorithms have also been introduced for variants of the MCNF problem [9, 2, 28, 25, 12, 30, 29, 19].

In this paper, we introduce a novel genetic algorithm to solve the MC-FCNF problem. The novel contribution of our genetic algorithm is the representation of a flow solution by an array of parameters that scale the fixed-costs for each edge in the network. This representation ensures that each array corresponds to a valid flow, thereby eliminating the need for computationally expensive repair functions that are required by other genetic algorithms for the MC-FCNF problem [9, 2, 25, 17, 30, 29, 19]. By avoiding costly repair functions, the proposed algorithm is able to efficiently find high-quality solutions to very large MC-FCNF problem instances. The proposed genetic algorithm is inspired by slope scaling techniques previously employed for the FCNF problem [14, 7]. It is a *matheuristic*, as it employs mathematical programming to calculate a flow solution from a linear program parameterized with the fixed-cost scaling arrays [11]. Our genetic algorithm is similar to an algorithm proposed by [6], though ours takes a different two-stage approach to handle multi-capacity edges. Additionally, we provide more insight into the existence of the optimal solution in the search space.

An evaluation is presented that designs CO_2 capture and storage (CCS) infrastructure deployments using real-world data composed of thousands of vertices and tens of thousands of edges. In the evaluation, the genetic algorithm is compared to the solution of an optimal integer linear program formulation of the MC-FCNF problem. Results from the evaluation demonstrate the utility of the genetic algorithm for very large networks, even if the solution is very small compared to the full network.

The rest of this paper is organized as follows: Section 2 formally introduces the MC-FCNF program and formulates it as an integer linear program. Section 3 presents a linear programming modification to the integer linear program that serves as the core to the genetic algorithm. Sections 4 and 5 introduce the genetic algorithm and discuss the existence of the optimal solution in the search space. Section 6 presents an evaluation of the genetic algorithm on real-world CCS data and the paper is concluded in Section 7.

2 Problem Formulation

MC-FCNF is formulated as follows: Given a directed graph with vertices V, edges E, a source $s \in V$ with no incoming edges, and a sink $t \in V$ with no outgoing edges, flow must be assigned to the edges such that the target amount of flow T is sent from the source vertex to the sink vertex. There is a set K of the possible capacities for each edge. Each capacity option $k \in K$ for an edge $e \in E$ has a fixed cost a_{ek}, and a variable cost b_{ek}. Assignment of flow to an edge incurs the total fixed cost, as well as the variable cost per unit of flow. The assigned flow must preserve the conservation of flow, meet the target flow amount, and minimize the overall cost.

This problem can also be formulated as an integer linear program (ILP), as shown below:

Instance Input Parameters:

V	Vertex set
E	Directed edge set
K	Set of possible capacities for each edge
$s \in V$	Source vertex with no incoming edges
$t \in V$	Sink vertex with no outgoing edges
c_k	Capacity of k
a_{ek}	Fixed construction cost of edge e with capacity k
b_{ek}	Variable utilization cost of edge e with capacity k
T	Target flow amount

Decision Variables:

$y_{ek} \in \{0,1\}$	Use indicator for edge e with capacity k
$f_{ek} \in \mathbb{R}^{\geq 0}$	Amount of flow on edge e with capacity k

Objective Function:

$$\min \sum_{e \in E} \sum_{k \in K} \left(a_{ek} y_{ek} + b_{ek} f_{ek} \right) \tag{1}$$

Subject to the following constraints:

$$f_{ek} \leq c_k y_{ek}, \forall e \in E, k \in K \tag{2}$$

$$\sum_{k \in K} y_{ek} \leq 1, \forall e \in E \tag{3}$$

$$\sum_{\substack{e \in E: \\ src(e)=v}} \sum_{k \in K} f_{ek} = \sum_{\substack{e' \in E: \\ dest(e')=v}} \sum_{k \in K} f_{e'k}, \forall v \in V \setminus \{s,t\} \tag{4}$$

$$\sum_{\substack{e \in E: \\ src(e)=s}} \sum_{k \in K} f_{ek} = T \tag{5}$$

Where constraint (2) enforces the capacity of each edge and forces y_{ek} to be set to one if f_{ek} is non-zero. Constraint (3) allows at most one capacity to be deployed on each edge. Constraint (4) enforces conservation of flow at each internal vertex. Constraint (5) ensures that the total flow amount meets the target.

Since the MC-FCNF problem is NP-Hard, solving this ILP is intractable for large instances [27]. The objective of this paper is to introduce a novel algorithm that efficiently finds high-quality solutions to this ILP without directly solving it.

3 Non-Integer Linear Program

In this section, we introduce a linear program (LP) that is a modification of the ILP presented in Section 2. Since it is an LP, this new formulation can be solved optimally in polynomial time. This LP forms the foundation of the genetic algorithm discussed in Section 4. Two components of the ILP change to turn it into an appropriate LP:

1. The binary decision variables y_{ek} are removed, thereby turning the model into an LP. Since the y_{ek} variables are removed, the fixed costs are then scaled and combined with the variable costs.
2. A new scaling parameter, d_{ek}, is introduced for each capacity on each edge that will scale the fixed cost of the edge. These parameters form the representation of a flow solution in the genetic algorithm.

Let g_{ek} be the decision variable representing the amount of flow on edge e with capacity k in the LP, analogous to the f_{ek} decision variable in the ILP. Then, the objective function of the LP is:

$$\min \sum_{e \in E} \sum_{k \in K} \left(\frac{a_{ek}}{d_{ek}} + b_{ek} \right) g_{ek} \tag{6}$$

The constraints in the LP mirror the constraints in the ILP:

$$g_{ek} \leq c_k, \forall e \in E, k \in K \tag{7}$$

$$\sum_{\substack{e \in E: \\ src(e)=v}} \sum_{k \in K} g_{ek} = \sum_{\substack{e' \in E: \\ dest(e')=v}} \sum_{k \in K} g_{e'k}, \forall v \in V \setminus \{s, t\} \tag{8}$$

$$\sum_{\substack{e \in E: \\ src(e)=s}} \sum_{k \in K} g_{ek} = T \tag{9}$$

Where constraint (7) enforces the capacity of each edge. Constraint (8) enforces conservation of flow at each internal vertex. Constraint (9) ensures that the total flow amount meets the target.

The output of this LP is a flow value for each g_{ek}. Of course, the optimal flow found by the LP is likely not an optimal solution for the ILP. The true cost of the LP's solution can be determined by calculating its value when input into the ILP's objective function in Equation (1). This is first done by defining an edge-use indicator function z_{ek} and assigning it values as follows:

$$z_{ek} = \begin{cases} 1, & \text{if } g_{ek} > 0 \\ 0, & \text{if } g_{ek} = 0 \end{cases} \tag{10}$$

This makes the true cost of the LP's solution equal to:

$$\sum_{e \in E} \sum_{k \in K} \left(a_{ek} z_{ek} + b_{ek} g_{ek} \right) \tag{11}$$

The genetic algorithm in Section 4 works by varying the d_{ek} scaling parameters and scoring the resulting optimal g_{ek} values found by the LP using Equation (11).

4 Genetic Algorithm

Genetic algorithms are a common evolutionary heuristic method used for searching and optimization. Genetic algorithms manage a population of *organisms* that each correspond to a solution to the problem. The population evolves over iterations of the algorithm using evolutionary processes observed in nature including selection, crossover, and mutation operations. Selection is the process of deciding which organisms of the population continue into the next iteration (i.e., next generation). This is the mechanism that allows the algorithm to prioritize organisms that correspond to better solutions to the problem and control the size of the population. Crossover is the generation of a new organism from two existing organisms, analogous to biological reproduction. Similarly, mutation is the slight modification of an organism into a new one corresponding to a different solution. Crossover and mutation operations are the mechanisms that allow the algorithm to search for new, and possibly better, solutions.

A number of genetic algorithms have been developed to solve various versions of the FCNF problem. In these algorithms, organisms are broadly represented as either individual edges, or predefined routes through the network. Representing organisms as individual edges typically involves a binary variable for each edge indicating its availability for use [9, 29]. Alternatively, representing organisms as predefined routes involves a binary variable for each route in a set of predefined routes through the network [2, 25, 19]. In the case of the individual edge representation, generating the initial population, crossover operations, and mutation operations often requires repairing the organism, as random sets of edges are unlikely to result in valid flows. Using predefined routes simplifies repairing operations, but may still require repair in the event of capacity constraint violations, and is likely to result in sub-optimal solutions due to the limited set of routing options. The genetic algorithms that use these representations address the issue by employing computationally expensive repair functions to make organisms correspond to valid flows. Instead of representing an organism in this fashion, we represent it as an array of the fixed-cost scaling parameters d_{ek} introduced in Section 3. Then, the solution corresponding to this organism is the set of flow values g_{ek} found by the LP from Section 3. The result of this representation is that we can guarantee the solution corresponding to any organism is a valid flow, since the LP enforces that. This avoids costly repair functions and is the key to the efficiency of our approach.

The motivation for using the fixed-cost scaling parameters as the organism representation is that it allows control over the amount of fixed-costs incurred, while also removing the integer variables from the optimal ILP. As the scaling parameter decreases to zero, the scaled fixed-cost increases to infinity, thereby dissuading selection of that edge by the LP. Conversely, as the scaling parameter increases to infinity, the scaled fixed-cost approaches zero, thereby encouraging selection of that edge by the LP. The genetic algorithm is tasked with searching for an organism of scaling parameters whose corresponding flow solution is as close to optimal for the ILP as possible. Given that the genetic algorithm is using the fixed-cost scaling parameters as a proxy for a solution instead of using a solution directly, an important question is whether or not there exists a set of scaling parameters that yields an optimal flow solution. A proof that such a set of scaling parameters is guaranteed to exist is presented in Section 5. The key components and workflow of the genetic algorithm are described below:

Fitness Function

Genetic algorithms use fitness functions to rank and compare the population of organisms to aid the selection process. Our fitness function first determines the flow solution for a given organism by solving the LP in Section 3 to get the g_{ek} flow values. After the g_{ek} values are determined, the solution's true cost in the context of the optimal ILP is calculated by Equation (11). The output of Equation (11) is used as the fitness of the organism, where lower values correspond to higher fitness.

Selection Function

To keep the size of the population computationally manageable, a selection function is employed to prune the population at each iteration. A selection function is also used to identify the organisms for crossover operations. Our genetic algorithm implements a binary tournament selection function in an effort to prioritize high fitness organisms, while not ignoring all low fitness organisms. In binary tournament selection, two random organisms are selected, and the one with the higher fitness is kept, while the other is discarded. This

ensures that high-fitness organisms are likely to remain in the population while maintaining the possibility for low-fitness ones to survive as well. Binary tournament selection is repeated until the number of organisms in the population is at the desired size.

Crossover Function

A crossover function is used to generate a new child organism from two parent organisms already in the population. Our crossover function first randomly selects two parent organisms from the existing population. A child organism is constructed by taking a random interval of the d_{ek} array from the first parent, combined with the remaining values from the second parent.

Mutation Function

In order to mimic evolution and introduce another element of randomness into the search, a mutation function is used to further alter child organisms. After a child organism is generated with the crossover function, it may be randomly selected for mutation. During a mutation operation, a number of d_{ek} values in the organism are selected and, with equal probability, either incremented up or down a random amount between zero and one. Mutated d_{ek} values are not allowed to go below a lower bound to avoid negative values and divide by zero issues.

Genetic Algorithm

Using the functions described above, our algorithm operates as follows: First, an initial population of organisms is randomly generated. Each organism in the initial population is initialized as a d_{ek} array filled with a random value between $\epsilon > 0$ and the average value of the fixed-costs in the input instance. After the initial population of organisms are created, the algorithm proceeds in an iterative fashion: At each iteration, the fitness of each organism is first calculated as described above. While the population size is less than some threshold, the crossover function is executed to generate child organisms. The child organisms are also subject to randomized mutations from the mutation function. Once the population has increased in size to the designated threshold, the selection function is run to reduce its size while statistically discarding the lower fitness organisms. The algorithm keeps executing iterations until the running time reaches a designated time limit.

CPLEX Polishing

Once the time limit has been reached, the most fit organism's g_{ek} flow values are used as a warm start for IBM's CPLEX optimization software. CPLEX polishing is run for one fifth of the total time limit resulting in the final flow values returned by the algorithm.

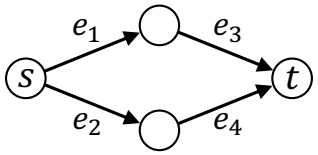
5 Optimal Search Viability

The purpose of this section is to show that the genetic algorithm is capable of finding the optimal solution of the ILP. This claim is not trivial, as the search space of the genetic algorithm is the set of possible d_{ek} arrays, which are merely a proxy for flow values, the property we are actually optimizing for.

The original motivation for formulating the LP in Section 3 and representing the organisms in the genetic algorithm as d_{ek} arrays follows from the following false claim:

▷ Claim 1. If each d_{ek} equals the optimal ILP flow value for edge e with capacity k, then the optimal flow found by the resulting LP is also an optimal flow for the ILP.

The rationale for this claim is that if each d_{ek} and g_{ek} both equal the optimal ILP flow values, then the cost of the LP's objective function from Equation (6) will equal the optimal cost of the ILP's objective function from Equation (1). Claim 1 is also the stated motivation behind other similar genetic algorithms for the FCNF problem [6]. This claim is shown to be false in Figure 1 with the displayed fixed costs (a_e), variable costs (b_e), capacities (c_e), and a capture target of three. In this instance, the optimal ILP solution is to set the amount of flow on e_1 and e_3 to two and the amount of flow on e_2 and e_4 to one for a total cost of 20. Setting d_{e_1} and d_{e_3} to two and d_{e_2} and d_{e_4} to one yields the LP objective of minimizing $7g_{e_1} + 6g_{e_2}$. The optimal solution to this is to set the amount of flow on e_1 and e_3 to one and the amount of flow on e_2 and e_4 to two for a total cost of 21, thereby contradicting Claim 1.



	a_e	b_e	c_e
e_1	10	2	2
e_2	3	3	2
e_3	0	0	2
e_4	0	0	2

Figure 1 Counterexample to Claim 1 with the displayed fixed costs (a_e), variable costs (b_e), capacities (c_e), and a capture target of three. In this instance, the optimal ILP solution is 20 with a flow of two units on e_1 and e_3 and one unit on e_2 and e_4. The corresponding optimal LP solution is 21 with a flow of one unit on e_1 and e_3 and two units on e_2 and e_4.

Since Claim 1 is false, along with the fact that the d_{ek} arrays are only proxies for the flow value solutions we seek, it remains to be shown that there actually exists a set of d_{ek} values that will result in the genetic algorithm finding optimal flow values for the ILP.

▶ **Theorem 1.** *For every problem instance, there exists a set of d_{ek} values such that the optimal flow found by the resulting LP is also an optimal flow for the ILP.*

Proof. Let f_{ek}^{opt} be the optimal flow values found by the ILP and define the set of d_{ek} values as follows:

$$d_{ek} = \begin{cases} \epsilon > 0, & \text{if } f_{ek}^{opt} = 0 \\ \infty, & \text{if } f_{ek}^{opt} > 0 \end{cases} \tag{12}$$

When d_{ek} is set to an ϵ-value near zero, the scaled fixed cost $\frac{a_{ek}}{d_{ek}}$ makes those edges prohibitively expensive to include in LP solutions, so long as valid solutions exist that do not use those edges (such as the valid solution f_{ek}^{opt}). Likewise, if d_{ek} is set to a very large value, the scaled fixed cost is near zero. These d_{ek} values effectively restrict the LP to only selecting the edges and capacities with non-zero f_{ek}^{opt} values.

Suppose that $H \subseteq E \times K$ is the set of edge-capacity pairs where y_{ek}^{opt} equals one. Then, given the d_{ek} values resulting from Equation (12), the objective for the LP becomes:

$$\sum_{e \in E} \sum_{k \in K} \left(\frac{a_{ek}}{d_{ek}} + b_{ek} \right) g_{ek} = \sum_{ek \in H} b_{ek} g_{ek}$$

Let g_{ek}^{opt} be optimal flow values to the LP. We aim to show that g_{ek}^{opt} is also an optimal flow for the ILP. First, g_{ek}^{opt} is a valid solution to the ILP since g_{ek}^{opt} is a valid flow of the ILP's target value on an identical graph with the same capacities. Showing that g_{ek}^{opt} is optimal for the ILP can be accomplished by using g_{ek}^{opt} to feed the definitions for z_{ek} in Equation (10) and showing that:

$$\sum_{e\in E}\sum_{k\in K}\left(a_{ek}z_{ek}^{opt}+b_{ek}g_{ek}^{opt}\right)=\sum_{e\in E}\sum_{k\in K}\left(a_{ek}y_{ek}^{opt}+b_{ek}f_{ek}^{opt}\right)$$

z_{ek}^{opt} must equal one for all edges in H. If z_{ek}^{opt} equals zero for some edge in H, then the fixed costs incurred by g_{ek}^{opt} are lower than the fixed costs incurred by f_{ek}^{opt}. Also, since g_{ek}^{opt} is optimal for the LP,

$$\sum_{ek\in H} b_{ek}g_{ek}^{opt}\leq\sum_{ek\in H} b_{ek}f_{ek}^{opt}$$

Thus, if z_{ek}^{opt} equals zero for some edge in H, g_{ek}^{opt} is a lower cost flow for the ILP than the optimal f_{ek}^{opt}, which is a contradiction. Therefore, f_{ek}^{opt} and g_{ek}^{opt} must incur identical fixed costs and z_{ek}^{opt} must equal one for all edges in H.

Suppose that,

$$\sum_{ek\in H} b_{ek}g_{ek}^{opt}<\sum_{ek\in H} b_{ek}f_{ek}^{opt}$$

This implies that,

$$\begin{aligned}
&\sum_{ek\in H} a_{ek}+\sum_{ek\in H} b_{ek}g_{ek}^{opt}<\sum_{ek\in H} a_{ek}+\sum_{ek\in H} b_{ek}f_{ek}^{opt}\\
\implies&\sum_{ek\in H}\left(a_{ek}+b_{ek}g_{ek}^{opt}\right)<\sum_{ek\in H}\left(a_{ek}+b_{ek}f_{ek}^{opt}\right)\\
\implies&\sum_{e\in E}\sum_{k\in K}\left(a_{ek}z_{ek}^{opt}+b_{ek}g_{ek}^{opt}\right)<\sum_{e\in E}\sum_{k\in K}\left(a_{ek}y_{ek}^{opt}+b_{ek}f_{ek}^{opt}\right)
\end{aligned}$$

which is a contradiction, since f_{ek}^{opt} is an optimal flow value for the ILP, and thus cannot be more expensive than the valid flow g_{ek}^{opt}. Thus,

$$\sum_{ek\in H} b_{ek}g_{ek}^{opt}=\sum_{ek\in H} b_{ek}f_{ek}^{opt}$$

which implies that,

$$\sum_{e\in E}\sum_{k\in K}\left(a_{ek}z_{ek}^{opt}+b_{ek}g_{ek}^{opt}\right)=\sum_{e\in E}\sum_{k\in K}\left(a_{ek}y_{ek}^{opt}+b_{ek}f_{ek}^{opt}\right)$$

Therefore, defining the d_{ek} values as in Equation (12) yields an LP whose optimal flow values correspond to optimal flow values of the ILP. ◀

6 Evaluation

To demonstrate the efficiency and effectiveness of the genetic algorithm presented in Section 4, an evaluation was conducted using CO_2 capture and storage (CCS) infrastructure design data. CCS is a climate change mitigation strategy that involves capturing CO_2 from industrial sources, notably power generation, transporting the CO_2 in a pipeline network, and injecting

it into geological reservoirs for long-term sequestration. Large-scale CCS adoption will require the optimization of infrastructure for hundreds of sources and sinks and thousands of kilometers of pipelines. The CCS infrastructure design problem aims to answer the question: What sources and sinks should be opened, and where should pipelines be deployed (and at what capacity) to process a defined amount of CO_2 at minimum cost. CCS sources and sinks both have fixed construction (or retrofit) costs, variable utilization costs, and capacities (or emission limits). Pipelines have multiple capacities available, depending on the diameter of the pipeline installed. Pipelines also have fixed construction costs and variable transportation costs that are dependent on the capacity selected. Unlike the MC-FCNF problem, the CCS infrastructure design problem has multiple sources and sinks, as well as node-specific costs and capacities. However, CCS infrastructure design instances can be reduced into MC-FCNF instances by introducing a super source and super sink, and by translating node costs and capacities onto edges [24, 27].

The genetic algorithm was implemented and integrated into *SimCCS*, the Java-based CCS infrastructure optimization software, which uses CPLEX as its optimization model solver [20]. Initial performance simulations guided the parameterization of the genetic algorithm to have a population size of 10, and mutation and crossover probability of both 50%. A mutation probability of 50% means that each organism has a 50% chance of mutation, and a crossover probability of 50% means that 50% of the population (without organism repetition) is crossed over with a random other organism in each iteration of the genetic algorithm. All reported genetic algorithm values are the average of three runs. The optimal ILP from Section 2 was implemented in *SimCCS* using CPLEX as well. *SimCCS* was used as a standardized way to represent CCS data and for problem and solution visualization. Timing was coded directly into *SimCCS* to ensure only the algorithm of interest was being timed during simulation. Simulations were run on a machine with Ubuntu 20.04.5, an Intel Xeon W-2255 processor running at 3.7 GHz, and 64 GB of RAM. *SimCCS* on this machine used IBM's CPLEX optimization tool, version 22.1.1.0.

The genetic algorithm was tested on two CCS infrastructure design datasets. The first dataset covers the United State's state of California and consists of 190 sources with a total annual emission rate of 88.39 $MtCO_2$/yr, 102 sinks with a total lifetime storage capacity of 37.18 $GtCO_2$, and 1188 possible pipeline components (i.e., edges in the graph) with a total length of 17940.88 km and 11 possible capacities on each edge. This data was collected as part of the US Department of Energy's (DOE) Carbon Utilization and Storage Partnership project, one of the DOE's Regional Initiatives to Accelerate CCS Deployment. A map of this dataset is presented in Figure 2.

The second dataset covers the contiguous United States and consists of 2746 sources with a total annual emission rate of 532.61 $MtCO_2$/yr, 1202 sinks with a total lifetime storage capacity of 2691.86 $GtCO_2$, and 22597 possible pipeline components with a total length of 424674.41 km and 11 possible capacities on each edge. This data was collected by Carbon Solutions, LLC as part of a study conducted by the Clean Air Task Force [3, 5]. Storage data was generated using the SCO_2T geologic sequestration tool [23]. A map of this dataset is presented in Figure 3.

Candidate pipeline routes were generated in *SimCCS* using its candidate network generation algorithms [31]. The National Energy Technology Laboratory's CO_2 Transport Cost Model was used by *SimCCS* to determine fixed construction and variable utilization costs for the 11 discrete pipeline capacity options [22].

To assess the efficiency of the genetic algorithm, its solution cost was compared to the solution cost found by CPLEX solving the optimal ILP, with both methods being allowed to run for set running time periods. For the California dataset, those running time periods

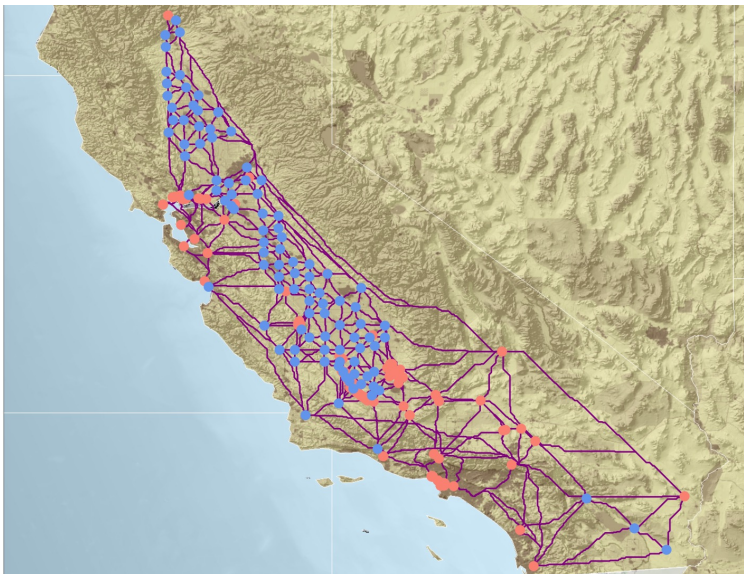


Figure 2 CCS dataset for the state of California consisting of sources (red), sinks (blue), and possible pipeline routes.

were 0.5, 1, 2, 4, and 8 hours. The target flow amount (T) was set to 80 $MtCO_2/yr$ for all of the California scenarios. For the contiguous United States dataset, the running time periods were 0.5, 1, 2, 4, 8 and 16 hours. The target flow amount was set to 500 $MtCO_2/yr$ for the contiguous United States scenarios. Figure 4 presents each algorithm's solution cost over the running time periods for the California dataset, and Figure 5 presents the same results for the contiguous United States dataset. The cost of the best solution found by the genetic algorithm in the California dataset was within 0.5% of the ILP's solution across all running times. Conversely, the cost of the best solution found by the genetic algorithm in the contiguous United States dataset was 17% lower than the ILP's solution after one hour, 7% lower after four hours, and 2% lower after 16 hours. This suggests that the genetic algorithm may have utility for very large problem instances. The utility for small instances is likely limited, due to the speed of CPLEX. Further, in applications that require rapid computation of MC-FCNF solutions, the genetic algorithm may exhibit beneficial performance for even smaller instances.

Problem solvability is not only related to instance size, but also the amount of target flow being found. To identify the impact that target flow amount has on solution quality, scenarios were run on the contiguous United States dataset where the target flow amount was varied from 1 $MtCO_2/yr$ to 532 $MtCO_2/yr$. The maximum annual capturable amount of CO_2 for this dataset is 532.61 $MtCO_2/yr$. Each algorithm was given two hours to solve each scenario. Figure 6 presents each algorithm's solution cost for the various target flow amounts. Table 1 presents the specific solution cost values and the percent improvement of the genetic algorithm's solution over the ILP's solution. Other than in very low and high target flow amount scenarios, the genetic algorithm was fairly consistent in its improvement over the ILP. Interestingly, in the low target flow amount scenario, the genetic algorithm performed the best relative to the ILP. This suggests that the genetic algorithm could have utility in a very large problem instance, even if the target flow amount is quite small. This is likely a very realistic scenario, where a relatively small target is sought amongst a very large space of options, and provides a compelling argument for the utility of the genetic algorithm.

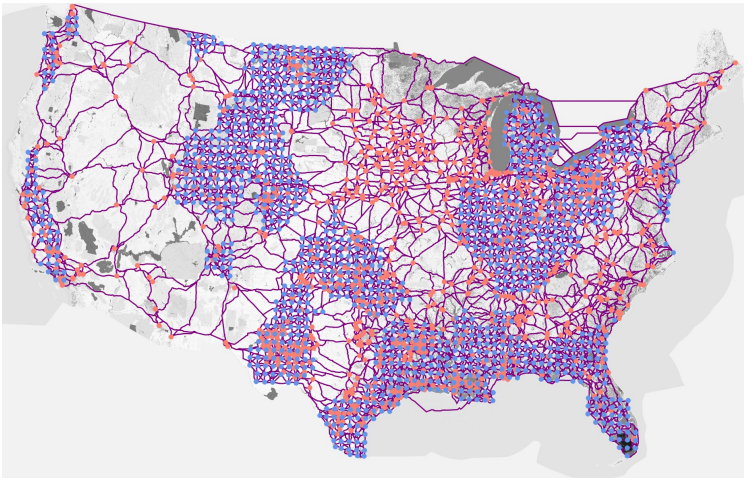


Figure 3 CCS dataset for the contiguous United States consisting of sources (red), sinks (blue), and possible pipeline routes.

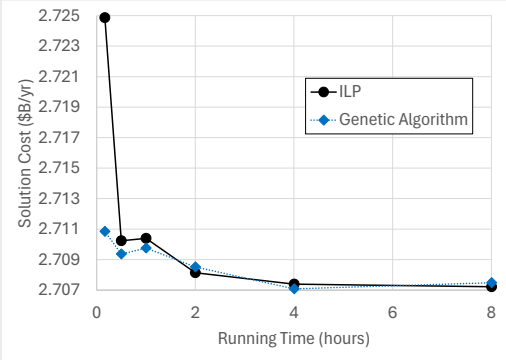


Figure 4 Solution cost versus running time for the genetic algorithm and optimal ILP on the California dataset.

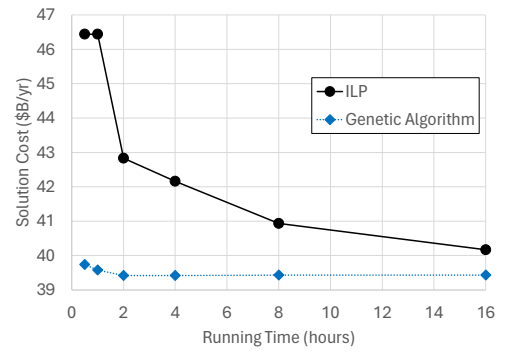


Figure 5 Solution cost versus running time for the genetic algorithm and optimal ILP on the contiguous United States dataset.

7 Conclusion

In this paper, we addressed the MC-FCNF problem by formulating it as an ILP and proposing a novel genetic algorithm to find high-quality solutions efficiently, using a relaxation of the ILP to an LP to ensure the solutions of the GA are valid solutions and do not need repairing. The key novel component of our approach is the use of fixed-cost scaling parameters as a proxy for direct flow values, allowing the genetic algorithm to search the solution space effectively without the need for computationally expensive repair functions.

Our genetic algorithm demonstrated significant efficiency and effectiveness in solving the MC-FCNF problem. By integrating the algorithm into the *SimCCS* infrastructure optimization software, we were able to evaluate its performance on real-world CCS infrastructure design data. The results showed that the genetic algorithm consistently outperformed CPLEX solving an ILP on very large problem instances, and matched CPLEX's performance on moderately sized problem instances. The evaluation demonstrated the potential of the genetic algorithm in handling large and complex networks with varied target flow objectives, across a wide range of running time requirements.

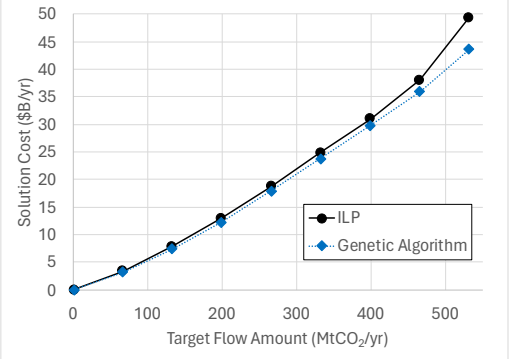


Figure 6 Solution cost versus target flow amount for the genetic algorithm and optimal ILP. Specific solution cost values are presented in Table 1.

Table 1 Solution Values With Varying Flow Targets

Target Flow Amount	ILP	Genetic Algorithm	% Improvement
1.00	0.04	0.03	25.00
66.50	3.35	3.22	3.88
133.00	7.84	7.41	5.48
199.50	13.00	12.25	5.77
266.00	18.78	17.94	4.47
332.50	24.84	23.77	4.31
399.00	31.06	29.86	3.86
465.50	38.05	35.98	5.44
532.00	49.44	43.58	11.65

The genetic algorithm presented in this paper offers a robust and scalable solution to the MC-FCNF problem, providing an efficient alternative to traditional ILP solvers in realistic scenarios. Future work could involve tailoring the genetic algorithm to more specialized versions of the FCNF problem, including phased network deployments [18]. The genetic algorithm could also be generalized to multi-commodity fixed-charge network design problems. The fixed-cost scaling parameter technique may also prove useful in building evolutionary algorithms for other problems that can be modeled as network flow problems (e.g., facility location problems). Future work could include implementing this genetic algorithm approach for use in facility location applications and testing it against benchmark datasets. Future work could also include pursuing real-time applications where low computational running time is more critical than infrastructure design problems. Finally, further exploring the performance of the genetic algorithm on very large instances with small target flows could reveal useful applications of the genetic algorithm to real-world problems.

References

1 Ashwin Arulselvan, Mohsen Rezapour, and Wolfgang A. Welz. Exact Approaches for Designing Multifacility Buy-at-Bulk Networks. *INFORMS Journal on Computing*, 29(4):597–611, November 2017. doi:10.1287/IJOC.2017.0752.

2 Huynh Thi Thanh Binh and Son Hong Ngo. Survivable flows routing in large scale network design using genetic algorithm. In *Advances in Computer Science and its Applications: CSA 2013*, pages 345–351, 2014.

3 Carbon Solutions, LLC. https://www.carbonsolutionsllc.com/, 2024.

4 Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online Buy-at-Bulk Network Design. *SIAM J. Comput.*, 47(4):1505–1528, January 2018. doi:10.1137/16M1117317.

5 Clean Air Task Force. More for less: Strengthening epa's proposed carbon pollution standards can achieve greater emissions reductions at a lower cost, 2023. URL: https://www.regulations.gov/comment/EPA-HQ-OAR-2023-0072-0893.

6 Ovidiu Cosma, Petrica C Pop, and Cosmin Sabo. An efficient hybrid genetic algorithm for solving a particular two-stage fixed-charge transportation problem. In *Hybrid Artificial Intelligent Systems: 14th International Conference, HAIS 2019, León, Spain, September 4–6, 2019, Proceedings 14*, pages 157–167. Springer, 2019. doi:10.1007/978-3-030-29859-3_14.

7 Teodor Gabriel Crainic, Bernard Gendron, and Geneviève Hernu. A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design. *Journal of Heuristics*, 10(5):525–545, September 2004. doi:10.1023/B:HEUR.0000045323.83583.BD.

8 Moustapha Diaby. Successive Linear Approximation Procedure for Generalized Fixed-Charge Transportation Problems. *Journal of the Operational Research Society*, 42(11):991–1001, November 1991.

9 Samira Doostie, Tetsuhei Nakashima-Paniagua, and John Doucette. A novel genetic algorithm-based methodology for large-scale fixed charge plus routing network design problem with efficient operators. *IEEE Access*, 9:114836–114853, 2021. doi:10.1109/ACCESS.2021.3104794.

10 Burak Ekşioğlu, Sandra Duni Ekşioğlu, and Panos M. Pardalos. Solving Large Scale Fixed Charge Network Flow Problems. In *Equilibrium Problems and Variational Models*, Nonconvex Optimization and Its Applications, pages 163–183. Springer US, Boston, MA, 2003.

11 Martina Fischetti and Matteo Fischetti. Matheuristics. In *Handbook of Heuristics*, pages 121–153. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-07124-4_14.

12 Dalila B.M.M. Fontes and José Fernando Gonçalves. Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50:67–76, April 2007. doi:10.1002/NET.20167.

13 Zachary Friggstad, Mohsen Rezapour, Mohammad R. Salavatipour, and Jose A. Soto. LP-Based Approximation Algorithms for Facility Location in Buy-at-Bulk Network Design. *Algorithmica*, 81(3):1075–1095, March 2019. doi:10.1007/S00453-018-0458-X.

14 Bernard Gendron, Saïd Hanafi, and Raca Todosijević. Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research*, 268(1):70–81, July 2018. doi:10.1016/J.EJOR.2018.01.022.

15 Bernard Gendron and Mathieu Larose. Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization*, 2(1-2):55–75, 2014. doi:10.1007/S13675-014-0020-9.

16 Fred Glover. Parametric ghost image processes for fixed-charge problems: A study of transportation networks. *Journal of Heuristics*, 11:307–336, 2005. doi:10.1007/S10732-005-2135-X.

17 Jung-Bok Jo, Yinzhen Li, and Mitsuo Gen. Nonlinear fixed charge transportation problem by spanning tree-based genetic algorithm. *Computers & Industrial Engineering*, 53(2):290–298, September 2007. doi:10.1016/J.CIE.2007.06.022.

18 Erick C Jones Jr, Sean Yaw, Jeffrey A Bennett, Jonathan D Ogland-Hand, Cooper Strahan, and Richard S Middleton. Designing multi-phased CO_2 capture and storage infrastructure deployments. *Renewable and Sustainable Energy Transition*, 2:100023, 2022.

19 Sam Kwong, Tak-Ming Chan, Kim-Fung Man, and HW Chong. The use of multiple objective genetic algorithm in self-healing network. *Applied Soft Computing*, 2(2):104–128, 2002. doi:10.1016/S1568-4946(02)00033-9.

20 Richard S Middleton, Sean P Yaw, Brendan A Hoover, and Kevin M Ellett. SimCCS: An open-source tool for optimizing CO_2 capture, transport, and storage infrastructure. *Environmental Modelling & Software*, 124:104560, 2020.

21 Artyom Nahapetyan and Panos Pardalos. Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. *Computational Optimization and Applications*, 39(1):37–50, January 2008. doi:10.1007/S10589-007-9060-X.

22 National Energy Technology Laboratory. FE/NETL CO_2 transport cost model, 2018. URL: https://www.netl.doe.gov/research/energy-analysis/searchpublications/vuedetails?id=543.

23 Jonathan Ogland-Hand, Kyle J Cox, Benjamin M Adams, Jeffrey A Bennett, Peter J Johnson, Erin J Middleton, Carl J Talsma, and Richard S Middleton. How to net-zero america: Nationwide cost and capacity estimates for geologic CO_2 storage, 2023.

24 Daniel Olson, Caleb Eardley, and Sean Yaw. Planning for edge failure in fixed-charge flow networks, 2024. doi:10.48550/arXiv.2407.20036.

25 Diane Prisca Onguetou and Wayne D Grover. Solution of a 200-node p-cycle network design problem with ga-based pre-selection of candidate structures. In *2009 IEEE International Conference on Communications*, pages 1–5, 2009. doi:10.1109/ICC.2009.5199466.

26 Hossain Poorzahedy and Omid Rouhani. Hybrid meta-heuristic algorithms for solving network design problem. *European Journal of Operational Research*, 182:578–596, 2007. doi:10.1016/J.EJOR.2006.07.038.

27 Caleb Whitman, Sean Yaw, Brendan Hoover, and Richard Middleton. Scalable algorithms for designing CO_2 capture and storage infrastructure. *Optimization and Engineering*, 23(2):1057–1083, June 2022.

28 Fanrong Xie and Renan Jia. Nonlinear fixed charge transportation problem by minimum cost flow-based genetic algorithm. *Computers & Industrial Engineering*, 63:763–778, 2012. doi:10.1016/J.CIE.2012.04.016.

29 Yufeng Xin, George N Rouskas, and Harry G Perros. On the physical and logical topology design of large-scale optical networks. *Journal of lightwave technology*, 21(4):904, 2003.

30 Shangyao Yan, Der shin Juang, Chien rong Chen, and Wei shen Lai. Global and local search algorithms for concave cost transshipment problems. *Journal of Global Optimization*, 33:123–156, 2005. doi:10.1007/S10898-004-3133-5.

31 Sean Yaw, Richard S Middleton, and Brendan Hoover. Graph simplification for infrastructure network design. In *International Conference on Combinatorial Optimization and Applications*, pages 576–589. Springer, 2019. doi:10.1007/978-3-030-36412-0_47.