Utilizing Genetic Algorithms for Optimizing a Metro Network in Pulaski County, AR

Paul Murphy, Titus Reynolds, and Kavan Patel

Abstract

This project uses genetic algorithms to optimize metro station placements in Pulaski County, Arkansas. This was done by collecting and analyzing geospatial data to find feasibility (fitness). Based on this geospatial data, a weighted feasibility grid was created to evaluate station placement feasibilities in the county. A population raster (a heatmap of population density) was generated using the population density data. These two were used to generate a corridor (a bounding box where station placements are restricted to). After the first corridor was generated, a genetic algorithm found the best station placements for a metro line in the corridor. This was done by evaluating populations of generated placements based off of a fitness function. The fitness function included penalties for linearity, number of stations, distances between stations, population served, and mean feasibility of stations. The algorithm iteratively improved the population via selection, crossover, and mutation until a good solution was found. The second corridor was centered on the metro line generated from the first corridor, and the best corridor placement was found using the same procedure as the first. The genetic algorithm ran again to generate a second metro line. The same process was repeated for the third corridor, except the corridor was centered on the first and second metro lines generated. The resulting station placement generations were then connected using a path-finding algorithm. This was repeated ten times, and the best metro network solution was chosen. The result is a metro network with optimal population coverage and efficiency.

Introduction

Pulaski Country, Arkansas, with its growing population, increasingly struggles to meet transportation demands. Introducing a metro rail network offers the potential to reduce this struggle by improving connectivity within a metropolitan area. A low-speed metro system would be faster and more efficient for public use than the current alternatives offered by Pulaski County such as bus transportation. We also hope to create a solution that may be

used to simulate a system for other similar areas given the same input data. However, developing such a metro system requires careful consideration of many factors such as population density, traffic data, and census data for jobs and households.

This project builds on the work of Xiaorong Lai and Paul Schonfeld in 2016 [1]. As a generalization, Lai and Schonfeld define feasibility as the accessibility to commuters and the probability of building a station in an area, taking into account factors such as road traffic, number of jobs, number of households, land availability, cost, and stations interactions with each other. Their project uses genetic algorithms to optimize for the most fit (feasible) rail transit alignments and station locations in Baltimore, Maryland. We follow the same strategy for optimizing station placement. When using genetic algorithms like Lai and Schonfeld, potential solutions are run over multiple generations and evolve based on how "fit" they are. A solution's fitness is determined by how well it meets the feasibility requirements set. The algorithm's evolution involves the following mechanisms:

1. Crossover: Solutions reproduce to create new solutions that build off their parent solutions

2. Selection: Solutions with higher fitness are more likely to reproduce

3. Mutation: Small, random mutations are introduced into the solutions for more diversity.

This project differs from Lai and Schonfeld, however, by incorporating population (using population density data) into a station's demand calculation. Additionally, our project does not incorporate cost into building the metro system; focusing more on population served by a given metro line.

Procedures

Finding the optimal station placements for a metro system required an analysis of the geospatial data of Pulaski County [3-7]. Data was exported as a vector file into Quantum Geographic

Information System (QGIS) [2], and data outside the bounds of Pulaski County was cut. The datasets analyzed were population density, job density, bus station proximity, water bodies, and traffic data. Individual feasibility grids were created for each data set, and then these feasibility grids were combined into one overall feasibility grid (see section 0.1 Feasibility Analysis). From here, 3 rectangular corridors were made for the metro system to have three linear routes. Each corridor was generated based on the previous, and the genetic algorithms were run separately for each corridor to create station placements (see section 0.2 Station Corridors). From there, the feasibility grid was implemented into the genetic algorithm's fitness function, which measured the "fitness" of a potential station layout based on criteria such as the number of stations, distances between stations, and feasibility scores from the grid. Additionally, the population raster data was weighted in the fitness function for ideal population coverage in a radius around a given station. The algorithm iterated and optimized station placement feasibility (see section 0.3 Genetic Algorithms). Once stations were generated, each line of stations was connected by finding the minimum spanning tree of the line using the NetworkX Python library.

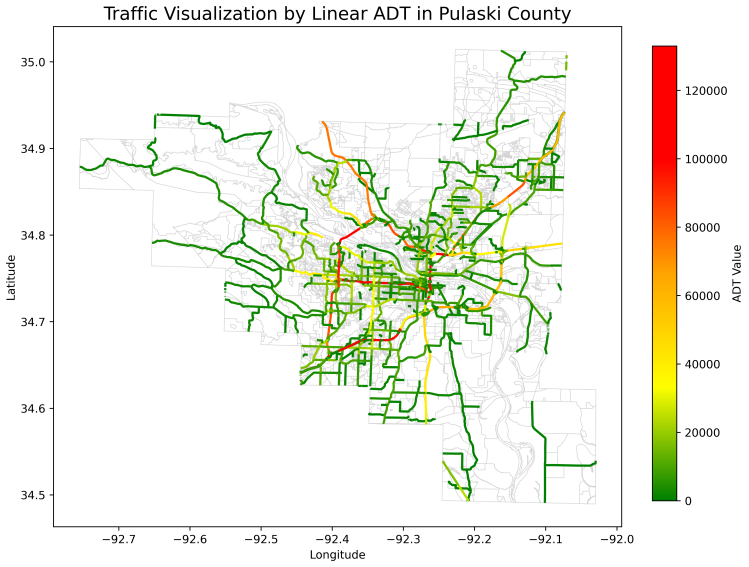

Figure 1: Traffic Data Visualization

The initial data needed to assess grid feasibility was water body data. This data was

obtained from Pulaski Area GIS [3], highlighting areas with significant water bodies such as lakes, streams, and swamps. This data was necessary to assess feasibility because stations can not be built on water. Next, bus station locations were found on Google Maps [4]. Then, these locations were plotted on a custom map using Google My Maps. This data was important because areas within proximity of a bus station attract more people as there is already preexisting transportation infrastructure. After this, ADT (Average Daily Traffic) data, shown in Figure 1, was gathered from ARDOT's GIS portal [5]. Traffic data was useful because areas with more traffic congestion are more likely to attract consumers of public transport. Then, job density data was found through Longitudinal Employer-Household Dynamics [6]. This data is necessary as areas with more jobs would attract more potential metro users. Finally, population per census block data was gathered from the U.S. Census Bureau [7]. This data crucial as areas without demand for public transport are infeasible.

0.1 Feasibility Analysis

After the data was collected, a grid layer consisting of 300 by 300 meter squares was made using the grid tool on QGIS. This grid was made for easier data/feasibility analysis, as the data was processed onto GIS and layered onto the grid using Python. The Geopandas Python library [8] was used to analyze the data and determine feasibility. In all the below steps, excepting the water steps, an exponential decay function was applied to a given grid to ensure the final feasibility grid was weighted. To do this, a new layer was made in a given grid called 'WEIGHTED FEASIBILITY.' Then, an exponential decay function was applied to a given grid so that squares previously marked as infeasible were given a feasibility based on their proximity to the squares marked as feasible. This was done on a sliding scale of approximately 0.001 (further) to 0.999 (closer).

The water body data was used to mark any squares on the grid with significant water coverage as infeasible. Squares that had more than a 30 percent water overlap were given 0 feasibility for potential station placement. These values were saved to a water feasibility

grid.

The population per census block data was used to evaluate feasibility based on the demand near the potential station location. Squares within 1,800 meters (walking distance) of the center of a census block with a population over 3000 were marked as feasible, with all other squares marked as infeasible. These values were saved to a population feasibility grid.

The population data was also used to create a population raster to use in the fitness function. To make the raster, a pixel size of 150 meters was defined. Then, the total population of each census block was evenly distributed among all of the pixels in the raster that intersected with the census block. The population raster was very important for steps beyond the weighted feasibility grid.

Figure 2: Population Raster

The traffic data was used to evaluate feasibility based on walking distance from high-traffic areas. Squares within 1800 meters of measured linear Average Daily Traffic values of over 10,000 were marked as feasible. All other squares and the squares directly intersecting with the linear Average Daily Traffic values over 10,000 were. marked as infeasible. These

values were saved to a traffic feasibility grid.

The bus station location data was used to select areas within walking distance of a bus station that were feasible. Squares within an 1,800-meter radius of a bus station were marked as feasible. These values were saved to a bus station feasibility grid.

The job density data was used to determine if blocks were within walking distance of areas of high job density. Squares within 1800 meters of the center of a census block with a job count over 1000 were marked as feasible, with all other squares marked as infeasible. These values were saved to a job density feasibility grid.

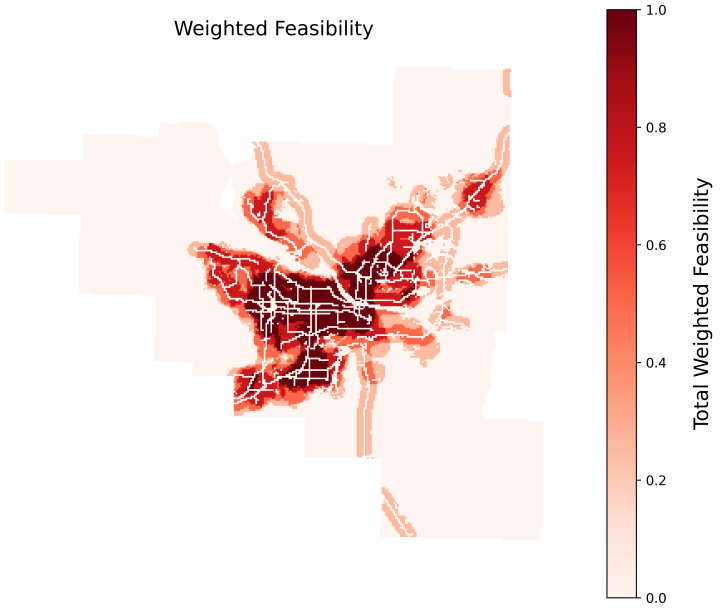

Figure 3: Weighted Feasibility Grid for station placements in Pulaski County

Finally, all of the grids were combined into a single feasibility grid (Figure 3) by averaging every square's feasibility from all of the feasibility grids. However, if one of the squares had over 30 percent water in it based on the water feasibility grid, or a weighted feasibility of zero according to any of the feasibility grids, then the square's total weighted feasibility value was set to zero in the final total feasibility grid.

0.2 Station Corridors

Station linearity should, theoretically, allow a train to run more efficiently as the amount of turns in the route are reduced. Therefore, the train has a faster average top speed, and can more effectively serve a given population. However, only having a single-line of train stations is not effective in serving the population of a larger area, like Pulaski County. Therefore, the final metro system solution had multiple metro lines. To do this, 3 different rectangular corridors (3,000 x 25,000 meters) in optimal orientations were found using the population raster and the total feasibility grid. To generate the first corridor, an initial corridor was centered on every grid with a feasibility greater than 0.75. Then, each temporary corridor was tested in 10 different angular orientations. The temporary corridor which covered the most population and had the greatest feasibility sum (the sum of the flexibilities in all the grids the corridor covers) was selected as the first corridor. Stations were generated in this corridor using a genetic algorithm. The next corridor was centered only on the station grids deemed feasible by the genetic algorithm in the previous corridor. The steps used to determine the best corridor placement and orientation of the first corridor were followed to select the second corridor. The last corridor was centered only on the station grids deemed feasible by the genetic algorithm in the first and second corridors. The steps used to determine the best corridor placement and orientation of the first corridor were followed to select the third corridor. The resulting corridors are shown in the below figure.

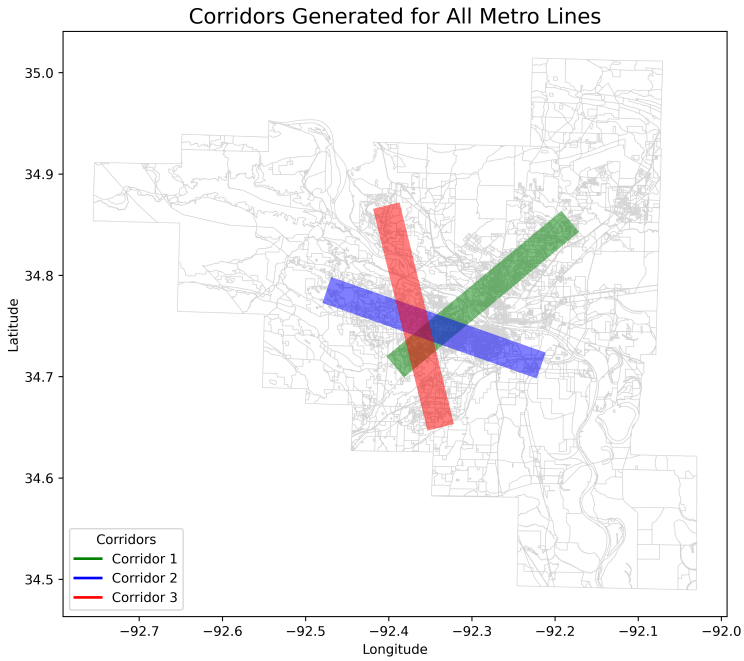

Figure 4: Station Corridors

0.3 Genetic Algorithms

Python libraries were used to implement genetic algorithms to optimize station placements.

1. Geopandas was used to render the overall feasibility grid
2. Distributed Evolutionary Algorithms in Python (DEAP) [9] was used to implement the genetic algorithm
3. Numpy [10] was used for advanced numerical computations
4. Matplotlib [11] was used to plot data

The parameters used in the genetic algorithm were the following:

Additionally, there were constraints added. Stations were not allowed to be placed too close or too far from each other. The minimum distance allowed between stations was walking distance (1800m), while the maximum distance allowed was 20,000 meters. If stations violated these distance constraints, the algorithm penalized the overall fitness of the line of

Parameter	**Value**
Population Size	150
Number of Generations	400
Crossover Probability	0.75
Mutation Probability	0.35
Minimum Number of Stations (per line)	5
Maximum Number of Stations (per line)	10
Minimum Distance Between Stations	1,800 m
Maximum Distance Between Stations	20,000 m
Population Radius	3,000 m
Fitness Weights (W_1, W_2, W_3, W_4, W_5)	$9.0, 3.0, 1.0, 2.0, 10.0$
Exponential Penalty Scaling (α, β)	$0.001, 1.0$

Table 1: Genetic Algorithm Parameters

station placements. This penalty was an exponential function of the violation's magnitude using the alpha and beta parameters. Additionally, the number of stations was constrained so that too many or too few stations did not generate in a metro line. This constraint (the total station count) was set to 5 to 10. A solution with less than 10 stations or more than 15 stations resulted in a penalty in the fitness evaluation (the penalty increased as the number of stations increased or decreased below the threshold). Lastly, the genetic algorithm was constrained by a feasibility parameter determined by the total feasibility grid. The lower the mean feasibility of all the stations in a population, the greater the penalty for this parameter in the fitness evaluation. This constraint was added to the fitness function so stations were not generated in infeasible areas.

A penalty was added to the fitness function to ensure stations in a line were linear. This was done because a train should, theoretically, run more effectively down a linear path. Linearity was calculated by taking the station placements' (x, y) coordinates and doing a linear regression. The R^2 coefficient was calculated to measure how closely the stations resembled a line, and the linearity weight was found by subtracting the coefficient from 1.

Population coverage was also considered as a parameter in the fitness evaluation. This parameter measured how effectively station placements served the local population. The population was calculated in a radius of 2,000 meters around a given station using the

population raster. Then, the calculated population was divided by the maximum possible population, giving the population coverage of the network of stations as a percentage.

The parameters outlined above were used in a fitness function to optimize station placements for a line of stations. The algorithm evolved a population of solutions through selection, crossover, and mutation.

The fitness function evaluated how feasible a potential solutions were. These solutions were then evolved based their individual fitness scores (higher fitness scores were prioritized). Over time, the solution of the genetic algorithm began to resemble the solution with the highest possible fitness score. In this way, the optimal solution for a line of stations was found. The function used to determine the fitness of a solution was the following:

$$\mathcal{F} = W_1 \cdot \left(\frac{\sum_{i=1}^{N} F_{\text{grid}}(i)}{N} \right) + W_2 \cdot \psi - W_3 \cdot P_{\text{distance}} - W_4 \cdot P_{\text{count}} - W_5 \cdot L \tag{1}$$

A description of the variables used in this equation is provided in the table below.

Variable	**Description**
$\mathcal{F}$	The overall fitness of the generated station placements
W_n	A constant used to weigh different components of the fitness function, as some components are more important than others
N	The number of stations in the potential solution
$F_{\text{grid}}(i)$	The feasibility score for the i-th station, from the feasibility grid
P_{distance}	The penalty for solutions that violate the distance constraint
P_{count}	The penalty for solutions that violate the station count constraint
ψ	The population coverage
L	The linearity of the solution

Table 2: Variables and their descriptions in the fitness function

The genetic algorithm was run multiple times inside a given corridor, and its parameters were adjusted until a solution of parameters that resulted in the best and most consistent

fitness evaluation was found. Once these parameters had been found, the code to generate one metro system using the given parameters was run 10 times to generate 10 slightly different metro systems (due to the inherent variability of random seeds and genetic algorithms). The overall fitness of each metro system was evaluated using the fitness function defined above, excepting the linear regression penalty. The generated metro system with the highest fitness from this group was chosen as the final solution.

Results

The overall goal of this project was to find an optimal metro system layout in Pulaski County. The final, selected solution's genetic algorithms average fitness per generation is shown below. All of the genetic algorithms that were used to find the metro lines converged at around 80 generations.

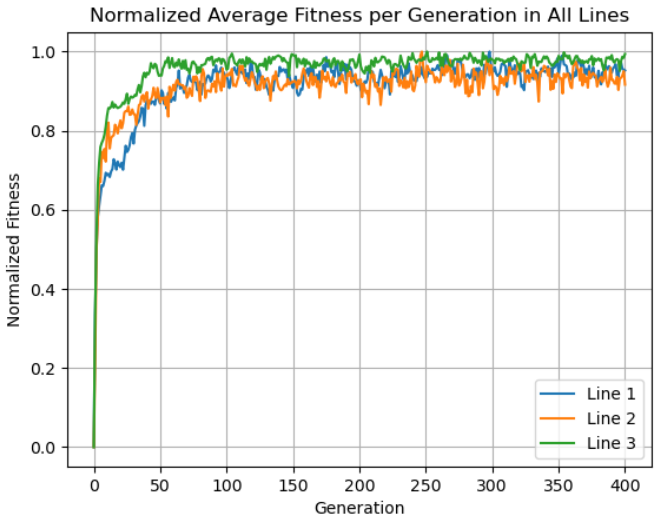

Figure 5: Fitness as a function of generations graph

This graph, showing fitness as a function of generations, gives insight into how quickly genetic algorithms are able to evolve. Given the fitness converged early and remained in a stable range, it is reasonable to assume the genetic algorithm was successful. The station

placements generated by the genetic algorithm are therefore optimal.

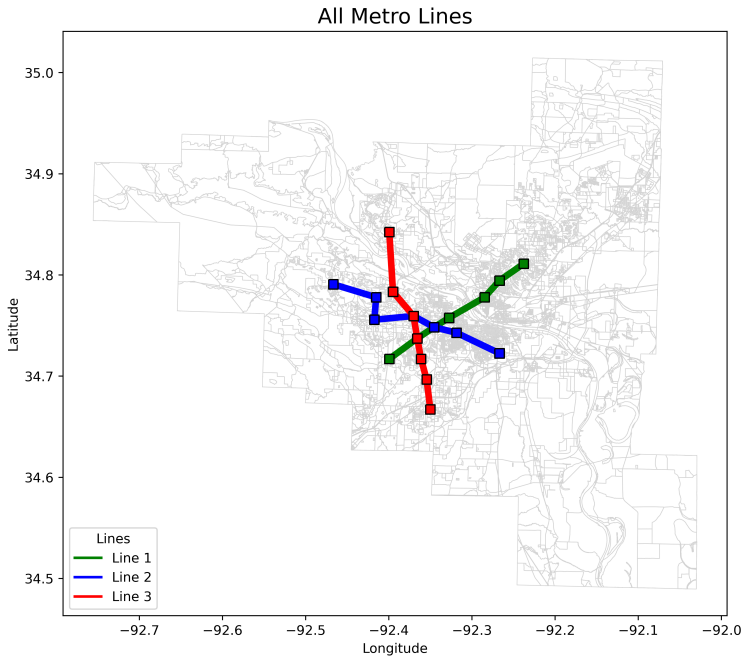

Figure 6: Generated Metro System

The above figure shows a map of the final metro system solution selected out of the 10 proposed metro systems, and the individual lines in the system. This system provides the optimal population and feasibility coverage out of all the generated solutions.

Trial	**Fitness**
1	13.4573
2	12.3447
3	12.8657
4	13.3273
5	12.7604
6	12.8594
7	-25.2737
8	13.1327
9	12.1313
10	12.3628

Table 3: Fitness per trial

Across all instances when the genetic algorithm was called, it took an average of 7 minutes and 13 seconds to run on a system with AMD Ryzen 7 6800H processor (relatively modern

and high end hardware). Once satisfactory parameters for weights and penalties were found, 10 separate solutions were made using these parameters. These solutions differed slightly, as the inherent nature of the genetic algorithm and random seed caused variance each time a solution was reached (even with the same parameters). The table above shows the fitness for each solution. As seen in this table, the average fitness was approximately 12, excluding Trial 7. The distance penalty for Trial 7 was much higher than any other trial, so Trial 7 had a much lower fitness score than any other solution.

Compared to Lai and Schonfeld's work, this solution focused more on the coverage that a set of stations provided to the population. This was done to ensure that stations are, theoretically, better accessible by the residents of Pulaski County.

Conclusion

This project effectively finds a feasible and efficient solution for a metro system. The total feasibility grid enabled the selection of a feasible population for the genetic algorithm, significantly enhancing the quality of the solution. The 3 stage corridor system and linear regression penalty implemented allowed the algorithm to optimize for covered population and feasibility, while generally remaining linear.

The solution highlights the ease and effectiveness of using genetic algorithms in infrastructure solutions. However, there is error present in the project. Data was gathered from public sources, and in many cases, is years old. Yet, there were no available alternatives. Cost was not taken into account, but could be implemented in a future version to improve the feasibility of the project. Additionally, a train schedule penalty was not included in the fitness function, and could also be added to improve the solution.

This project hopes to demonstrate the effectiveness and potential for genetic algorithms and machine learning to be used in urban and transportation planning. At the end of the day, the feasibility of such a solution for Pulaski County is questionable. However, there

should be more attention paid to the lack of public transportation infrastructure in Central Arkansas, and the lack of forward thinking put into public transportation. This project does not claim to be a professional implementation, but we do hope to inspire some change, no matter how small.

References

1. Lai X, Schonfeld P. 2016. Concurrent Optimization of Rail Transit Alignments and Station Locations. Urban Rail Transit. 2(1):1–15. https://link.springer.com/article/10.1007/s40864-016-0033-1

2. QGIS Development Team. Quantum Geographic Information System. Version 3.40. Available from: https://qgis.org.

3. Pulaski Area GIS (PAgis). Pulaski County Water Data. Available from: https://www.pagis.org/webapps/wab/water/.

4. Google Maps. Pulaski County Bus Stations. Available from: https://www.google.com/maps/d/u/0/edit?mid=1d-_YHWwNSNfJ--hp770fxf0owwG6aBc&ll=34.727551922140854%2C-92.31585899999999&z=11.

5. Arkansas Department of Transportation. Linear ADT Traffic Data in Pulaski County. Available from: https://gis.ardot.gov/portal/apps/webappviewer/index.html?id=7c81a313f4174b99b2a01713c328bb7a.

6. U.S. Census Bureau. LEHD Jobs Per Census Block in Pulaski County. Available at: https://lehd.ces.census.gov/.

7. U.S. Census Bureau. Census Block Data and Population. Available from: https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html.

8. Jordahl K, Van den Bossche J, Fleischmann M, Wasserman J, McBride J, Gerard J, Tratner J, Perry M, Garcia Badaracco A, Farmer C, Hjelle GA, Snow AD, Cochran M, Gillies S, Culbertson L, Bartos M, Eubank N, maxalbert, Bilogur A, Rey S, Ren C, Arribas-Bel D, Wasser L, Wolf LJ, Journois M, Wilson J, Greenhall A, Holdgraf C,

Filipe, Leblanc F. GeoPandas. Version 1.01. 2024. Available from: https://github.com/geopandas/geopandas

9. Fortin FA, De Rainville FM, Gardner MA, Parizeau M, Gagné C. Distributive Evolutionary Algorithmns in Python (DEAP). Available from: https://github.com/DEAP/deap

10. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, van Kerkwijk MH, Brett M, Haldane A, Fernández del Río J, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE. Numpy. 2020. https://github.com/numpy/numpy

11. Hunter JD. Matplotlib. Version 3.9.3. Availavle from: https://github.com/matplotlib/matplotlib