Title: Optimizing Vehicle Routes and Planning for Water Treatment Maintenance Team
Team024: Yu J Chang, Cemil Ovacik, Lauren R Wilder, Finot Berhanu, Sebastian A Sanchez, & Sukriti Raut

Introduction

At the crossroads of data science and community service, this project began with a unique blend of skills and opportunities. One of the team members works at Gwinnett County Department of Water Resources near Atlanta GA. This connection opened the door to using data for positive change in the county, focusing on a significant problem: the water treatment facilities – specifically the over two hundred stations used to pump water to the treatment facilities – struggle with overtime, project delays, and scheduling issues. With no data-driven scheduling system in place, a solution was proposed to optimize routes, cut technician overtime, and improve project timelines while also preparing for unexpected emergencies. A similar solution, with the functionality and flexibility proposed, does not exist in the utility maintenance industry.

Problem Definition

The maintenance staff at the Gwinnett County Department of Water Resources must perform routine maintenance work by visiting various pump stations each day. This team aims to create an interactive application powered by mathematical vehicle routing algorithms for the staff members to receive an optimized route to plan their workday to save time and improve efficiency.

Literature Survey

In the exploration of the traveling salesman and vehicle routing problems (TSP, VRP), numerous optimization models were found that effectively reduce travel time and costs in practical scenarios. Some methods provide precise solutions but with higher computational demands, while others strike a balance between accuracy and efficiency using heuristic or metaheuristic approaches. A summary of findings on current practices is below:

Nearest Neighbor Algorithm (NN) **–** NN is a heuristic approach often applied to routing optimization, such as TSP. Starting from a random location, it iteratively finds the nearest unvisited point (He et al., 2010; Tjandra et al., 2022; Dasari & Singh, 2023). Such heuristic methods can offer insights into balancing efficiency with different constraints, aligning with our project's goals (Dasari & Singh, 2023). However, NN may not always find the optimal route, requiring more advanced methods for optimal solutions.

Brute Force & Dynamic Programming **–** Brute Force examines all route combinations, while Dynamic Programming accelerates this process by avoiding repeated calculations. This approach is enhanced using elimination tests (Dumas et al., 1995). It efficiently manages constraints, as discussed by Samah et al. (2019), and aligns with our project's challenges. The downside of this approach is that Brute Force lacks scalability, and while Dynamic Programming helps, it does not fully resolve this issue.

Genetic Algorithm (GA) **–** GA simulates biological evolution. A notable variant, NSGA-II, optimized sewer system schedules, resulting in a 26% increase in daily job completions (Draude et al., 2022). GA suits TSP and route-optimization problems, resonating with our project's varied objectives. However, GA requires parameter fine-tuning and risks early convergence (Adewole et al., 2011).

Other Algorithms & Approaches **–** Multiple algorithms, such as those combining Facility Location Problem (FLP) with VRP (Oran et al., 2012), and GPU-based solutions (O'Neil, 2016), among others, tackle route optimization. These diverse methodologies broadened the understanding of TSP and route optimization. Every technique has pros and cons. The aim is to harness the best while mitigating challenges.

Methodology

Data: The dataset being used was acquired from the Gwinnett County Department of Water Resources with the directors' approval. The dataset (approx. 23K rows) fed into the routing algorithms comes from

an online check sheet system Maximo and eLogger and consists of data from June 2021 through October 2023, since their original installments. It includes information about the nature and locations of the pump stations visits. An additional but related large dataset (>350K rows) is also utilized to build other supporting models. This second dataset comes from the computerized maintenance management system (CMMS) consisting of completed work orders from the last thirteen years. It contains various characteristics data of work orders such as its type, count, labor hours, labor cost, material cost, and work order cost. This data was utilized to build classification models to predict priority levels of the work orders which will be the basis for grouping locations into different routes to plan a given workday.

Intuition: The utility already uses technologies to assist in the planning and scheduling of work, but it is woefully lacking in assistive technology to plan routes efficiently and dynamically when a maintenance technician needs to be in multiple sites throughout the 437 square miles of Gwinnett County. Currently, the information provided about each stop is usually contained within different software systems and not aggregated within one map, where it would be more useful for a manager planning work for the day or week.

This project is innovative as this level of specificity and information in a centralized tool does not currently exist in the water/wastewater maintenance industry. Google can add stops to routes, but it is up to the user to define what they perceive as the best route. Given a list of locations, for e.g., 1-2-3-4, generic routing solutions such as Google Maps will simply display the locations in the original order of 1-2-3-4 with the shortest duration between each pair of locations. However, the TSP and VRP algorithms implemented in the project will create a new optimal route with the shortest distance and duration while considering the facility's specific needs and constraints such as number of available vehicles, priority of locations, and optimal number of locations per route. Thus, the ordered list could look something like 2-3-1-4, for instance. The web application will then display the optimal routes on a single and user-friendly interface.

Algorithms: Three different versions of vehicle routing optimization algorithms were developed in Python to determine an optimal path from a given list of locations beyond what is currently offered in industry: Brute Force (BF), Nearest Neighbor (NN), and Genetic Algorithm(GA). The rest of this paper will refer to these algorithms by their abbreviations.

BF – Given a list of locations for a route with the same starting and ending location, the BF algorithm calculates the total distance for all permutations and picks the shortest route using the shortest distance and calculates the duration time of that route (refer to Figure 1 in the Appendix).

NN – Given a precomputed distance matrix, the NN algorithm starts at position 0 (refer to Figure 2 in the Appendix), identifies the nearest unvisited location, and continues this process until all locations are visited, creating an optimized route.

GA – *Step 1*: The GA begins with the initial population (comprising different routes); *Step 2*: It evaluates each route's fitness, favoring shorter distances; *Step 3*: It selects two of the highest scoring routes from step 1 as 'parents' using a roulette wheel method for reproduction. It performs a crossover, with each child (new route) inheriting half of the genes (locations) from each parent. *Step 4*: It introduces random mutations by swapping the positions of two genes (locations) in the resulting child. *Step 5*: It replaces a portion of the old population in step 1 with the children (new routes) from steps 3 & 4, shuffles the new population and returns it. The process is repeated for a specified number of generations before terminating (i.e., obtaining the route with minimum distance, as indicated in Figure 3 in the Appendix).

The GA was fine-tuned by adjusting four key hyper-parameters: population size, crossover rate, mutation rate, and number of generations. The goal of the performance tuning was to evaluate the algorithm's efficiency in computing minimal distances while comparing execution times across varying parameter settings. Further analysis on hyper parameter tuning (refer to figures 4, 5, and 6 in the Appendix) revealed that the parameters' impact on the algorithm followed this order of significance: population size > number of generations > crossover rate > mutation rate.

Classification Models – Using the additional large work order dataset, classification models such as k-nearest neighbors (KNN) and Random Forest (RF) were developed to determine the priority level of a pump station visit based on variables such as work order type and status, labor hours and cost, material cost, and work order cost. The original sample suffered from a class-imbalance issue where about 90% of the data belonged to the class of priority '3'. To mitigate the issue, models were also fitted using two resampling methods using the Python package imblearn: under-sampling and over-sampling. Both methods ensured that all classes had equal representation of data, based on the smallest class for under-sampling and largest class for over-sampling, respectively. The predictions of priorities for daily work orders will be manually assessed by a subject matter expert and the filtered list will be used as an input of the K-Means grouping mentioned below.

K-Means Grouping – Additionally, instead of randomly selecting the list of locations to form a route, a K-Means algorithm was developed to form cohorts of locations (i.e., routes) based on the available number of vehicles (K=number of vehicles) and priorities from the classification model. For example, if there are thirty locations that require maintenance on a given day and only five vehicles are available, K-Means will be used to divide them into optimal groupings (i.e., 5 routes of 6 locations for instance if no locations were removed based on priority assessment).

Web Application: A web application called *Shortest Path* has been developed by the project team, enabling users to enter multiple locations, number of vehicles and view the optimal route generated using the above algorithms. This web application stands out due to its robust backend, efficient and interactive front-end, and the use of sophisticated technologies such as FastAPI, Leaflet, Open Source Routing Machine (OSRM), and AWS**.**

Input – An open-source routing machine (OSRM) and OpenStreetMap data were used to seamlessly integrate a custom API within the web application server. The API serves to generate comprehensive distance and time duration matrices based on the shortest route's duration between pairs of supplied coordinates of pump stations within network, which are used as essential inputs for a range of algorithms.

Front-end – The interactive map is powered by Leaflet, an open-source JavaScript library, and OpenStreetMap to render detailed geographic data. The map allows users to place or remove markers (locations), mark a specific marker as 'Home', and even upload a list of locations using a CSV file that automatically self-populates the graph. The UI design is responsive, ensuring usability across various devices and screen sizes. The application also provides the flexibility of running single vs multi-vehicle algorithms by easily selecting from the dropdown.

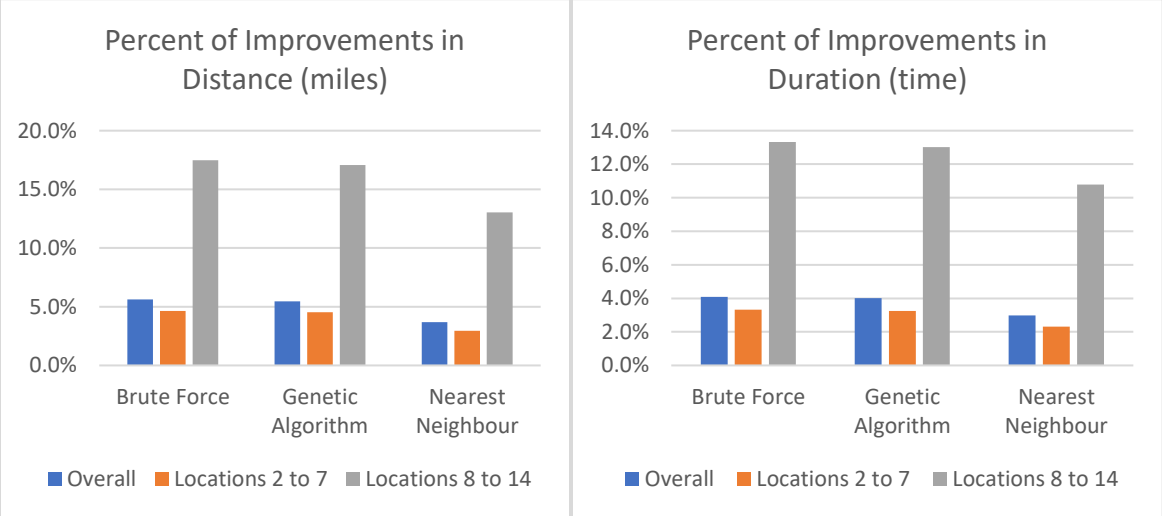
Back-end – The backend is powered by FastAPI, a modern, fast (high-performance) web framework for building APIs with Python. It is well-suited for asynchronous tasks and is optimized for speed, which is crucial for applications handling extensive data, as in geospatial computations. OSRM is utilized

in a Docker container for advanced geospatial computations, particularly for calculating distance matrices and optimal routes.

Experiments/Evaluation Methods

In this study, a multifaceted approach was used for evaluating the efficacy and efficiency of potential algorithms. Evaluations were based on comparative analysis, scalability testing, situational usage, accuracy metrics, and usability feedback. The primary objective was to understand how these methods compared against the original data, how they scale with increasing inputs, and when each algorithm is best employed based on a given situation. The results and insights gleaned from these evaluations are as follows:

Comparative Analysis – *How well does each algorithm perform compared to the original total distance and duration?*



Comparing three versions for each algorithm, a small percentage of improvements in distance and duration is observed when applied to routes containing two to seven locations. However, we see a significant improvement as the number of pump stations visited increases from eight to fourteen per route.

Improvement (%)	BF	NN	GA
Distance			
> 8 stations	17.49% (8 miles)	13.04% (6 miles)	17.07% (7.9 miles)
< 7 stations	4.65% (1.5 miles)	2.95% (0.9 miles)	4.52% (1.4 miles)
Duration			
> 8 stations	13.32% (13 minutes)	10.79% (10.5 minutes)	13.01% (12.9 minutes)
< 7 stations	3.32% (3.1 minutes)	2.32% (1.5 minutes)	3.25% (2.13 minutes)

Based on the savings in average distance travelled (measured in miles) and average duration (measured in minutes), BF seems to be performing the best followed closely followed by GA and NN. For more detail on the improvement statistics, refer to figures 7, 8 and 9 in the Appendix below.

Scalability Test *– How long does it take for each algorithm to return an output as the number of locations increases?*

- BF works well for up to 11 locations, but after that, it becomes slower due to many possible route combinations. For 14 locations, it takes around 12 hours to get the result.
- NN is quick and can handle many locations almost instantly because it simply finds the next shortest point without complex calculations.
- GA works efficiently for many locations. Its execution time stabilizes after 11 locations, averaging about 4 minutes.

Scalability vs Execution Time (Second)

Number of Locations	BF Time	NN Time	GA Time
2	0.00	0.00	0.00
3	0.00	0.00	0.10
4	0.00	0.00	0.19
5	0.00	0.00	0.71
6	0.00	0.00	3.09
7	0.01	0.00	19.49
8	0.05	0.00	85.36
9	0.46	0.00	80.23
10	5.17	0.00	69.95
11	17.19	0.00	261.28
12	424.00	0.00	379.02
13	2,255.45	0.00	207.81
14	44,480.34	0.00	231.74

Situational Analysis – *Which algorithm should be employed based on a situation?* A strategy has been developed to determine the optimal algorithm based on the number of locations.

- For routes with fewer than 11 locations, the BF algorithm is used in the application as it provides the optimal route based on the robustness of the algorithm. NN and GA have similar improvements in duration and distance, thus can be employed as well, preferring NN over GA for efficiency.
- For routes with more than 11 locations, NN proves to be the better choice due to its efficiency compared to GA, despite GA having slightly better improvements in both distance travelled and duration.

Accuracy Metrics *– How did the classification models perform based on accuracy metrics?* Below is the table with accuracy and balanced accuracy for the different classification models applied to the work order dataset to predict priority classes for work orders. Based solely on accuracies, KNN applied to the original sample with K=6 seems to be performing the best. However, the balanced accuracy for the same is low, which is more appropriate evaluation metric to evaluate models in case of imbalance. Thus, based on consistent accuracy and balanced accuracy of 77%, RF fitted on the over sampled dataset was utilized in the project. Optimized means that the hyperparameters of the models were tuned to find the best parameters.

Models	**Accuracy**	**Balanced Accuracy**
KNN w/ Original Sample (K=6)	0.94	0.55
KNN w/ Under-Sampling (Optimized)	0.62	0.62
KNN w/ Over-Sampling (Optimized)	0.66	0.66
Balanced RF w/ Original Sample (Optimized)	0.87	0.54
RF w/ Over-sampling (Optimized)	0.77	0.77

Similarly, values for precision, recall/sensitivity, and f-1 score were also the highest for the RF model fitted on the over sampled dataset for all classes compared to that of other models (refer to figure 10 in the Appendix). This provides further evidence that the model can correctly predict positive cases and can identify a large portion of positive instances from the dataset, even if it means occasionally classifying some negatives as positives. Capturing more false positives is not a significant issue in this case as there could be worse repercussions if the urgent work orders/locations are not captured correctly but do not uphold the same negative impact if non-urgent cases are falsely classified as urgent.

Usability Feedback – Team members interacted with the web application and gathered feedback highlighting its simplicity, versatility, interactivity, and informativeness. A test utilizing the K-Means + NN algorithm on thirty locations resulted in the generation of five optimal routes for five vehicles within seconds, as depicted in Figure 11 of the Appendix.

Discussion

Comparative analysis revealed that as the number of pump stations per route increased, significant improvements in distance and duration were observed, with BF showing the best performance, closely followed by GA and NN. Scalability testing indicated that Brute worked well for up to 11 locations but became significantly slower afterward, while NN demonstrated quick processing, and GA showed stable efficiency after 11 locations, taking about 4 minutes. Thus, based on situational analysis, BF is preferred for shorter routes (<11 locations) making it a default algorithm to show the optimal route for the web application. For routes greater than 11 locations, GA can be used to generate an optimal route while NN can be used for, potentially non-optimal, but faster route generation.

In terms of the classification modeling component of this project, the accuracy and balanced accuracy metrics, precision, recall, and F1 score values all indicated that RF on an oversampled dataset consistently outperformed other models, offering reliable predictions for priority classes in work orders. Regarding the web application *Shortest Path,* the design is not just focused on functionality but also on scalability and user experience, making it a comprehensive solution for interactive geospatial data handling and visualization tasks.

In terms of the future adaptability and scalability of the proposed model, it may be advantageous to automate certain manual aspects of the multi-vehicle routing process within. This may involve the automatic export of daily work orders from the County database (CMMS) to a network sharing folder for easy access. A Python script can be developed to execute the RF classification function, processing the daily work orders report from the CMMS database. The web application can be set up to automatically read the resulting CSV file containing predictions, streamlining the entire process.

Conclusion

In essence, our web application for pump station maintenance successfully integrates advanced mathematical and geospatial optimization techniques. This implementation proves pivotal in the strategic planning of routes, enhancing efficiency, and resulting in significant time and cost savings for the county. Our research indicates that, for route optimization, the BF algorithm excels in handling shorter routes, while NN algorithms demonstrate superior performance in optimizing longer routes. Through a comprehensive analysis and literature review, we ascertain that our methods stand out favorably against industry standards. These approaches effectively minimize travel distance, time, and costs, achieving a delicate balance between accuracy and efficiency through optimal or heuristic solutions. The approach used in this project stands out further through its integration of various methods, including the use of mathematical vehicle routing algorithms, classification models, and a web application, making it a robust and versatile platform.

Team Contributions: All project team members contributed equally, leveraging their strengths.

References

1. Adewole, P., Akinwale, A. T., & Otunbanowo, K. (2011). A Genetic Algorithm for solving travelling salesman problem.
2. Atefi, R., Iori, M., Salari, M., & Vezzali, D. (2022). Solution of a Practical Vehicle Routing Problem for Monitoring Water Distribution Networks. arXiv preprint arXiv:2202.02549.
3. Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. omega, 34(3), 209-219.
4. Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.
5. Cheikhrouhou, O., & Khoufi, I. (2021). A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. Computer Science Review, 40, 100369.
6. Dasari, K. V., & Singh, A. (2023). Two heuristic approaches for clustered traveling salesman problem with d-relaxed priority rule. Expert Systems with Applications, 224, 120003.
7. Draude, S., Keedwell, E., Kapelan, Z., & Hiscock, R. (2022). Multi-objective optimisation of sewer maintenance scheduling. Journal of Hydroinformatics, 24(3), 574-589.
8. Dumas, Y., Desrosiers, J., & Gelinas, E. (1995). An Optimal Algorithm for the Traveling Salesman Problem with Time Windows Operations Research, 43.
9. Fontecha, J. E., Guaje, O. O., Duque, D., Akhavan-Tabatabaei, R., Rodriguez, J. P., & Medaglia, A. L. (2020). Combined maintenance and routing optimization for large-scale sewage cleaning. Annals of Operations Research, 286, 441-474.
10. Griffin, C. (2012). Linear Programming: Penn State Math 484 Lecture Notes.
11. Haixiang, G., Fang, W., Wenwen, P., & Mingyun, G. (2021). Period sewage recycling vehicle routing problem based on real-time data. Journal of Cleaner Production, 288, 125628.
12. He, R., Xu, W., Wang, Y., & Zhan, W. (2010, April). A route-nearest neighbor algorithm for large-scale vehicle routing problem. In 2010 Third International Symposium on Intelligent Information Technology and Security Informatics (pp. 390-393). IEEE.
13. Jaillet, P., & Lu, X. (2014). Online traveling salesman problems with rejection options. Networks, 64(2), 84-95.
14. Kiruthika, R., Yiping, L., Laohakangvalvit, T., Sripian, P., & Sugaya, M. (2023, July). Recommendation of Sustainable Route Optimization for Travel and Tourism. In International Conference on Human-Computer Interaction (pp. 385-396). Cham: Springer Nature Switzerland.
15. Mendoza, J. E., Medaglia, A. L., & Velasco, N. (2009). An evolutionary-based decision support system for vehicle routing: The case of a public utility. Decision Support Systems, 46(3), 730-742.
16. O'Neil, M. A., Tamir, D., & Burtscher, M. (2011). A parallel gpu version of the traveling salesman problem. In Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
17. Oran, A., Tan, K. C., Ooi, B. H., Sim, M., & Jaillet, P. (2012). Location and routing models for emergency response plans with priorities. In Future Security: 7th Security Research Conference, Future Security 2012, Bonn, Germany, September 4-6, 2012. Proceedings (pp. 129-140). Springer Berlin Heidelberg.

18. Samah, K. A. F. A., Sabri, N., Hamzah, R., Roslan, R., Mangshor, N. A., & Asri, A. A. M. (2019). Brute Force algorithm implementation for traveljoy travelling recommendation system. Indonesian Journal of Electrical Engineering and Computer Science, 16(2), 1042-1049.
19. Tjandra, S. S., Setiawan, F., & Salsabila, H. (2022). Application of Genetic Algorithm s to Solve MTSP Problems with Priority (Case Study at the Jakarta Street Lighting Service). Jurnal Optimasi Sistem Industri, 21(2), 75-86.
20. Woodward, Robert & Kelleher, Edmund. (2016). Towards 'smart lasers': Self-optimisation of an ultrafast pulse source using a Genetic Algorithm . Scientific Reports. 6. 10.1038/srep37616.

Appendix

Given: 0 -> 1 -> 2 -> 3 -> 0	(4! / 4 = 6 possible combinations)				
0-> 1 -> 2 -> 3 -> 0	18	0 -> 1 -> 3 -> 2 -> 0	22	0 -> 2 -> 1 -> 3 -> 0	15
0 -> 2 -> 3 -> 1 -> 0	**11**	0 -> 3 -> 1 -> 2 -> 0	23	0 -> 3 -> 2 -> 1 -> 0	12

Figure 1: Brute Force Algorithm Example

```
0  [[    0.  24196.  23125.5 17530. ]      0 -> 3 -> 1 -> 2 -> 0
1   [24189.      0.   1714.7 11028.1]
2   [23118.5  1714.7     0.  12762.9]
3   [17535.2 10648.9 12454.6     0. ]]
```

Figure 2: Nearest Neighbor Algorithm Example

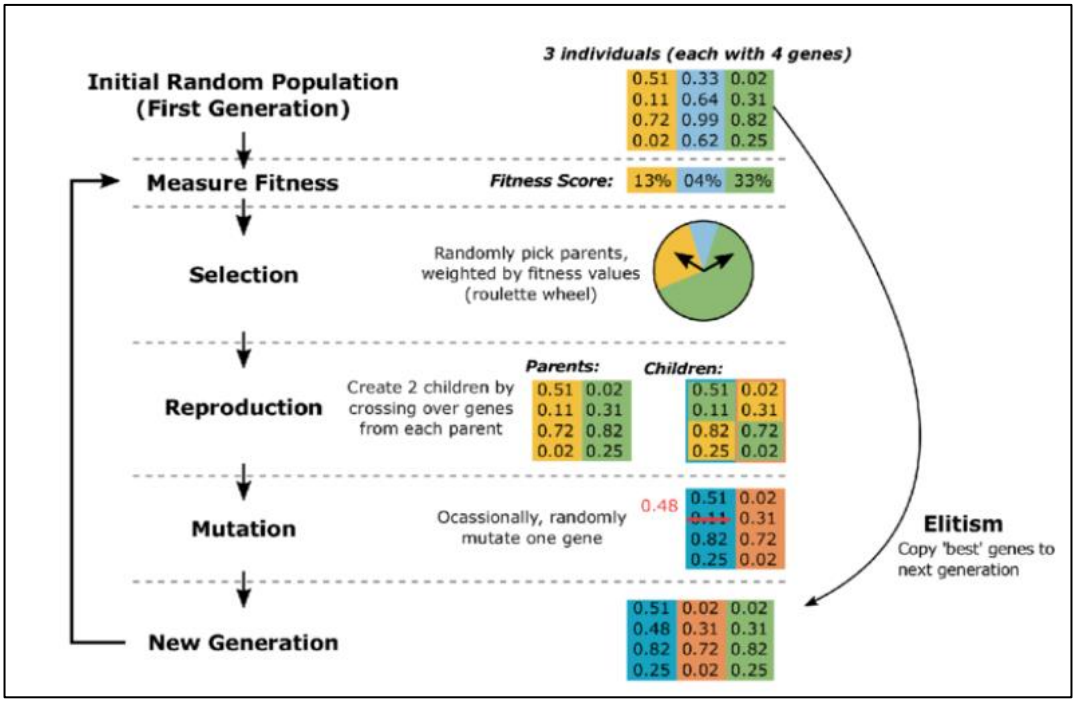
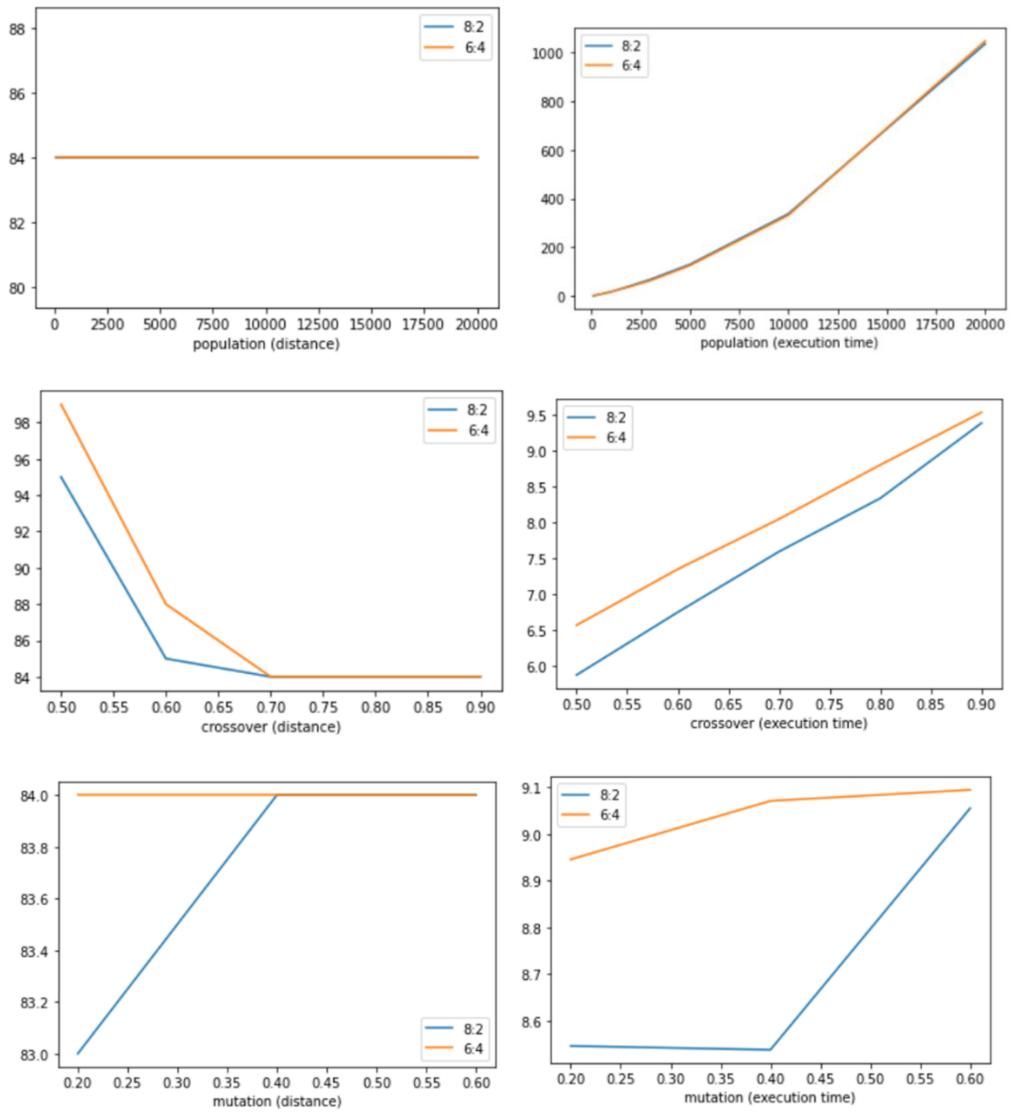


Figure 3: Genetic Algorithm Example

Genetic Algorithm was fine-tuned by adjusting four key hyper-parameters:
1) Population size (n_population), defined as the number of populations relative to the sample size,
2) Crossover rate, indicating the percentage of offspring subjected to crossover,
3) Mutation rate, denoting the percentage of offspring undergoing mutation, and
4) Number of generations (n_generations), determining the number of iterations the algorithm runs before termination.

Group Size	Station Range	Optimal n_population	Crossover Rate	Mutation Rate	Optimal n_generations
10 stations	0 to 9	2500	0.8	0.2	500
20 stations	11 to 19	6000	0.8	0.2	500

Figure 4: Genetic Algorithm Hyperparameter tuning



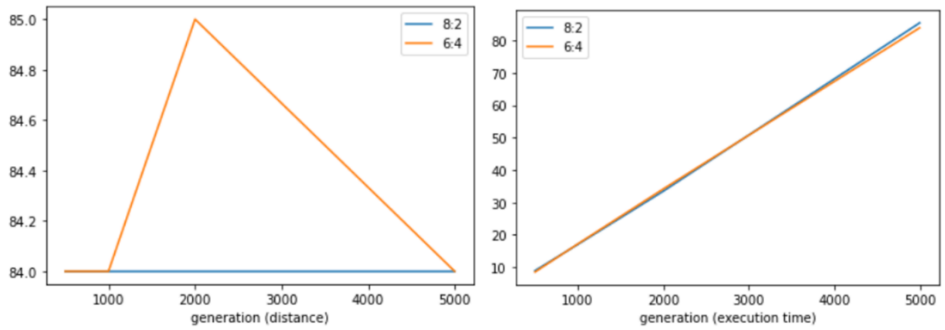
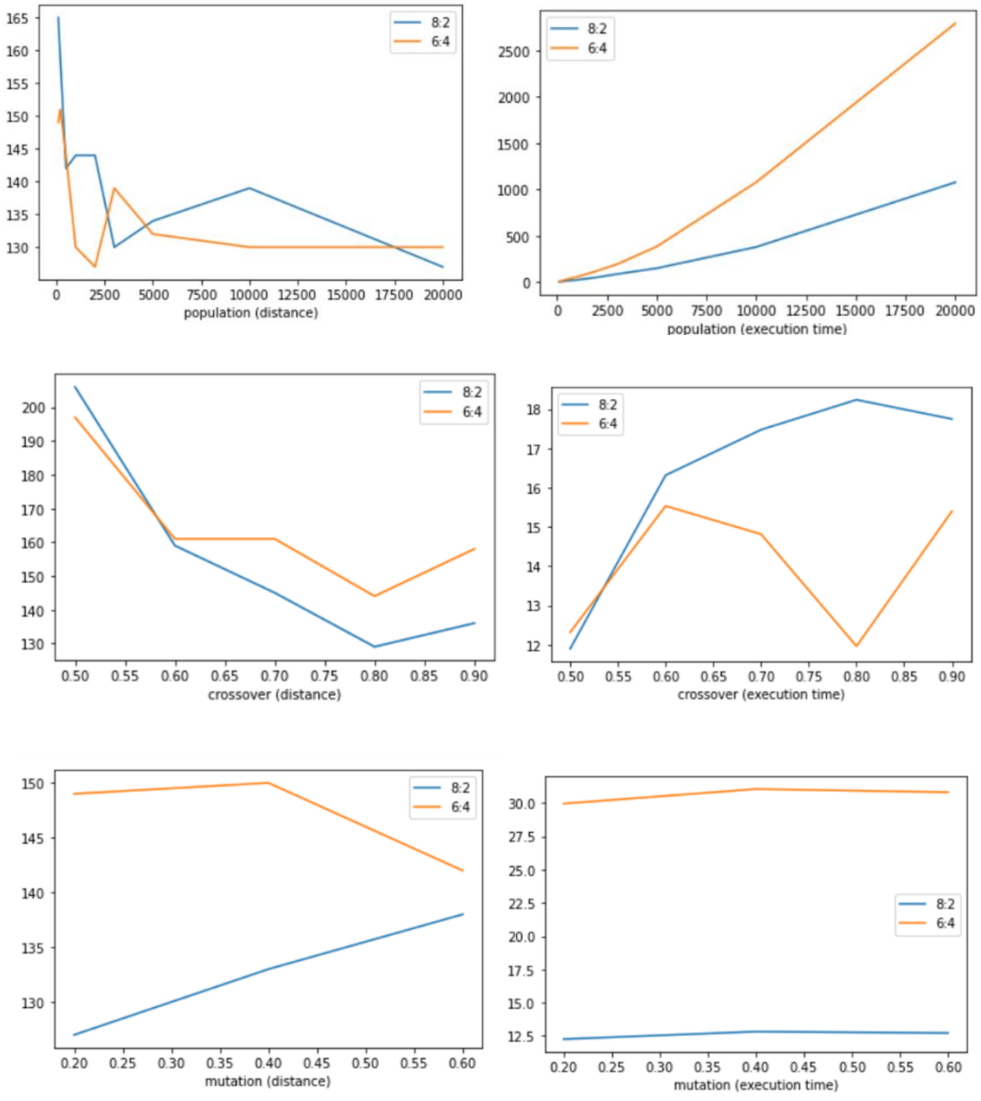
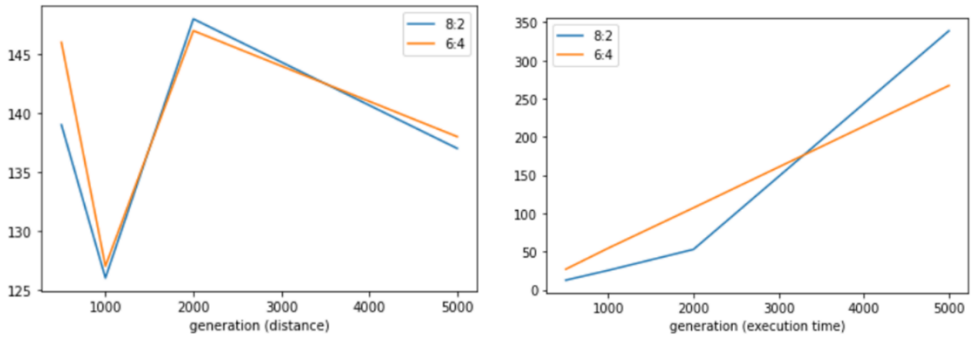


Figure 5: Genetic Algorithm Parameters Tuning 10





Figures 6 Genetic Algorithm Parameters Tuning 20

Average Distance/Duration

Total Count	Original Distance	BF Distance	NN Distance	GA Distance	Original Duration	BF Duration	NN Duration	GA Duration	BF Time	NN Time	GA Time
6195	**32.95**	**31.11**	**31.73**	**31.16**	**67.55**	**64.79**	**65.54**	**64.84**	**7.81**	**0.00**	**8.59**

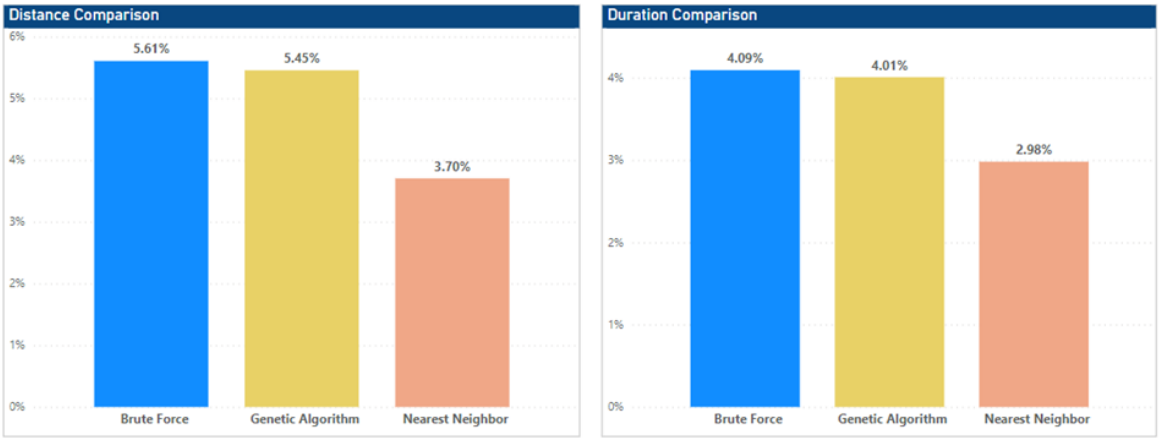


Figure 7: Total Average

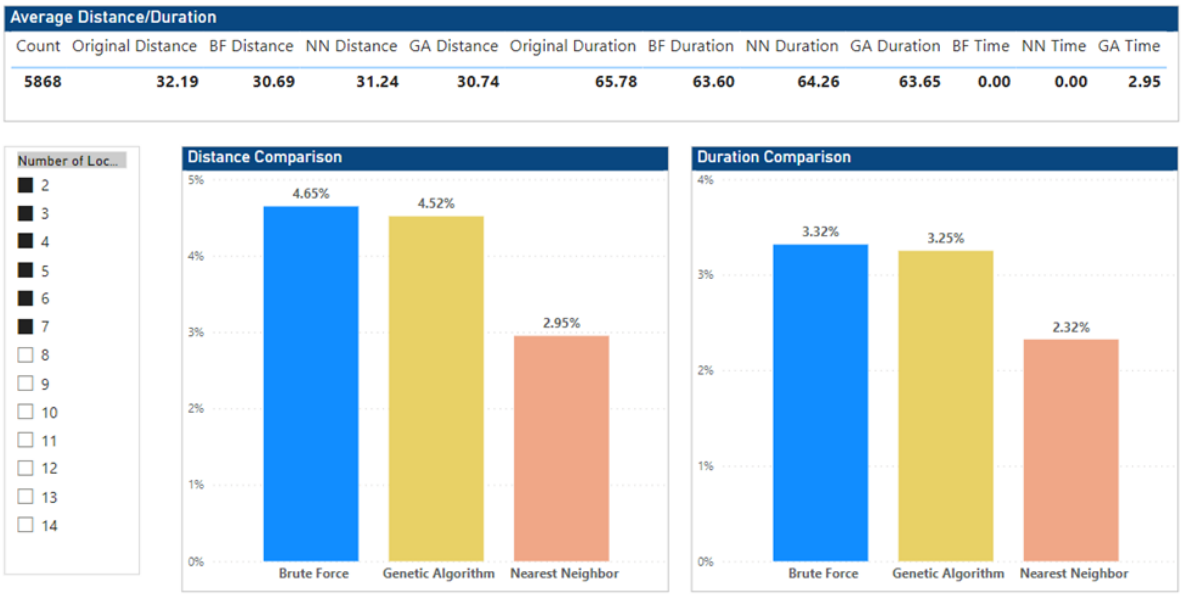


Figure 8: Number of locations between 2-7

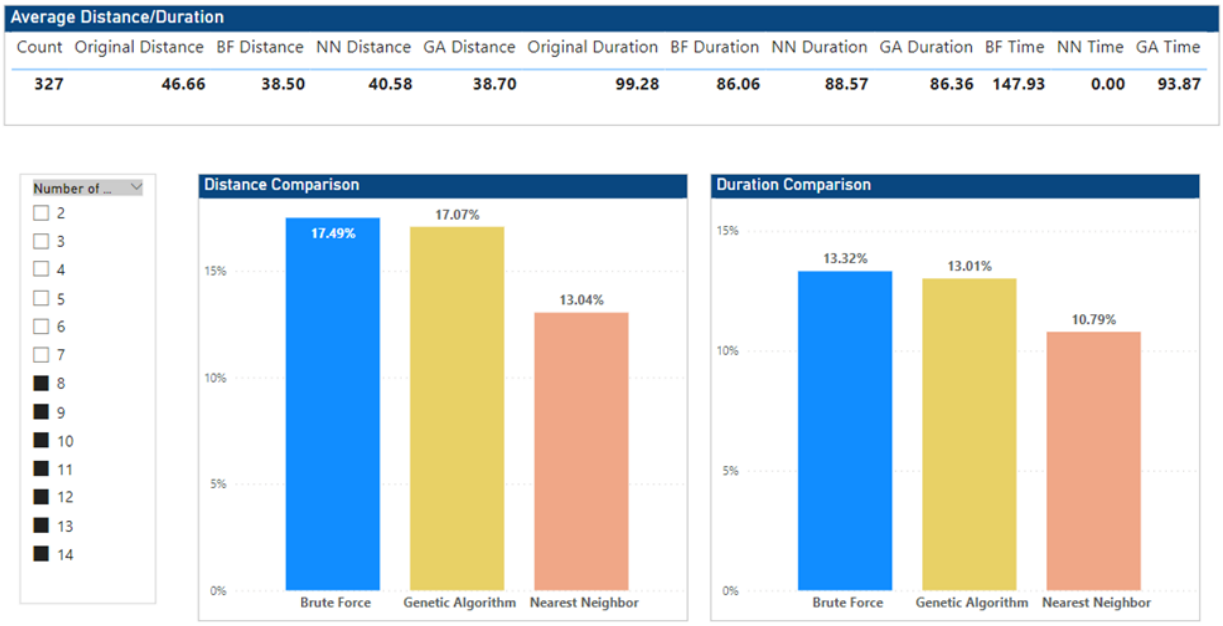


Figure 9: Number of locations between 8-14

precision	k-nearest neighbors W original sample (k=6)	k-nearest neighbors W undersamplin g (optimized)	k-nearest neighbors W oversampling (optimized)	balanced random forest W original sample (optimized)	random forest W oversamplin g (optimized)
0	0.88	0.87	0.99	0.03	0.99
1	0.62	0.52	0.60	0.37	0.76
2	0.43	0.41	0.45	0.32	0.57

3	0.98	0.78	0.68	1.00	0.92
4	0.52	0.51	0.58	0.13	0.62
5	0.39	0.61	0.80	0.01	0.83

**Higher precision indicates fewer false positives.

recall	k-nearest neighbors W original sample (k=6)	k-nearest neighbors W undersampling (optimized)	k-nearest neighbors W oversampling (optimized)	balanced random forest W original sample (optimized)	random forest W oversampling (optimized)
0	0.81	0.93	0.94	0.97	0.96
1	0.75	0.64	0.67	0.78	0.81
2	0.40	0.37	0.54	0.56	0.65
3	0.98	0.80	0.94	0.89	0.91
4	0.25	0.42	0.49	0.00	0.70
5	0.11	0.58	0.40	0.03	0.58

**Higher recall indicates fewer false negatives.

f-1 score	k-nearest neighbors W original sample (k=6)	k-nearest neighbors W undersampling (optimized)	k-nearest neighbors W oversampling (optimized)	balanced random forest W original sample (optimized)	random forest W oversampling (optimized)
0	0.84	0.90	0.96	0.06	0.98
1	0.68	0.58	0.63	0.51	0.78
2	0.41	0.39	0.49	0.41	0.61
3	0.98	0.79	0.79	0.94	0.92
4	0.34	0.46	0.53	0.01	0.65
5	0.17	0.59	0.53	0.02	0.68

**The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall. It's a single metric that combines both precision and recall into one value. The F1 score reaches its best value at 1 and worst at 0.

support	k-nearest neighbors W original sample (k=6)	k-nearest neighbors W undersampling	k-nearest neighbors W oversampling (optimized)	balanced random forest W original sample (optimized)	random forest W oversampling (optimized)

		(optimized)			
0	104	104	30000	104	30000
1	4138	104	30000	4138	30000
2	3671	104	30000	3671	30000
3	93432	104	30000	93432	30000
4	1024	104	30000	1024	30000
5	144	104	30000	144	30000

**Support refers to the number of actual occurrences of the class in the dataset. It represents the number of instances of each class in the dataset and can help in understanding the importance of a particular class. It's not a score but rather provides context about the distribution of classes.

Figure 10: Classification Models Metrics

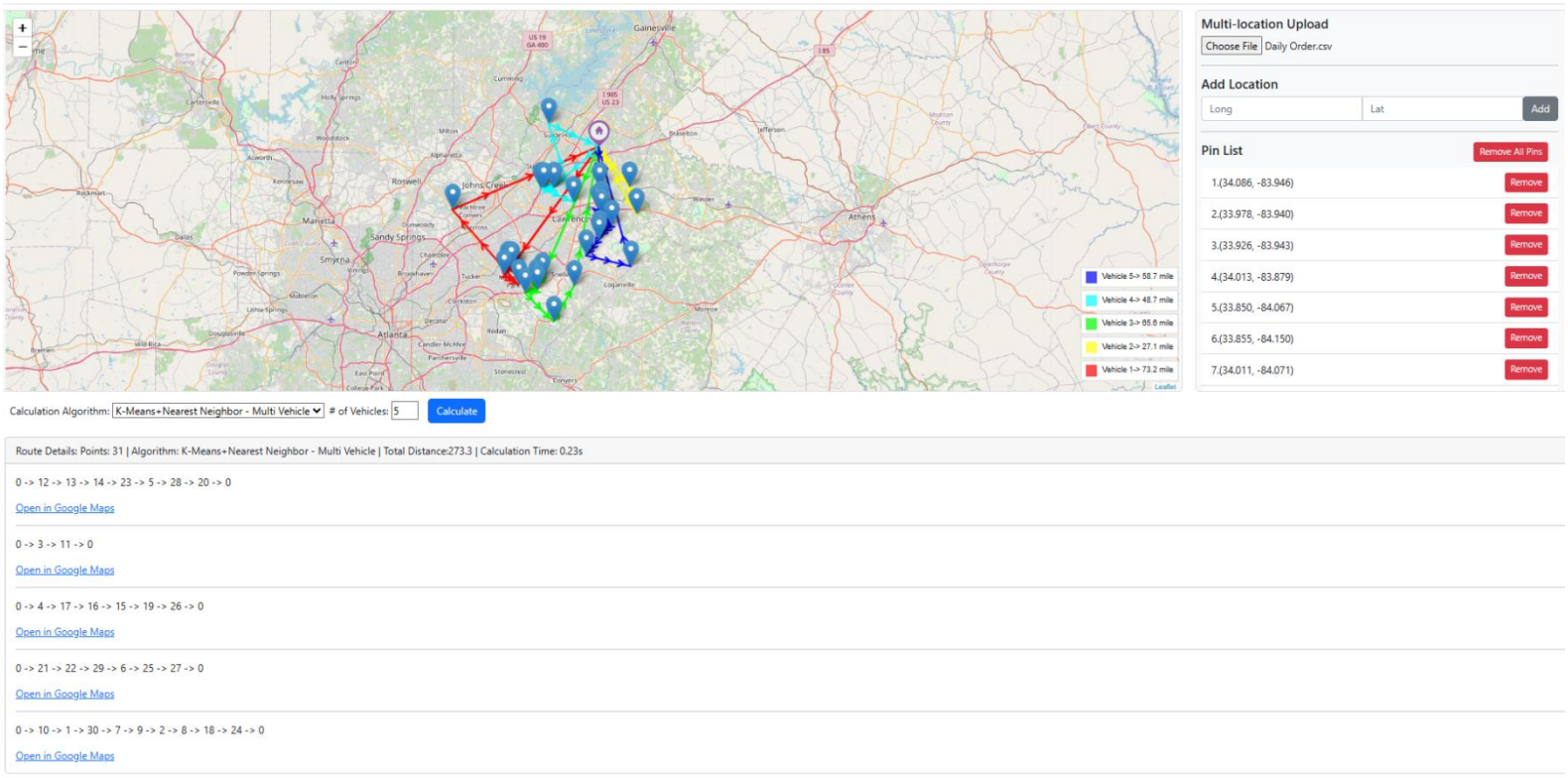


Figure 11: Demonstration of Shortest Path app