0. TODO

Required Functions:

- function that calculates cost breakdown
- hook up hyper parameters to UI
- Make scaling down an option to illustrate scaling ease

UI Components

- detailed cost breakdown:
    - need to total cost per size
    - total cost per cost term
    - cost per year
- vehichle type breakdown
    - per year
    - per type
- fuel type breakdown
    - per year
    - per type
- constraints checker

Requirements for level 2 submission

Submissions for Level 2 must be in a digital format and include:

1. A comprehensive document in PDF format which outlines the model's methodology, assumptions, data sources and limitations;
2. A slide deck which explains the model's functionality, performance, business model and business impact;
3. A web link to the prototype of the Concept;
4. A video screen capture of the executable files in the .mp4 format;
5. The deadline for Level 2 Submissions will be by the end of 4 August 2024 Indian Standard Time and late Submissions will not be accepted.

Judging Criteria

Contestants will be shortlisted based on the criteria below:

1. Functionality and Usability:
    - Does the prototype of the Concept effectively demonstrate the mathematical model for optimizing fleet decarbonization strategies?
    - Is the prototype of the Concept fully functional and does it perform as expected?

    - Is the prototype of the Concept user-friendly and easy to navigate?
    - Are there clear instructions or guidance on how users should to interact with the prototype of the Concept?
2. Innovation:
    - Innovation of modelling techniques.
    - Does the prototype of the Concept showcase innovative approaches or techniques in fleet decarbonization optimization?
    - Are there unique features or functionalities that set the prototype of the Concept apart from existing solutions?
3. Scalability and Performance:
    - Can the prototype of the Concept be scaled to accommodate larger fleets or different scenarios?
    - Does the prototype of the Concept demonstrate scalability in terms of computational resources and data handling?
    - How efficiently does the prototype of the Concept execute the optimization algorithms?
    - Are there any performance bottlenecks or limitations encountered during testing?

1. Introduction

1.1 Background

Professional, delivery, and operational fleets play a significant role in the global supply chain, offering flexibility, door-to-door service, and connectivity between cities and towns. However, these fleets are also a major contributor to global greenhouse gas emissions. Fleet owners face the challenge of transitioning to net-zero emissions while maintaining business sustainability and customer satisfaction. This transition involves a complex decision-making process that must account for various factors such as timing, location, and approach.

In this competition, the primary challenge is to develop an optimal fleet decarbonization strategy. This involves solving a non-linear optimization problem characterized by a large number of decision variables. The complexity of the problem arises from the need to balance multiple objectives, such as minimizing carbon emissions, meeting customer demand, and controlling operational costs.

Non-linear optimization problems with a large number of decision variables are inherently difficult to solve. The non-linearity introduces complexities that prevent straightforward solutions, requiring specialized algorithms. Moreover, the high dimensionality of the decision space means that there are numerous possible configurations and combinations of decision variables to consider, making the search for an optimal solution computationally intensive.

The optimization model must account for various constraints and objectives simultaneously. These include emission constraints, vehicle capabilities, fuel consumption rates, and other operational costs. Addressing this challenge requires robust optimization techniques capable of handling the non-linearity and the vast decision space.

Several advanced optimization techniques can be considered for this purpose. Potential algorithms include particle swarm optimization, simulated annealing, and genetic algorithms. Each of these techniques offers unique strengths in exploring and exploiting the search space to find optimal or near-optimal solutions.

This document discusses the choice of using a genetic algorithm to address the optimization problem. The methodology, assumptions, data sources, and limitations of this approach are detailed, demonstrating how genetic algorithms can effectively balance the competing objectives and constraints in fleet decarbonization strategies.

1.3 Fleet Transition Problem Statment

The fleet transition problem we are optimizing is stated as a Mixed-Integer Nonlinear Programming (MINLP) Problem with Nonlinear Constraints. The goal is to minimize the cost function while respecting all the constraints:

The cost function:

$ C_{total} = \sum_{yr=2023}^{2038} \left(C_{buy}^{yr} + C_{ins}^{yr} + C_{mnt}^{yr} + C_{fuel}^{yr} - C_{sell}^{yr} \right) $

$ C_{buy}^{yr} = \sum_{v_{yr} \in V_{yr}} C_{v_{yr}} * N_{v_{yr}} $

$ C_{ins}^{yr} = \sum_{v_{yrp} \in F_{yr}} C_{v_{yrp}} * I_{(yr-yrp)}^{v_{yrp}} * N_{v_{yrp}} $

$ C_{mnt}^{yr} = \sum_{v_{yrp} \in F_{yr}} C_{v_{yrp}} * M_{(yr-yrp)}^{v_{yrp}} * N_{v_{yrp}} $

$ C_{fuel}^{yr} = \sum_{v \in U_{yr}} \sum_{f \in F_{v}} DS_{v}^{f} * N_{v}^{f} * m_{v}^{f} * C_{u,f}^{yr} $

$ C_{sell}^{yr} = \sum_{v_{yrp} \in F_{yr}} C_{v_{yrp}} * D_{(yr-yrp)}^{v_{yrp}} * N_{yr,v_{yrp}}^{sell} $

Constraints:

1. A vehicle of size Sx can only meet the demand for a bucket of size Sx.
2. Any request for distance bucket D1 to Dx can be fulfilled by any vehicle that belongs to distance bucket Dx. For instance, a car in distance bucket D4 can meet the need for buckets D1, D2, D3, and D4; in a similar vein, bucket D3 can meet the needs of buckets D1, D2, and D3 but NOT D4.
3. The annual total carbon emissions from fleet operations shall not exceed the carbon emissions limits for that year listed in carbon_emissions.csv. The annual total of carbon emissions is computed using: $ Carbon_{tot}^{yr} = \sum_{v \in U_{yr}} \sum_{f \in F_{v}} DS_{v}^{f} * N_{v}^{f} * m_{v}^{f} * CE^{f} $
4. Every year, the entire demand for every size and distance bucket must be met.
5. Only the year 20xx vehicle model is available for purchase. Diesel_S1_2026, for instance, is only available for purchase in 2026 and cannot be purchased in any earlier or later years.
6. Each vehicles has a 10-year lifespan and needs to be sold at the conclusion of that time. A vehicles purchased in 2025, for instance, needs to be sold before the end of 2034.
7. You are unable to purchase or sell a vehicle in the middle of the year. Every purchase transaction occurs at the start of the year, and every sale transaction occurs at the conclusion of the year.
8. Only 20% of the current fleet's vehicles may be sold annually.

Definitions:

C_{total}: Total cost of fleet ownership and operations across all the years.
C_{buy}^{yr}: Total cost of buying vehicles in year yr.
C_{ins}^{yr}: Total insurance cost incurred on the vehicles in the fleet for the year yr.
C_{mnt}^{yr}: Total maintenance cost incurred on the vehicles in the fleet for year yr.
C_{fuel}^{yr}: Total fuel cost incurred on the operating fleet in the year yr.
C_{sell}^{yr}: Amount received by selling some vehicles in the fleet in the year yr.
V_{yr}: Set of all vehicles purchased in the year yr.
$C_{v_{yr}}$: Purchase cost of a single vehicle with ID vyr.
$N_{v_{yr}}$: Number of vehicles of ID vyr that have been purchased.
F_{yr}: Fleet of vehicles in the year yr.
$C_{v_{yrp}}$: Cost of vehicle in fleet purchased in the year yrp. yrp is the year of purchase and yr is the year of operation such that $yrp \leq yr$.
$N_{v_{yrp}}$: Number of vehicles currently in the fleet in the year yrp.
$I_{(yr-yrp)}^{v_{yrp}}$: Insurance cost in the year yr for vehicle $vyrp$ purchased in the year yrp.
$M_{(yr-yrp)}^{v_{yrp}}$: Maintenance cost in the year yr for vehicle $vyrp$ purchased in the year yrp.
$D_{(yr-yrp)}^{v_{yrp}}$: Depreciation cost in the year yr for vehicle $vyrp$ purchased in the year yrp.
U_{yr}: Vehicles being used (driven) in the year yr. This is a subset of Fyr .
F_{v}: All fuel types applicable for vehicle v.
Ds_{v}^{f}: Distance travelled by vehicle v using fuel f.
N_{v}^{f}: Number of vehicles of type v driving fuel type f.
m_{v}^{f}: Fuel Consumption of vehicle type v driving with fuel type f.
$C_{uf,f}^{yr}$: Cost of unit fuel of type f in the year yr.
$N_{yr,v_{yrp}}^{sell}$: Number of vehicles $vyrp$ to be sold in the year yr.
$Carbon_{tot}^{yr}$: Total carbon emission in the year yr.
CE^{f}: Carbon emission for the fuel type f.\

All the necessary datasets can be found in the dataset folder.

1.2 Objectives

- The solution should converge to an exceptable cost value within a reasonable amount of time
- The solution should be scalable to accommodate larger fleets and longer time periods.
- The solution should be scalable to accommodate new constraints.
- The solution should be packaged in a user friendly way.

2. Methodology

2.1 Overview of Genetic Algorithms

A genetic algorithm (GA) is an optimization technique inspired by the principles of natural selection and genetics. It is particularly well-suited for solving complex optimization problems with large and non-linear search spaces. The key principles and operations of a genetic algorithm are selection, crossover, and mutation. Figure 1 displays the general flow of a genetic algorithm

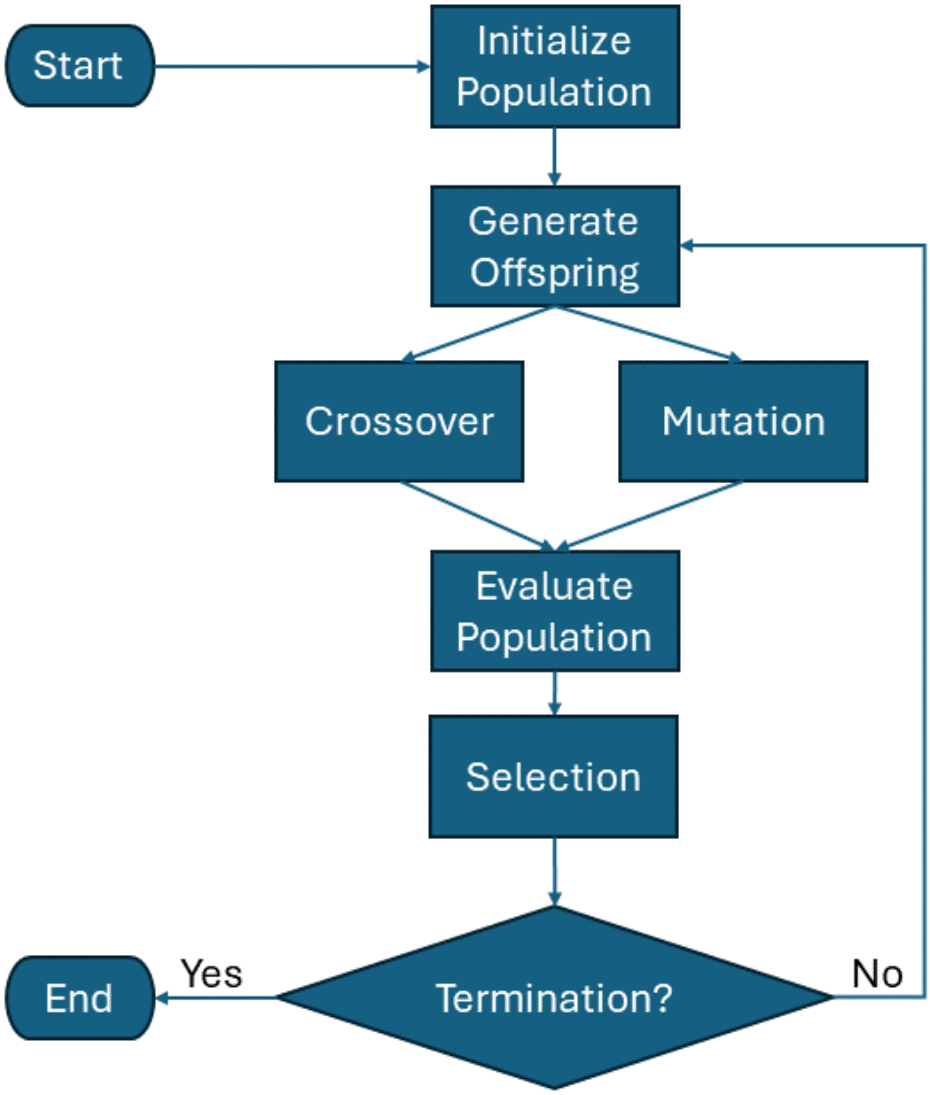

Figure 1: General Genetic Algorithm Flow

Key Principles and Operations

Individual:

An individual refers to a single candidate solution within the population. Each individual is typically represented as a chromosome, which can be encoded in various formats such as binary strings, real-valued vectors, or permutations, depending on the problem domain. The chromosome comprises genes, each representing a particular decision variable or a component of the solution.

Population:

The population is a collection of individuals. It represents the current set of candidate solutions that evolve over time through the application of genetic operators such as selection, crossover, and mutation. The population is iteratively updated to improve the quality of the solutions based on their fitness values.

Selection:

The selection process is analogous to natural selection where the fittest individuals are chosen to reproduce. In a genetic algorithm, a fitness function evaluates each individual in the population, assigning a fitness score based on how well it solves the problem. The selection operation then chooses individuals for reproduction, typically giving higher probability to those with better fitness scores. Common selection methods include roulette wheel selection, tournament selection, and rank-based selection.

Crossover:

Crossover is a genetic operator used to combine the genetic information of two parent individuals to produce new offspring. This operation mimics biological reproduction. The crossover process typically involves selecting a crossover point on the parents' chromosome and exchanging the segments beyond this point between the two parents. Common crossover techniques include one-point crossover, two-point crossover, and uniform crossover. The goal is to produce offspring that inherit the best traits from both parents.

Mutation:

Mutation introduces genetic diversity into the population by randomly altering the genes of individuals. This operation prevents the algorithm from becoming stuck in local optima by ensuring a wider exploration of the search space. Mutation is typically performed with a low probability, modifying one or more genes in an individual's chromosome. Common mutation techniques include bit-flip mutation for binary representations and Gaussian mutation for real-valued representations. Real-World Applications and References Genetic algorithms are used in various fields, including engineering, economics, bioinformatics, and artificial intelligence, for tasks such as scheduling, design optimization, and machine learning.

2.2 Why a Genetic Algorithm?

Genetic algorithms (GAs) are particularly effective for Mixed-Integer Nonlinear Programming (MINLP) Problems. The fleet optimization problem has a complex, irregular, and discontinuous search space. GAs do not require gradient information or smoothness in the search space, making them suitable for this problems where traditional optimization methods may fail. GAs perform a global search rather than a local search. This means they explore a wide range of potential solutions in the search space, which helps in avoiding local optima.

Genetic algorithms are easily parallelized because the evaluation and operations on individuals within the population can be performed simultaneously. This parallelism can be exploited to speed up the search process, especially in large decision spaces where evaluating all possible solutions sequentially would take an unreasanable amount of time.

These two reasons – the ability to handle complex, non-smooth search spaces and the potential for parallelization to enhance computational efficiency – are why a Genetic Algorithm was selected for the solution.

2.?? Assumptions

The following assumptions were made in the development and implementation of the solution:

1. Having a vehicle that drives 0km in a year add to the total cost without providing any other benefit. It only makes sense to keep a vehicle or aqquire a new one if it will be utilized. This means that all

vehicles in the fleet will always drive non-zero km per year.
2. Over satisfying the yearly demand adds to the fuel cost without any benefit. This means that only a total of ceil(yearly_bucket_demand/yearly_vehicle_range) vehicles are required for each bucket. Consequently we do not need to decide on the number of vehicles required for each year, since we can precompute the number for each demand bucket.
3. The only time it makes sense to drive less than a vehicle's yearly range is when it would mean that the yearly demand would be exceeded. This means thst if yearly_bucket_demand = n*yearly_vehicle_range for some integer n then n vehicles will drive the total yearly range. And if yearly_bucket_demand/yearly_vehicle_range is not an integer there are ceil(yearly_bucket_demand/yearly_vehicle_range) - 1 "demand slots" where the allocated vehicle drives its total yearly range and one "demand slot" where a vehicle drives mod(yearly_bucket_demand/yearly_vehicle_range) km.
4. For years < 2032 there is a ceiling on the amount of BEVs that can be utilized, since the electric vehicles (EVs) are restricted to certain demand buckets for years < 2032.

2.3 Algorithm Design

The solution can be broken into four main components:

1. **Data Preprocessing**: In this step the input datasets are used to generate cost and auxillary arrays and matrices. Theses matrices are used in the GAs.
2. **General Optimization GA**: This is where the bulk of the optimization happens. This step determines which vehicles get bought and sold. It is described in detail in section 2.3.1
3. **Fuel Refinement GA**: This is where the fuel allocation gets optimized based on the vehicles that are present in the optimized individual.
4. **Submission Generator**: This is where the costum representation of the individual gets translated into the standard submission format.

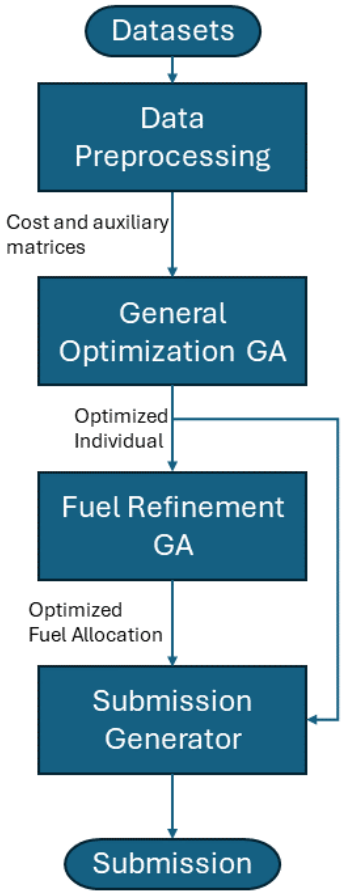

Figure 2: Solution Flow Chart

2.3.1 General Optimization GA

This is where the main optimization occurs. The General Optimization GA is a μ+λ evolutionary algorithm. The algorithm begins by initializing a population of size μ. This population is evaluated, and a new set of offspring is produced by applying crossover and mutation operators to individuals in the initial population. The number of offspring produced is equal to λ. The next generation is selected from the original population of size μ and the offspring of size λ (hence μ+λ). This evolutionary process is repeated for a predetermined number of generations. The individual with the lowest cost is then returned as the optimal solution.

The algorithm has the following hyper parameters:

- **μ (mu)**: The number of individuals to select for the next generation.
- **λ (lambda)**:The number of offspring individuals to generate from the population of size mu.
- **mutation chance**: The chance the an individual from the original population will be mutated to produce a new individual.
- **crossover chance**: The chance that crossover will be applied to two individuals from the original population to produce a new individual.
- **number of generations**: The number of generations that the evolutionary algorithm should be run.

The following sections give a detailed description of each part of the algorithm.

2.2.1 Individual Representation:

Each individual is represented as a 3-dimensional array of integers. This array has dimension (16,16,20) where the each dimension will be reffered to as (depth,rows,columns) respectfullly. An individual can be imagined as a block composed out of 16 x 16 x 20 entries. The number in each entry directly reffers to vehciles of a specific type. Each column in the block represents a different fuel type and size class pair:
| S1 Electricity | S1 HVO | S1 B20 | S1 LNG | S1 BioLNG | S2 Electricity | S2 HVO | S2 B20 | S2 LNG | S2 BioLNG | ..... | S4 BioLNG |
Each row represents the year the vehicles were bought in. Each depth slice of the array represents the year that the vehicles are part of the fleet. So for example:

- (0,0,0) = 5: this represents that 5 S1 EVs were bought in 2023
- (15,14,6) = 3: this represents that 3 S2 Diesels that were bought in 2037 are being used in 2038 and that these diesels vehicles are running on HVO.

Figures 3 and 4 display the layout of an individual. Figure 4 illustartes the fact that if row>depth then the entry must be equal to 0, since we cannot use vehicles from the future in the current year.

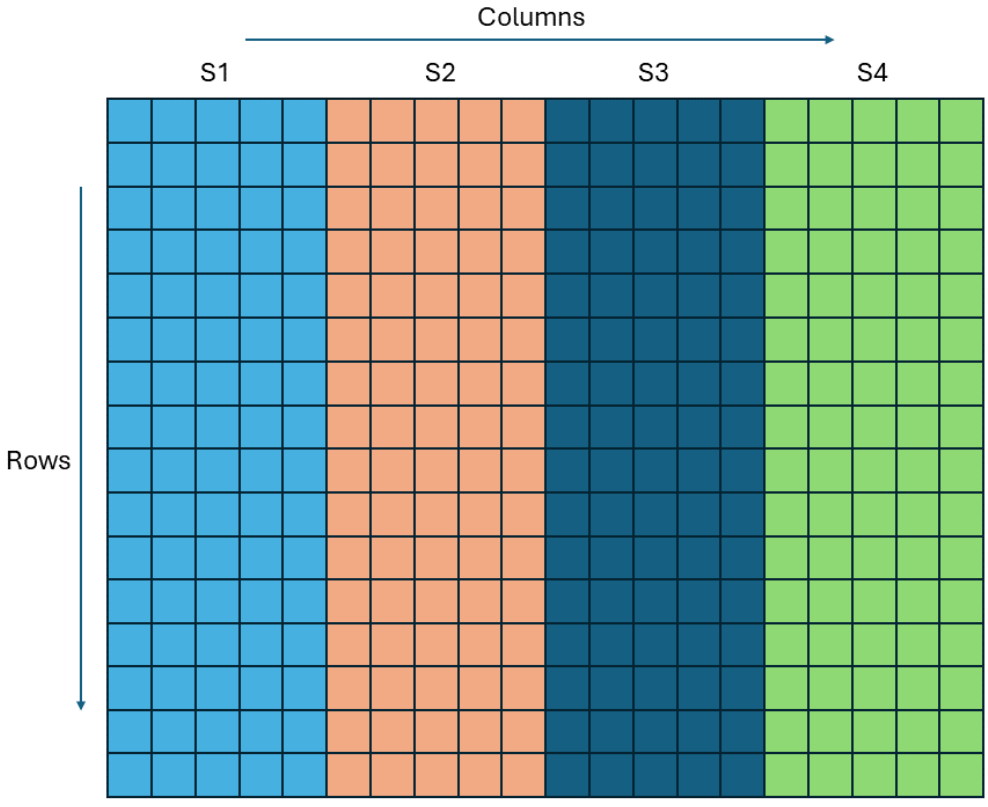

Figure 3: Individual Top View

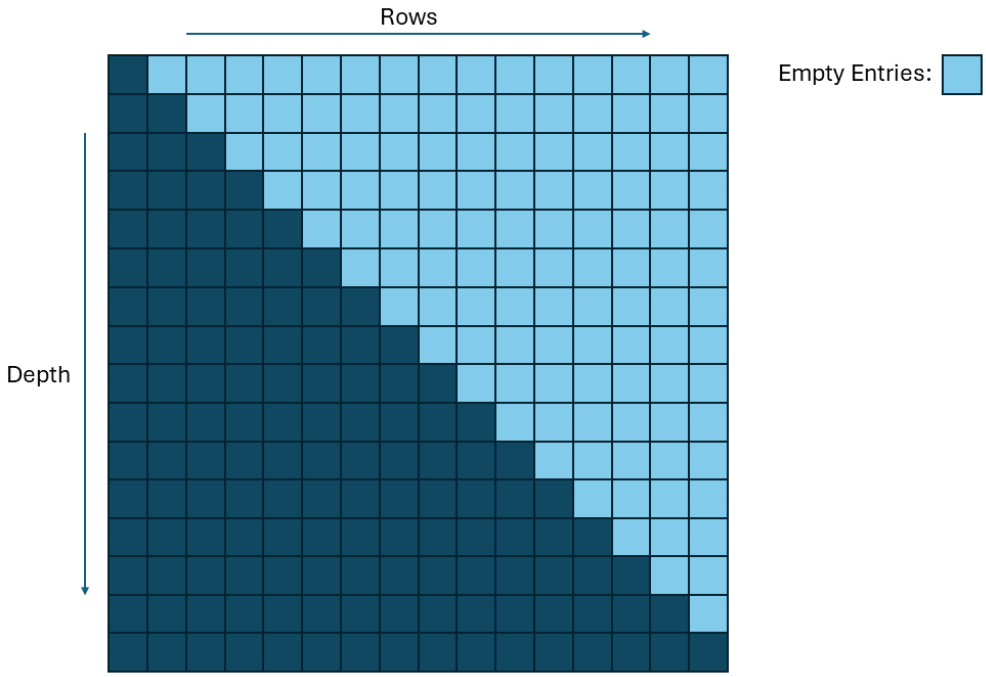

Figure 4: Individual Side View

This representation has a few advantages:

- It is compact which means that it has a smaller memory foot print.
- Contains all the information needed to generate a solution.
- We can use fast Python libraries to act on our arrays.
- The representation is easy to interpret and it does not have to be modified to be read by a human.
- The total vehicles in the fleet for each year can easily be calculated by summing that year's depth slice.
- The vehicles that are sold each year can be calculated by deducting the depth slice below it. ???? An example of an individual is provided in the Appendix.

2.2.2 Population Initialization:

At the start of the algorithm, a population is initialized. The function responsible for creating these individuals is create_individual(), located in the utils.gen_funcs.py module. This function generates a random valid individual. By "valid," it means that the vehicles are properly accounted for, but it does not ensure that all constraints are met. Therefore, an individual might still contain vehicles kept for longer than 10 years or have a fuel spread that exceeds the emissions cap for a year.

The initial population size, denoted by μ, is a parameter defined by the user. The create_individual() function is called μ times to generate the initial population, ensuring that each individual is unique and maintains a high level of diversity.

The fitness function, discussed later, is responsible for weeding out individuals that do not meet the constraints. The create_individual() function guarantees only that the produced individuals have a valid fleet inventory and that the population's diversity is sufficient for fast initial convergence.

Example:
For instance, if μ is set to 100, the create_individual() function will generate 100 distinct individuals, each representing a potential fleet configuration. These configurations will include various combinations of vehicle types, ages, and fuel usages, providing a diverse starting point for the optimization process.

In summary, the population initialization step sets the stage for the genetic algorithm by creating a diverse set of candidate solutions, which will then be refined through the evolutionary process.

2.2.5 Mutation Method:

The mutate() function is one of the functions called during the creation of offspring, located in the utils.gen_funcs.py module. It is responsible for modifying an existing individual so that the solution search space "around" that individual can be explored. There are three types of mutations that are applied to an individual:

- **Buy Reshuffle**: A year and a size class are randomly selected. All the vehicles of this size class, bought in this year are reshuffled. For example if 5 EVs can be reshuffled to 3 diesel and 2 LNG vehicles. The fuel allocation is also randomized for the new spread of vehicles.
- **Fuel Reshuffle**: A year and size class are randomly selected. The fuel allocation of the vehicles in this size class are reshuffled from the selected year to the final year. For example if 2 diesels were allocated to run on HVO and 2 on B20 the reshuffel might result in 3 running on HVO and 1 running on B20. This mutation preserves the number of EVs, diesels and LNGs.
- **Single Vehicle Mutation**: This mutation randomly selects a single vehicle contained in the individual and then mutates its type, fuel allocation and the year that it is sold in.

Only one of the mutations is applied per call. All of them contain extra logic to ensure that the vehicles are properly accounted for and sometimes leads to other parts of the individual being mutated to ensure that the new individual is valid. These mutation were chosen because they provide the right mixture of large and small steps through the search space. This ensures that the population does not get stuck in local minima and that the optimization does not converge too slowly.

2.2.4 Crossover Method:

The mate() function is the other function that is called to create new offspring, located in the utils.gen_funcs.py module. It takes two individuals as input and returns a new offspring individual that is a mixture of the two. A random size class is selected and this size class is then swapped between the two individuals. For example if S1 is selected the columns 0 to 4 are swapped and a new individual is produced that has columns 0 to 4 from the first parent and 5 to 19 from the second parent. Figure 5 shows an example of a depth slice where S2 was selected.

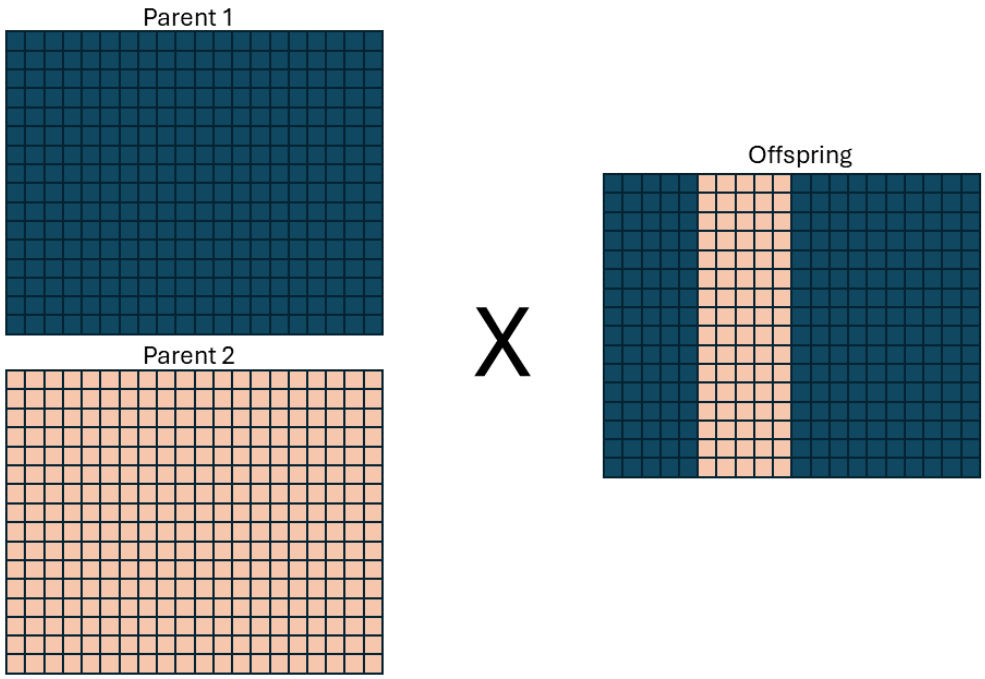

Figure 5: Crossover with S2 Selected

2.2.6 Fitness Function:

The evaluate() function is responsible for calculating an individual's fitness/cost and penelizing individuals that do not meet the constraints. The steps that the evalute function does the following:

1. **Checks Constraints**: The function checks if constraints 2,3,6,8 mentioned in section 1.3 are violated. All the other constraints are inherently respected by the way the solution is setup. If any of these contraints are violated a hefty penalty is added to the individuals cost and the function even returns early on some violations to safe processing time.
2. **Calculates C_{buy}, C_{ins} and C_{mnt} Cost Terms**: If the individual does not violate any of the constraints were the functions returns early these cost terms are calculated. Each of these terms are calculated by taking the Hadamard product of the individual and a precomputed cost array of the same shape. The sum of each resulting array is calculated. The resulting sum of these results represent C_{buy} + C_{ins} + C_{mnt}. Section ?? provides more information about the precomputed cost arrays.
3. **Calculate C_{sell}**: The amount of vehicles sold in each year is calculated by the get_sold() function. This function returns an array with shape (16,16,12). Its columns represent the size classes (S1/S2/S3/S4) and vehicle types (EV/Diesel/LNG) combinations (4x3) and the value in each entry is the number of vehicles sold in that specific year. The Hadamard product of this array and a precomputed cost array of the same shape is taken and the sum of the resulting array is C_{sell}.
4. **Conduct Simple Fuel Optimization and Calculate C_{fuel}**: The evalution function performs a simple fuel allocation oprimization. In this step all the demand buckets are assigned a fuel from the individual based on a simple greedy minimization algorithm. The optimum allocation cannot be used, since it is also a Integer Programming problem and would take too long to calculate. The C_{fuel} term is then calculated from the greedy allocation by calculating $DS_{v}^{f} * m_{v}^{f} * C_{u,f}^{yr}$ for each demand bucket and summing all the results.

2.2.3 Selection Method:

The selection function used in this algorithm is elitist selection (or truncation selection). In this method, individuals are ranked based on their fitness scores, and the top μ individuals with the highest fitness values (the lowest costs) are chosen to form the next generation. This approach ensures that the best solutions are preserved, promoting convergence towards optimal solutions while maintaining a high quality of individuals in the population.

2.3.2 Fuel Refinement GA

Describe the specific design of your genetic algorithm, including:

2.2.1 Encoding:

How solutions are represented (e.g., binary strings, real numbers).

2.2.2 Population Initialization:

How the initial population is generated.

2.2.4 Crossover Method:

How crossover is performed (e.g., one-point, two-point, uniform crossover).

2.2.5 Mutation Method:

How mutations are introduced into the offspring.

2.2.6 Fitness Function:

How the fitness of each individual is evaluated.

2.2.3 Selection Method:

The selection function used in this algorithm is also elitist selection.

2.4 Algorithm Implementation

2.?? Languages and Tools

All of the code for this solution is written in Python. The following packages were used to aid in the creation of this solution:

- Numpy: A fundamental package for scientific computing in Python, providing support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. Numpy is exstensively used throughout the entire solution. It is used to ease the mamipulation of arrays and because of its speed and synergy with Numba.
- Numba: A just-in-time compiler for Python that translates a subset of Python and NumPy code into fast machine code, enabling high-performance execution of numerical functions. Numba is exstensiveley used in both GAs. Without Numba these algorithms would be orders of magnitude slower. Numba is used to compile almost all the functions in the GAs and to parralalize the work load across multiple CPU cores.

- Pandas: A powerful data manipulation and analysis library for Python, offering data structures like DataFrame and Series to handle structured data and providing tools for data cleaning, preparation, and analysis. It is used in the preprocessing and submission generation to transform the data into the required formats.
- Streamlit: An open-source app framework for creating and sharing custom web applications for machine learning and data science projects. It allows users to build interactive and visually appealing applications with minimal code. This packages is used for the solution's simple front-end.

2.?? Suggested Hyperparameters

The hyperparameters, except for the number of generations, for both GAs were selected after a grid search was conducted. The number of generations was selected based off of when the algorithm converged.

Hyperparameter	**General Optimization GA**	**Fuel Refinement GA**
Number of generations	5000	1000
Lambda	14000	4000
Mu	6000	1000
Mutation Chance	0.5	0.95
Crossover Chance	0.5	0.05

4. Data Preprocessing

The provided CSV are used to

6. Limitations

Discuss the limitations of your genetic algorithm, including: Model Limitations: Limitations related to the design and assumptions of the model. Data Limitations: Any limitations related to the data used. Computational Limitations: Constraints related to computational resources and efficiency.

7. Conclusion

Summarize the key findings and contributions of your work. Discuss the implications of your results and suggest possible future work or improvements to the genetic algorithm.

8. References

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Mitchell, M. (1998). An Introduction to Genetic Algorithms. MIT Press.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

9. Appendix

9.1 Individual Example

$$ 2023: \begin{bmatrix} 8 & 0 & 32 & 0 & 33 & 0 & 0 & 0 & 29 & 5 & 14 & 0 & 27 & 5 & 35 & 0 & 0 & 0 & 11 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\

2024: \begin{bmatrix} 8 & 0 & 31 & 3 & 30 & 0 & 0 & 0 & 31 & 0 & 13 & 0 & 25 & 13 & 27 & 0 & 0 & 0 & 7 & 0 \ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 4 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\

2025: \begin{bmatrix} 6 & 0 & 28 & 6 & 26 & 0 & 0 & 0 & 18 & 1 & 11 & 0 & 24 & 26 & 12 & 0 & 0 & 0 & 1 & 0 \ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 0 \ 0 & 0 & 0 & 7 & 2 & 0 & 0 & 0 & 14 & 1 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 7 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2026: \begin{bmatrix} 6 & 0 & 25 & 17 & 8 & 0 & 0 & 0 & 10 & 0 & 8 & 0 & 20 & 13 & 23 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0

& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 10 & 3 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 7 & 0 \ 0 & 0 & 0 & 12 & 2 & 0 & 0 & 0 & 10 & 2 & 0 & 0 & 0 & 15 & 1 & 0 & 0 & 0 & 4 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2027: \begin{bmatrix} 3 & 0 & 24 & 19 & 4 & 0 & 0 & 0 & 4 & 5 & 7 & 0 & 16 & 14 & 19 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 5 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 4 & 0 \ 0 & 0 & 0 & 7 & 7 & 0 & 0 & 0 & 7 & 2 & 0 & 0 & 0 & 9 & 6 & 0 & 0 & 0 & 3 & 0 \ 0 & 0 & 0 & 7 & 4 & 0 & 0 & 0 & 6 & 6 & 0 & 0 & 0 & 10 & 6 & 0 & 0 & 0 & 5 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2028: \begin{bmatrix} 3 & 0 & 20 & 1 & 20 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 16 & 11 & 22 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 12 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 9 & 4 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 8 & 0 & 0 & 0 & 3 & 8 & 0 & 0 & 0 & 10 & 4 & 0 & 0 & 0 & 4 & 0 \ 0 & 0 & 7 & 3 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 6 & 4 & 0 & 0 & 0 & 8 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2029: \begin{bmatrix} 2 & 0 & 15 & 0 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 20 & 5 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 9 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 11 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 5 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 9 & 1 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 5 & 4 & 0 & 0 & 0 & 6 & 0 \ 11 & 0 & 0 & 3 & 6 & 0 & 0 & 0 & 2 & 5 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 6 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 &

0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2030:
\begin{bmatrix} 0 & 0 & 6 & 3 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 13 & 3 & 16 & 0 & 0 & 0 & 0 & 0 \ 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 5 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
& 11 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 4 & 0 & 0 & 0 & 1 & 6 & 0 & 0 & 0 & 1 & 9 & 0 & 0 & 0 & 0 &
0 \ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & 0 & 2 & 5 & 0 & 0 & 0 & 1 & 0 \ 11 & 0 & 0 & 6 & 3
& 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 2 & 17 & 0 & 0 & 0 & 4 & 0 \ 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 3
& 1 & 0 & 0 & 8 & 6 & 0 & 0 & 0 & 7 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2031: \begin{bmatrix} 0 & 0 & 4
& 9 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 1 & 3 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 9 & 1 & 0 & 0 & 0 & 0 & 0 \ 0
& 0 & 0 & 1 & 7 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 5 & 4 & 0 &
0 & 0 & 5 & 6 & 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 \ 11 & 0 & 0 & 3 & 6 & 0 & 0 & 0 & 2 & 5 & 0 & 0
& 0 & 0 & 18 & 0 & 0 & 0 & 1 & 0 \ 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 2 & 1 & 0 & 0 & 8 & 6 & 0 & 0
& 0 & 7 & 0 \ 8 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 3 & 0 & 6 & 0 & 0 & 16 & 4 & 0 & 0 & 0 & 4 & 0 \ 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2032: \begin{bmatrix} 0 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0
& 0 & 4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 3 & 5 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0
& 0 & 2 & 0 & 0 & 0 & 4 & 3 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 8 & 1 & 0 & 0 & 0 & 6 & 1 & 0 & 0 & 0 & 5
& 1 & 0 & 0 & 0 & 0 & 0 \ 11 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 7 & 9 & 0 & 0 & 0 & 0 &
0 \ 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 1 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 1 & 0 \ 8 & 0 & 0 & 0 & 0
& 9 & 0 & 0 & 3 & 0 & 6 & 0 & 0 & 14 & 6 & 0 & 0 & 0 & 3 & 0 \ 20 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0
& 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 \ \end{bmatrix}\ 2033: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 &
0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 &
1 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 3 &

1 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \ 9 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 9 & 1 & 0 & 0 & 0 & 0 & 0 \ 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 10 & 1 & 0 & 0 & 0 & 0 & 0 \ 8 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 3 & 0 & 6 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 \ 20 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \ 14 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2034: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \ 5 & 0 & 0 & 6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 3 & 0 & 0 & 0 & 0 & 0 \ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 7 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 3 & 0 & 6 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 \ 20 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 14 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \ 16 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2035: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \ 4 & 0 & 0 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \ 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 1 & 0 & 0 & 0 & 0 & 0 \ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 8 & 1 & 0 & 0 & 0 & 0 & 0 \ 18 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 14 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 16 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \ 17 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2036: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \ 4 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 0 & 0 & 0 & 0 & 0 \ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \ 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 12 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 16 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 17 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \ 20 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0

& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 \ \end{bmatrix}\ 2037: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
& 0 & 0 & 0 & 0 \ 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \ 10 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 \ 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 \ 14 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 19 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 14 & 0 & 0 & 0
& 0 & 15 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \ 20 & 0 & 0 & 0 & 0 & 12 & 0 & 0 &
0 & 0 & 20 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \ 17 & 0 & 0 & 0 & 0 & 13 & 0 & 0 & 0 & 0 & 23 & 0 & 0
& 0 & 0 & 7 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 \ \end{bmatrix}\ 2038: \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 \ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\ 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 10 & 0 & 0 & 0 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0
& 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
& 0 & 0 \ 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 13 & 0 & 0 &
0 & 0 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 12 & 0 & 0 & 0 & 0 & 13 & 0 & 0 &
0 & 0 & 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 16 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 18 & 0 & 0
& 0 & 0 & 0 & 0 & 0 & 0 & 0 \ 17 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 6 & 0 & 0
& 0 & 0 \ 17 & 0 & 0 & 0 & 0 & 14 & 0 & 0 & 0 & 0 & 26 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 0 \
\end{bmatrix}

$$

9.2 Cost Arrays