

Original Article

A Multi-Objective Ensemble Model to Suggest an Optimal Path for Private Ride-Hailing Services

K. Raghuraman¹, P. L. K. Priyadarsini², Meera Jasmine M³

^{1,3}Department of Computer Science, Srinivasa Ramanujan Centre, SASTRA Deemed University, Kumbakonam, Tamilnadu, India.

²School of Computing, SASTRA Deemed University, Thanjavur, Tamilnadu, India.

²Corresponding Author: priya.ayyagari@it.sastra.edu

Received: 01 July 2026

Revised: 12 September 2026

Accepted: 18 September 2026

Published: 27 October 2026

Abstract - Optimal path suggestion for taxi cabs being operated by private Fleet Management Services (FMS) seems a classical combinatorial problem, yet attracts researchers when more conflicting objectives are met. To improve the Quality of Service (QoS), the FMS employed a variety of strategies, compromising on some conflicting objectives, such as reducing passenger waiting time by having cabs cruise on roads, thereby increasing traffic, air, and noise pollution. To find an optimal solution for such kind of objectives, herewith a multi-objective optimization model is proposed. This model fuses the best features of Bio-inspired algorithms with Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs), providing optimal paths for FMS cabs. The proposed model has been trained and tested on the 2024 Chicago Taxi trips and Traffic datasets, which were downloaded from the official Chicago government portals. The Points of Interests (POIs) were crawled using the Google Maps API. Results showed that the proposed model outperformed the baseline model by reducing CO₂ emissions by an average of 11.4%. In addition, the time taken to pick up is also reduced to a total of 3.70 km for the spatiotemporal parameters taken. These reductions indicate that the proposed model has minimized unnecessary cruising time, waiting time, and CO₂ emissions to a greater extent. This model is modular, scalable, and provides balanced revenue for the FMS that runs in a resource-constrained environment.

Keywords - Forecasting traffic congestion areas, Graph Neural Networks, Heuristic algorithms, Recurrent Neural Networks, Ride-hailing services, Route suggestion for taxis.

1. Introduction

In metropolitan cities, traffic congestion disrupts day-to-day activities, and it has become routine for commuters to plan early to overcome it. Vehicles that run for emergency services will be allowed to move faster, but the situation becomes worse if that path gets blocked by hectic traffic. Although the infrastructure of the road network has been expanded, the usage of private vehicles on the road grows exponentially. The reason is either that the public sector vehicles were not enough to fulfil all demands or the improper timing was maintained by them. To mitigate this issue, private Fleet Management Services (FMS) are allowed to provide additional services. These service providers attract daily commuters because of their last-mile pickup and drop-off services. FMS uses strategies suggested by researchers around the world. Versatile models were developed by researchers to reduce traffic. Factors include abnormal flooding of vehicles at the same time, the working hours of popular places, etc. contributes heavily to creating congestion on roads. Commuters who travel on an emergency basis using FMS prefer to reach their destination ahead of time. At that moment,

travelling via the shortest path Having hectic traffic is a mere waste of time. Using the shortest path is different from choosing the 'Optimal' path, which is a clever choice. The term 'Optimal' is a quality metric that can be determined by providing paths that satisfy both the driver and the passenger. Normally, drivers rely on the shortest path to use less fuel, whereas the passenger demands on-time drop-off, although the kilometres travelled are high. To achieve solutions for these sorts of conflicting objectives, researchers develop models using neural networks, Heuristic algorithms, or a combination of these, etc. These models use High-Performance Computing (HPC) machines to compute and provide a balanced solution. Expecting all FMS to implement HPCs is not a feasible solution, but developing models that run on resource-light environments can be achieved. Models that are developed, tested, and have shown promising results were clustered under the literature review section.

2. Literature Review

The authors of the article [1] developed a trajectory analysis framework to address inefficiencies in traditional



travel surveys and mobile phone data for on-demand services. It identified key challenges such as GPS errors and socio-demographic biases, with historical models failing to capture real-time supply-demand changes. To overcome these issues, analytical models were divided into demand, supply, and mixed components, using techniques like map matching and trajectory clustering. Ridership was analysed through ARIMA, LSTM, and CNN models, while driver behaviour was tested using multinomial logit and SVM models. The framework was validated with diverse datasets, including NYC taxi and Uber trips, but faced limitations due to low sampling rates and a lack of high-resolution, real-time data.

The authors of the article [2] focused on forecasting the short-term passenger demand using RNNs such as simple RNN, LSTM, and GRU. To train these models, they used Tehran's TAP30 dataset, which has 64 areas with 68 features. These models were compared with the ARIMA and XGBoost models, and concluded that the simple RNN and the GRU performed effectively and produced higher accuracy when compared with the LSTM, too. To reduce the MAPE, the researchers clustered the data into 15-minute intervals, thereby improving the prediction accuracy.

To forecast the ride-sourcing request effectively, the authors of the article [3] constructed an ST-ED-RMGC model, which was trained using the NYC trip records. The proposed model learnt the complex relationships and temporal information effectively while handling the missing data cautiously. This model outperformed the LASSO and Random Forest, but it took more computation time. To enhance the prediction even further, the authors suggested including weather data and local events.

[4] The Discrete Sparrow Search Algorithm is really good at solving the Symmetric Travelling Salesman Problem. This problem is very hard to solve. The Discrete Sparrow Search Algorithm tries to find a way to visit each city one time. It uses a way of figuring out the order of the cities, and it also uses a method to make sure the population of solutions is very diverse. The Discrete Sparrow Search Algorithm has a way of looking at the problem from different angles. It uses a mutation to look at the problem in a big way, and it uses a swap operator to look at the problem in a small way.

The Discrete Sparrow Search Algorithm also uses a 2-opt search to make sure the solution is the best it can be. The Discrete Sparrow Search Algorithm was tested on 34 problems. These problems occurred in between 30 and 1,002 cities. The Discrete Sparrow Search Algorithm found the solution in 26 of these problems. The Discrete Sparrow Search Algorithm was better than five methods and 11 new methods. However, the Discrete Sparrow Search Algorithm can only solve problems where the distance from city A to city B is the same as the distance from city B to city A.

The authors of the article [5] looked at how to deal with traffic jams in cities by checking out ways to predict what will happen using neural networks like CNN, LSTM, GRU, and GCN. They tried these methods with information from cities, including Los Angeles, Beijing, and Shenzhen. These methods used tools to pay attention to important things and looked at what happened in different places and times. Even with these methods, urban traffic congestion models still had some problems. Urban traffic congestion models often do not work well in all cities. Urban traffic congestion models also assume that things do not change much in a city, which is not true. It is hard to understand how urban traffic congestion models work because they are so complicated. Sometimes they do not think about things like the weather or accidents, which makes it hard to use them to make cities better for everyone.

Researchers in the article [6] came up with an idea called the Improved Firefly Algorithm. This Improved Firefly Algorithm is used to solve a problem called the Multi-Depot Vehicle Routing Problem with Time Windows. The Improved Firefly Algorithm tries to find the distance using the fewest vehicles and takes the least amount of time for the route. The Improved Firefly Algorithm makes sure that the vehicles do not carry too much and that they arrive at the places they need to go to at the right time. If a vehicle gets to a place early, it will wait. The Improved Firefly Algorithm first decides which customers to send to which vehicles. Then it uses an old method to plan the routes.

After that, it tries to make the routes better by looking at them. The Improved Firefly Algorithm uses a way of moving around to avoid getting stuck in a bad solution. It is like the firefly is moving around and looking for a place to go. The people who came up with the Improved Firefly Algorithm tested it on some data called the Cordeau benchmark data. The Improved Firefly Algorithm worked better than some algorithms, like the traditional Firefly Algorithm, the Genetic Algorithm, and the Ant Colony Algorithm. The Improved Firefly Algorithm is very good at solving problems that involve sending vehicles to many different places.

The study [7] is about the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows. The goal of the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows is to find a way to reduce the distance that vehicles travel and the number of vehicles that are needed. The study uses a method called Ant Colony Optimisation with Destroy and Repair strategies. This method uses a rule that considers the direction vehicles are going, how urgent the deliveries are, and how long it takes to make each delivery. The destructive part of this method gets rid of the customers who are not as important. Then the repair part puts these customers back into the route in a way that tries to save money and reduce regret. The researchers tested this method on some problems and found that it worked really

well. In fact, the Ant Colony Optimisation with Destroy and Repair strategies was able to reduce the distance that vehicles travelled by up to 36 percent compared to a basic version of this method. The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Windows method also worked well, as did some other top methods. However, sometimes it was not as good at reducing the number of vehicles that were needed. Also, it was hard to compare the results of this study to those of other studies because the computers used were different.

The authors in [8] looked into the challenges of predicting traffic flow in edge IoT environments by creating a smart hybrid framework called PROMOTION. They tried to handle data and improve hyperparameter tuning in large road network graphs. To do this, they used the Local Outlier Factor, or LOF, in pre-processing to remove nodes and shrink the graph size. Then they processed the cleaned data with a Graph Convolutional Neural Network, or GCNN. The GCNN used normalization for aggregation.

The authors of the article [9] have identified that traditional methods and PSO struggle a lot in solving the Vehicle scheduling problem for ore companies. These models often get stuck in local optima. To overcome this issue, these authors developed a modified PSO by introducing four methods, such as elite reverse, adaptive weights, a jump-out, and mutation. These methods increased the search effectiveness and population diversity. The proposed model's operational cost was found to be lower when compared with practical 10 VSP simulations and tested with advanced algorithms using 12 benchmark functions. This model could not be generalized as it has been constructed using a private dataset, and its optimization made the model hang between speed and accuracy.

The authors focused on the Energy Minimising Vehicle Routing Problem by aiming to reduce energy use based on vehicle distance and load [10]. Four hybrid algorithms were developed, combining capacity-restricted K-Means/K-Medoids clustering with Best-Worst Ant System (BWAS) and Variable Neighbourhood Search (VNS). Evaluated on CVRPLIB benchmarks, two frameworks were tested: Free Ant (which uses clustering for pheromone start points) and Restricted Ant (which breaks the problem into smaller TSPs). Free Ant outperformed Machine Learning-Tabu Search (ML-TS) on "Set A" with a 7.08% better solution gap, but required more computational time and parameter tuning. Restricted Ant was faster but slightly less accurate. The proposed strategy also suffered from Scalability and parameter sensitivity.

The authors [11] have constructed the Multi-View Dynamic Graph Convolution Network (MVDGCN), which addresses the shortcomings of traditional traffic prediction models by capturing complex, multi-periodic temporal

patterns and dynamic spatial relationships. It uses three encoder-decoder modules for hourly, daily, and weekly views, integrating a Coupled Graph Convolution Module (CGCN) and a Dynamic Fusion Module (DFM) for adaptive feature weighting. The proposed MVDGCN was evaluated using RMSE, MAE, PCC, and MAPE on the NYCTaxi and NYCBike datasets and compared with eight baseline models, including CCRNN and XGBoost. MVDGCN outperformed all baselines with a maximum of 12.9% improvement in RMSE compared to CCRNN. The ablation studies validated the contribution of each component, but the complexity increases the computational cost, which has raised interest in model pruning to improve efficiency.

The article [12] addresses the issue of creating novel hybrid metaheuristic algorithms that will be capable of solving complex optimization tasks by finding a balance between the exploration of a larger space and the use of search algorithms. The developed models include a combination of several computational approaches, including multi-ant systems combined with local search algorithms, large neighbourhood search with Genetic Algorithm (GA), and Ant Colony Optimization (ACO) applied to the multi-objective Vehicle Routing Problem (VRP). Even though the advanced combinations of these models benefit from the strengths of each algorithm in order to generate optimal results, the article lacks important information for making an appropriate evaluation. In particular, the exact problem that is to be solved is not mentioned at all, and the information about testing datasets (their number, features, etc.) is absent as well.

The Vehicle Routing Problem with Time Windows is really tough to solve. This is because it is an NP-hard problem, and it takes a long time to find a solution. The Self-Competitive Particle Swarm Optimisation algorithm is a way to solve this problem [13]. It is also called ScPSO. The ScPSO algorithm works by letting the particles that are not doing well learn from the ones that are doing well. It also uses a weight that changes to help the particles search for the best solution. The algorithm starts with a solution and then makes it better by looking at the solutions that are close to it. The ScPSO algorithm was tested on a lot of problems. These problems are called Solomon's 56 benchmark problems. The ScPSO algorithm did better than or just as well as other algorithms like S-PSO-VRPTW, HGA, MS-EAS, and TSA. It even did better than the solutions that were already known. This is especially true when the customers are in places. The ScPSO algorithm is really good because it has increased the profit from 48.5% to 71.8%, more than the basic PSO algorithm. However, it is still hard to set up the algorithm and to make it work well.

The researchers in the article [14] constructed the GMTP framework to figure out how long it takes to travel by vehicle. They wanted to understand the roads and how traffic changes at different times and places. The GMTP framework uses a

tool called GATv2 to see how different areas are connected, and it also looks at how often buses and trains run. It uses something called BERT to understand the path that vehicles take. The framework also looks at things like rush hour, day of the week, etc. The GMTP framework was tested on a lot of data from Porto and Beijing. It worked better than models like Traj2vec and Transformer. The only problem with the GMTP framework is that it only uses GPS data and does not use information, such as where people check in.

The authors introduced the Hierarchical Spatio-Temporal Graph Convolutional Neural Network (HSTGCN) to address traffic speed data imputation, especially under complex "blockout" and non-random missing patterns [15]. Tested on real-world datasets (PEMS-BAY and Seattle), HSTGCN uses a dual-layer architecture: a macro layer that clusters nodes using Pearson correlation for learning from non-adjacent roads, and a road layer combining adaptive adjacency matrices with temporal attention mechanisms. The model fuses feature across layers for final imputation. Evaluated against deep generative and tensor decomposition models (BGCP, DSAE, MIWAE, BTMF, GA-GAN), HSTGCN consistently achieved higher accuracy, maintaining strong performance even with 70% missing data. However, feature interpretability remains an area for future research.

The researchers have developed a HetGAT, a heterogeneous graph neural network, that addresses the shortcomings of traditional User Equilibrium Traffic Assignment Problems (UE-TAP) for large urban networks [16]. Using a two-encoder design-combining a Virtual Graph Encoder for capturing long-range dependencies with Real XG for local processing-HetGAT effectively estimates link flows, even with up to 40% missing origin-destination data. The model was validated on Sioux Falls, East Massachusetts, and Anaheim datasets, using a composite loss function that balances supervised accuracy and physics-based constraints. Compared to FCNN, GAT, and GCN, HetGAT achieved up to 56.4% lower mean absolute error in the EMA network. While robust and efficient, its application is limited to static traffic scenarios.

To handle the balance between increasing service requests and rising operational costs, the authors of the article [17] proposed a novel logistics distribution model through the application of an Improved Fireworks Artificial Bee Colony (IFW-ABC) algorithm. The model introduced the use of an adaptive search strategy with real-time variable control for step size and search space. The focus was on minimizing an all-inclusive cost function for transport, fixed costs for a vehicle, and time, while observing strict vehicle load and time-window constraints. The authors successfully optimized the distribution model through the use of mathematical functions to control the balance between a global search and optimal locality. The simulation and validation runs were based on the Solomon benchmark problem and a real-world case for

reference. In comparison to the other form of the ABC model, the IFW-ABC was proven to be more effective, with an average delivery cost of 3.2 % and a 15.4 % reduction in the total cost in the case study. The study appreciated the vast improvements regarding the efficiency and utilization of vehicles, but acknowledged that, due to the ABC model, the same local optimality and universal parameter problems were present. To enhance the Quality of Service in multiple trips and depots, using the complex Fuel-Electric Hybrid VRP, the authors of the article [18] have developed an improved DSSA by introducing the tent mapping and key encoding techniques for initialization and discrete variables. This model is tested against the Cordeau benchmarks and proves its ability to reduce the total distribution cost by 4.22% and path distance by 5.84%. Even then, the proposed model has limitations, such as the assumption that vehicles are moving at a constant speed, which is impossible in the real world.

Accurate forecasting of traffic volume and speed is always challenging. The authors of the article [19] have addressed this issue by developing a Spatial-Temporal framework called the GNN-DW-LSTMA model. This model uses MLPs to enhance the Quality of learning and to reduce the over-smoothing problem. This model is compared with the baseline models, such as STGCN and the DCRNN, on two datasets, PeMSD7 (M) and PeMSD7 (L). It outperformed all the baselines in accuracy and reduced the MAE values, too. Hence, it is considered robust in traffic forecasting by the developers

To effectively forecast traffic prediction from various perspectives, the authors of the article [20] created an MSTFF model. Since it is a time-series problem where the model should be trained on continuous timestamps, it uses a sliding window technique to separate normal traffic and sudden blocks. This model is trained and tested with 4 datasets, and it performed well against 12 baseline models. Despite its success in forecasting congestion, the model is limited to a one-hour forecasting window. The authors thus concluded by confirming that this model, in the future, will improve its accuracy if local events and weather conditions data are included to manage city traffic.

Researchers in [21] constantly focus on developing hybrid models to mitigate the problem of acquiring the Spatial-Temporal correlations, which were disturbed by the complex road networks and flow dispersion at junctions. The authors of this article addressed this issue by constructing a model called the STGM, which uses a GCN and a temporal attention-enhanced LSTM. The model used four different graphs, such as Distance, transfer, speed similarity, and occupancy matching. This model is thus evaluated using the PeMSD8 and Xuancheng datasets. This model has outperformed ARIMA and advanced GNNs such as MSTGCN, ASTGCN, and STSGCN in metrics including MAE, RMSE, MAPE, and NRMSE. Researchers thus

concluded that the complexity of real-world traffic is always challenging.

The model GNN-SDE, which is a graph neural network that estimates shortest distances very fast, even faster than conventional methods such as Dijkstra's, has been proposed in [22]. It makes use of information from node (coordinates, hop distance) and edge (travel time) features and propagates this information using layers in a graph neural network. The Performance of the model is optimized using the minimization of the Mean Absolute Error Loss function, while shortest path estimation is performed using an algorithm similar to Dijkstra's algorithm. The validation of the model has been done on both synthetic grid graphs (1,000 - 100,000 nodes) and real-world road networks in Manhattan and Hillsborough County (with floods as risk), where the GNN-SDE showed a Pearson Correlation of over 0.979.

[23] The Spatial-Temporal Graph Wavelet Neural Network (SWNN) aims at overcoming difficulties in traffic flow forecasting, such as complicated road networks, insufficient spatiotemporal data, and over-smoothing in GNNs. The SWNN approach is a combination of Graph Convolutional Neural Networks to extract spatial features and Long Short-Term Memory layers to analyse temporal aspects. The wavelet transforms act as an activation function, whereas historical context attention helps in capturing long-range dependence and preserving the integrity of the signal. The SWNN was tested on real-world data sets such as PEMS04 and the California traffic dataset and demonstrated the best results compared to other classical and recent methods (such as T-GCN, GMAN, STGCN, and WaveletCNN), reaching smaller RMSE and MAE.

The everyday hectic traffic congestion in Bangalore city is taken as the study area by the authors of the article [24]. To overcome this issue, these authors have constructed an ST-GNN model that integrates a multi-agent system to forecast the bottleneck and to optimize the signal. To train this model, they have used the dataset of Bangalore's traffic data for a period of 200 days, which consists of features such as volume, speed, roadwork, etc. This model got an F1-Score of 86.99% and suggested speed and volume as the important features. Although this model performed effectively, it lagged in throughput or queue length, which will be addressed as future work.

The researchers [25] have developed a Physics-Guided Spatiotemporal Graph Neural Network (PG-STGNN) to enhance urban intersection traffic flow forecasting through the incorporation of physics-based constraints, such as queue length and volume-to-capacity ratio, into a four-step deep learning approach. In PG-STGNN, Graph Convolutional Networks (GCNs) have been applied for spatiotemporal learning, whereas Long Short-Term Memory (LSTM) networks have been used for learning temporal dynamics

based on actual traffic data collected from a signalized intersection in Beijing. PG-STGNN outperformed other baseline algorithms (KNN, ARIMA, RF, GNNs) with respect to Mean Absolute Percentage Error (MAPE) and prediction error by a margin of 9.452% and 9.8%, respectively. The study noted limitations regarding computational complexity and interpretability, recommending future transitions toward multimodal data integration.

Some of the problems encountered in conventional optimization techniques, such as poor convergence rate and falling into local minima, were overcome through enhanced Tent mapping and a dimensional reverse learning approach. In the study [26], an Improved Whale Optimization Algorithm with Recurrent Neural Networks (IWOA-RNN) is utilized to cope with the nonlinear dynamics of traffic flow analysis by using actual highway data of Guangzhou. The parameters for optimizing the RNN include population size and adaptive weights. When compared with other models such as Support Vector Machine (SVM), KNN, CNN-LSTM, and hybrid techniques (PSO-SVM, GWO-ANN), IWOA-RNN outperformed all these in terms of accuracy by getting the lowest MAE, MRE, MSE, and RMSE values and improving 9% performance of the basic RNN.

Researchers [27] addressed the multi-depot, multi-salesman problem by integrating Ant Colony Optimization (ACO) with the Artificial Bee Colony (ABC) algorithm, utilizing a tensor ($I \times J \times K$) method for efficient route and cost management. ACO generated an optimal pheromone matrix, serving as input for the ABC process, which minimized route costs via bee-inspired agents. The model was tested on multiple TSPLIB benchmarks with 2-10 salesmen. The hybrid AC-ABC algorithm outperformed other methods (AC-PGA, IPGA, TWPS), achieving a 1.48% variation and reducing execution time by 580%. For a single objective, the model performs well, whereas further research should be done for multi-objective problems and real-world applications like robotics and delivery scheduling.

The research conducted in [28] addressed the problem of traffic signal prediction in a large-scale urban network, crucial for automated traffic control. Two versions of the Spatial-Temporal Graph Convolutional Neural Network (STGCNN) model have been designed, including a basic one and an improved one with Signal Plan IDs and traffic state. For the investigation, the urban network of Tel Aviv and the arterial road Even Gvirol have been used, providing speed information based on Bluetooth sensors and signal plans of the city. Traffic is represented in terms of spatiotemporal graphs employing Chebyshev expansion. STGCN models perform better than traditional LSTM and RNN models. The addition of temporal external features and fixed cycle steps has led to improved results, although certain limitations are associated with partial Bluetooth data and mismatched data. To address the issues arising from unreliable and insufficient information

in traditional traffic flow forecasting in the context of flooding in an urban environment, the authors of the article [29] presented the PINN-GNN model that uses Graph Neural Networks (GNNs) and Physics-Informed Neural Networks (PINNs). Features of the model include a field effect module and the use of a modified LWR traffic equation with a composite loss function. The model is tested on historical traffic data and rainfall data in London and then tested in terms of scalability in 16 cities. The model outperformed existing baselines by 17.5% in heavy rain (in terms of MAPE, MAE, and RMSE). Although very accurate in the case of traffic, it demands much computing power.

3. Proposed Model

Based on the nature of the problem, a multi-objective optimization problem is formulated to assign cabs to the user requests, with the following objectives and corresponding constraints.

1. min f (Waiting_Time+) - To reduce the waiting time of riders, allowing cabs to cruise on roads leads to traffic, air, and noise pollution. Hence, it should not exceed the threshold.

$$\min f (Waiting_Time) = T_{Pickup_{n,r_n}} - T_{Request_{n,r_n}}$$

where,

$T_{Request_{n,r_n}}$ - Time of Request made by the Rider 'n'

$T_{Pickup_{n,r_n}}$ - Time the cab took to reach the Rider 'n'

Subject to:

$$T_{Pickup_{n,r_n}} - T_{Request_{n,r_n}} \leq T_{Threshold}$$

(where $T_{Threshold}$ = 15 minutes, for example)

$$T_{Pickup_{n,r_n}} \leq T_{Request_{n,r_n}} + C$$

(where C is a buffer time to reach the pickup location)

$$Dist(Taxi_{pos}, Pickup_{rn}) \leq 10 \text{ km}$$

2. min f (Cruising_Time) - Reducing fuel consumption by making the cab idle increases the waiting time of the passenger booked for a ride.

$$\min f (Cruising_Time) = \sum_{r \in R} \sum_{v \in V} CT_{v,r}$$

where,

CT_v denotes the Cruising time for picking up ride r for vehicle v

Subject to:

$$\text{for each ride } r, \sum_{\text{for each vehicle } v} U_{v,r} \leq 1$$

where $U_{v,r}$ is an Indicator random variable for vehicle v assigned to ride r

$$Distance_{(VehicleLocation, PassengerPickup)} \leq Distance_Max$$

where Distance_Max is the maximum distance the vehicle can travel to satisfy the overall minimization of cruise time

$$T_{StartCruise} \geq T_{LastDrop_offpassenger} + \Delta t$$

where Δt is the time taken to change the taxi flag from booked to available.

$$3. \min f (Idle_time) = \sum_{\text{for each driver } d} T_{idle}(d)$$

But if too much idle time is reduced by keeping the cab busy, the driver's resting time will be disturbed.

Subject to:

- Maximum Work Condition:

$$T_{RunningTime}(d) \leq T_{Max_WorkingHours} \text{ for each driver } d$$

- Minimum Rest Condition:

$$T_{Idle}(d) \geq T_{Min_RestTime}(d), \text{ for each driver } d.$$

4. min f (Travel_Time) - During peak hours, using the shortest path to reduce travel time may lead to more fuel usage if the chosen path is stuck with hectic congestion. So, we try to minimize the travel time, which includes the time spent in traffic. Hence, for each taxi:

$$\begin{aligned} \min f (Travel_Time) &= \sum_{(i,j) \in E} (T_{Travel(i,j)} + T_{Traffic(i,j)}) \\ &\quad * X_{(i,j)} \end{aligned}$$

where,

E is the set of Edges of the form (i,j),

i & j are the consecutive nodes along the path and $i \neq j$

$$x(i,j) = \begin{cases} 1 & \text{If the road between i and j is taken} \\ 0 & \text{Otherwise} \end{cases}$$

$\forall i,j$ where $i \neq S$ and $j \neq D$

Once the above multi-objective optimisation problem is formulated, we designed an integrated architecture, as given

in Figure 1, to provide an optimal assignment of vehicles to users, followed by finding the best route for the ride. The proposed model in Figure 1 consists of three modules: (i) Cab and User mapping, (ii) Prediction of traffic and POIs to avoid congestion, and (iii) Optimal routes generated using GNNs. The first phase focuses on matching the passenger with the cab that is idle or cruising nearby. This is done by solving the above-mentioned multi-objective optimisation problem using the meta-heuristic model, the Discrete Sparrow Search algorithm. The second phase identifies the congested areas to be avoided when reaching each passenger's destination from their respective sources using the pre-trained LSTM model. The last phase utilises Graph Neural Networks (GNNs) to forecast alternative paths to the destination without any hurdles. The modules are clustered into two groups: the first cluster tries to minimise the idle time, waiting time, and cruising time in the first module, and the second cluster tries to reduce travel time, mapped with the second and third modules of the proposed framework. The overall increase in service quality can be achieved by the entire model. Each module will be discussed in the subsequent sections. To train and test the model, the Chicago Taxi Trip Dataset [30] and its Traffic dataset have been collected for the year 2024 [31], and the Points-of-Interest (POI) or Hotspots were crawled using the Google Maps API. These datasets have undergone many pre-processing steps that were explained in the corresponding section. The overall process flow of this architecture is shown in Figure 2.

Phase 1: Data Pre-Processing: For any model to be trained, tested, and launched, it demands historical datasets that are cleaned and processed. To train and test the proposed model, herewith 3 datasets were collected and pre-processed. They are the Chicago taxi-trip dataset, the traffic dataset, and the POIs dataset. These datasets were perfectly aligned for the July 2024 period, and the “flash cab” FMS taxi trip records were taken as inputs. These datasets underwent initial processing, such as finding missing values for each feature and using interpolation techniques, and type conversions were made. The Chicago taxi trip dataset has been identified with absurd records, i.e., records with 0.0 miles travelled; there were around 10,217, and merely 33 trips had 0.0 seconds recorded, which means that the trip has the same source and destination geo-coordinates. Such records were truncated from the dataset. The Chicago traffic dataset has around 2,042,169 sensor speed errors, which are (-1), indicating whether those sensors failed to capture a valid reading due to network loss or whether those sensors might have gone for maintenance during that time. These records were removed from the dataset in this phase. Thus, after pre-processing, the Chicago Taxi trips dataset for the month of July 2024 has 22,346 valid trips. Similarly, the traffic sensor dataset has 5,83,188 valid speed records. Using KD-Tree and spatial binning methods, the POIs' geo-coordinates were clustered, and has identified that 84,219 POIs were available for July 2024. All the features are standardized and cleaned so that the

model can integrate the 3 datasets clearly and understand the patterns, such as on which day, time, and which POIs and intersections were busy. Using this knowledge, the model can start to forecast the future whenever a trip is initiated.

Phase 2: DSSA Phase: The following objectives were mainly focused on in this section: $\min(f(\text{waiting_time}))$, $\min(f(\text{idle_time}))$, $\min(f(\text{cruising_time}))$, subject to each function's constraints. These objectives come to a trade-off space when each customer is mapped to each cab, and each cab is utilised optimally to get booked for a ride. Thereby, cruising time and idle time are implemented properly. To map the driver with the rider, herewith the framework uses the bio-inspired algorithm called the Discrete Sparrow Search-based Dispatch Engine operating over the previously constructed Chicago transportation graph, transforming static pre-processing outputs into a dynamic, topology-aware fleet optimisation system. The core optimisation loop executes a discrete variant of Sparrow Search, where each taxi represents a “sparrow” candidate competing to service a user request. For every incoming trip, the pickup location is mapped to the nearest graph node using a KD-tree, and for each taxi, its current position is similarly mapped. The shortest path road distance is computed via Dijkstra over the graph using edge length as weight; if routing fails, a fallback Manhattan-style geodesic approximation is used. A fitness function is calculated as weighted road distance plus a load-based penalty, discouraging over-utilisation of the same vehicle. The taxi with minimum fitness is selected, and its state is updated to the drop-off location, thereby chaining dispatch decisions sequentially-this constitutes the discrete, iterative search mechanism tailored for fleet allocation rather than continuous parameter optimisation.

Tent chaotic mapping is introduced as an “explorer”, generating bounded chaotic sequences that enhance entropy across iterations; here, it ensures non-deterministic yet controlled variability within the optimisation cycle, preventing predictable dispatch repetition patterns. Stochastic key exchange, a technique used as a “translator”, further strengthens system security by generating a SHA-256-based cryptographic key derived from taxi ID, trip ID, chaotic state, and nanosecond timestamp entropy, embedding tamper-resistant authentication metadata within each dispatch record. In the earlier technique, the DSSA search space is widened, and thousands of different taxi-passenger combinations are tried. Before the matching begins, this module finds the possibilities to match the passenger. In the latter, the matchmaking happens, i.e., the “decoder” finds which passenger gets the “best” cab first. Once a taxi is assigned to a person, it will be removed, from the pool of taxis to satisfy our condition that: $\sum_{r \in R} U_{vr} \leq 1 \forall v \in V$. The bio-inspired algorithm eventually forces too many idle taxis during peak hours to get the pickups in a parallel manner, including the taxis that are cruising for pickups. Therefore, this model reduces the time of:

$$F(\text{fitness}) = w1 \sum_{r \in R} (T_{\text{Wait}}, r) + w2 \sum_{v \in V} (T_{\text{Idle}}, v) + w3 \sum_{v \in V} (T_{\text{Cruise}}, v)$$

where $w1$, $w2$, and $w3$ are the weighting factors that are introduced to find a trade-off between the objectives taken by this phase. These weighting factors ensure that the primary objective of minimizing customer waiting time is met without degrading drivers' downtime and increasing the cruising time. Therefore, this phase provides overall satisfaction through balanced workload, reduced waiting time, and the emission of CO_2 is reduced by avoiding cruising taxis on roads.

Phase 3: LSTM Phase: This module is constructed to understand the traffic zones and the POIs' locations when more people are attracted to them. To forecast the traffic horizons, this LSTM uses the 80/10/10 methodology, where 80 percent is for training, 10 percent for validation, and another 10 percent for testing. It consists of 2 stacked layers having 32 units, each with a dropout regularization of 0.2. The model uses the "Adam" optimizer and MSE. The inputs are normalized using the MinMaxScaler. It calculates a congestion Index ranging from (0 - 1), where '0' is heavily congested and '1' is a freeway. To forecast the POIs, another LSTM uses a 2-stacked layer consisting of 128 and 64 units, each with a dropout of 0.3 and 0.2, respectively. These two layers use the "Adam" and "Huber loss" optimizers. The POIs' records are expanded further to hourly records, and an interruption score is created. Finally, the forecasted locations are processed using DBSCAN to find which zones are highly active.

Phase 4: GNN phase: Once the previous module is ready to forecast the traffic congestion and the POIs, which occurs during the beginning of each trip, this module uses the information received into a unified format and as batches that will be forwarded to 3 widely used Graph Neural Networks namely, the Graph Attention Networks version 2 (GATv2),

Spatiotemporal Graph Convolutional Neural Network (STGCNN), and the Diffusion Convolutional Recurrent Neural Network (DCRNN). These models have already shown promising outputs that were discussed in the literature survey section. These GNNs have the ability to generate the path while considering the spatiotemporal data from the 3 datasets. To create a path that can be implemented in real-world that 'osmnx' library is used to crawl the original road networks of the city of Chicago. Once the city map is fixed, the GNNs are constructed with their respective layers. To begin with, the GATv2, it is constructed with 2 distinct layers: the Convolution layer and a multi-head attention layer to capture the complex neighbourhood relationships and data patterns. It integrates with a Dijkstra shortest-path algorithm to generate a path that avoids POIs and congested zones. It is trained with a batch size of 32 and 64 units for 100 epochs. Early stopping of 15 epochs with a learning rate of 0.0005 is also implemented. The STGCNN model is constructed using the GCNConvLayer with LSTM to learn the spatiotemporal patterns clearly. It uses a 12-step input sequence to predict a multi-step forecast of 3-6 steps, i.e., 15 to 30 mins. Early stopping at 10 and 20 epochs was implemented with a learning rate of 0.001. The DCRNN is constructed with a bidirectional Diffusion convolution with Gated Recurrent Units (GRU) to read the long-term temporal features in the city road network. Here, the input sequence length was 12 to perform a multi-step forecast of 3, 6, and 12, i.e., 15, 30, and 60 minutes. Early stopping at 12 epochs is implemented with a learning rate of 0.001. All the 3 models used the 'Adam' optimizer and the MSE loss function. These GNNs were trained and tested with the given inputs to accurately generate the path, based on the guidance of our previous module, which predicts traffic and POI information. By steering the route to reach the destination, avoiding the POIs and the congested locations, each of the GNNs generates an individual path. Thus, the generated paths may sometimes be longer than usual for some trips, but greatly avoid POIs and congested zones.

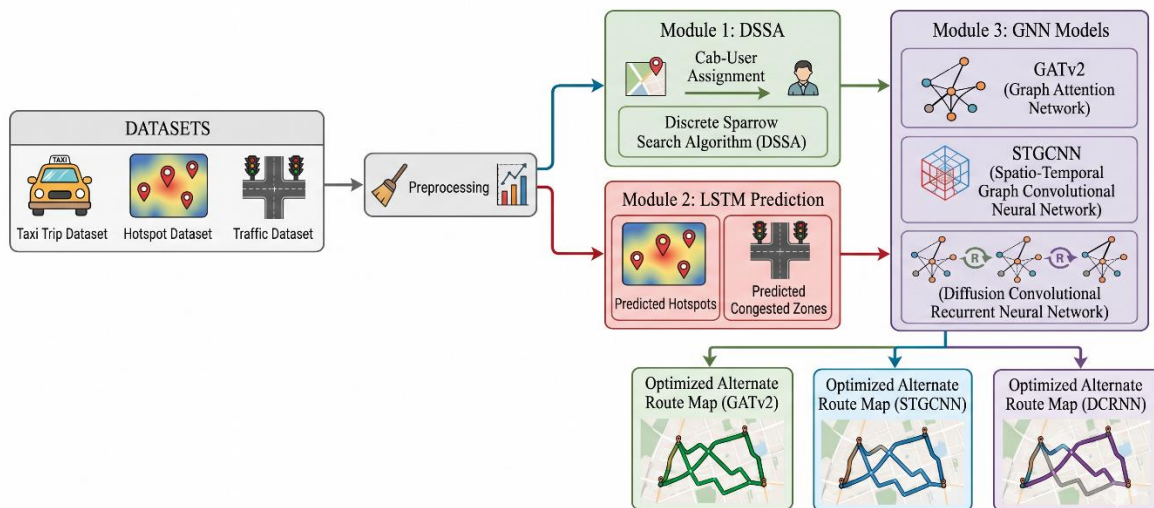


Fig. 1 Proposed model

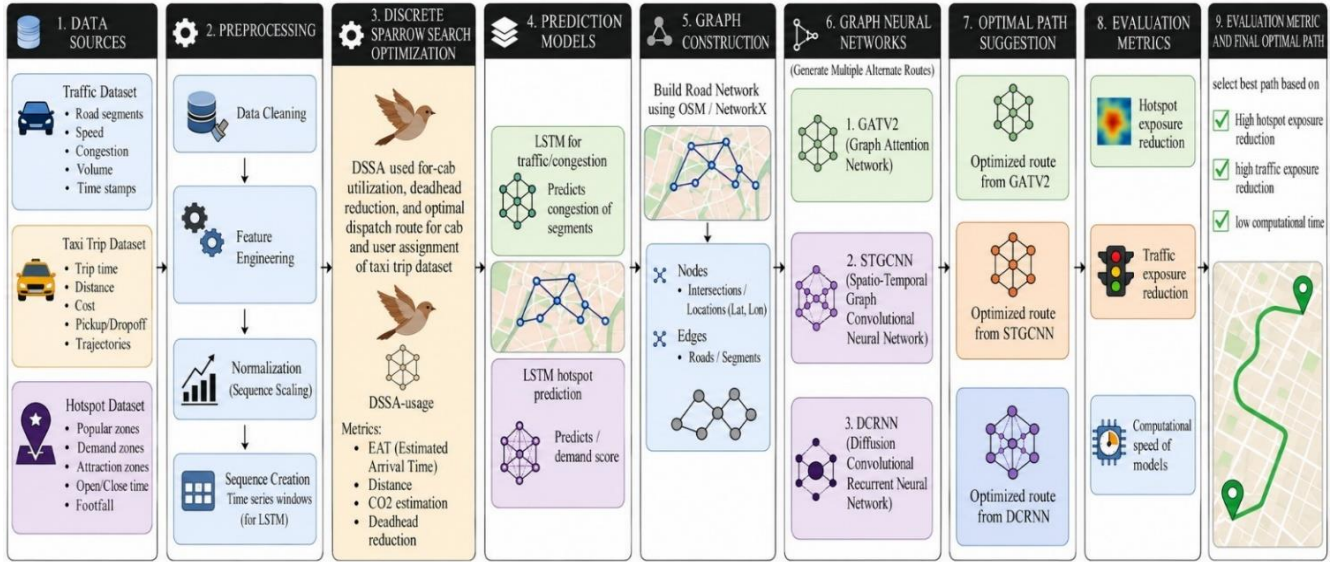


Fig. 2 Process flow diagram

4. Results and Discussions

To identify peak service usage, this study trained and tested its framework on data from Flash Cab, Chicago's largest taxi fleet management service, which operates 577 vehicles. Figure 3 illustrates the daily demand from July 25th to 31st, 2024. This specific week was chosen because the taxi trip and traffic datasets perfectly overlap during this period. The peak usage of cabs was on 25th July 2024, with 863 trips made by 280 flash cabs.

Figure 4 portrays the peak-hour usage of these cabs on that day. All ride requests within that peak hour are sorted chronologically based on taxi_id, and the trip_start_timestamp

features to simulate a real dispatch queue conditions. Fleet initialisation is performed using each taxi's Last Known Location (LKL), which is updated sequentially using past trip end points; if unavailable, the pickup coordinate of its first July trip is used, ensuring state continuity.

Each taxi is registered with latitude, longitude, and a cumulative load counter, enabling closed-loop state updates. Around 78 trips were made between 11 a.m. and 12 p.m. The taxis used to provide services were only 57, as shown in Figure 5. A baseline deadhead metric is computed using mean-centred spatial dispersion in meters to estimate pre-optimisation inefficiency for later comparison.

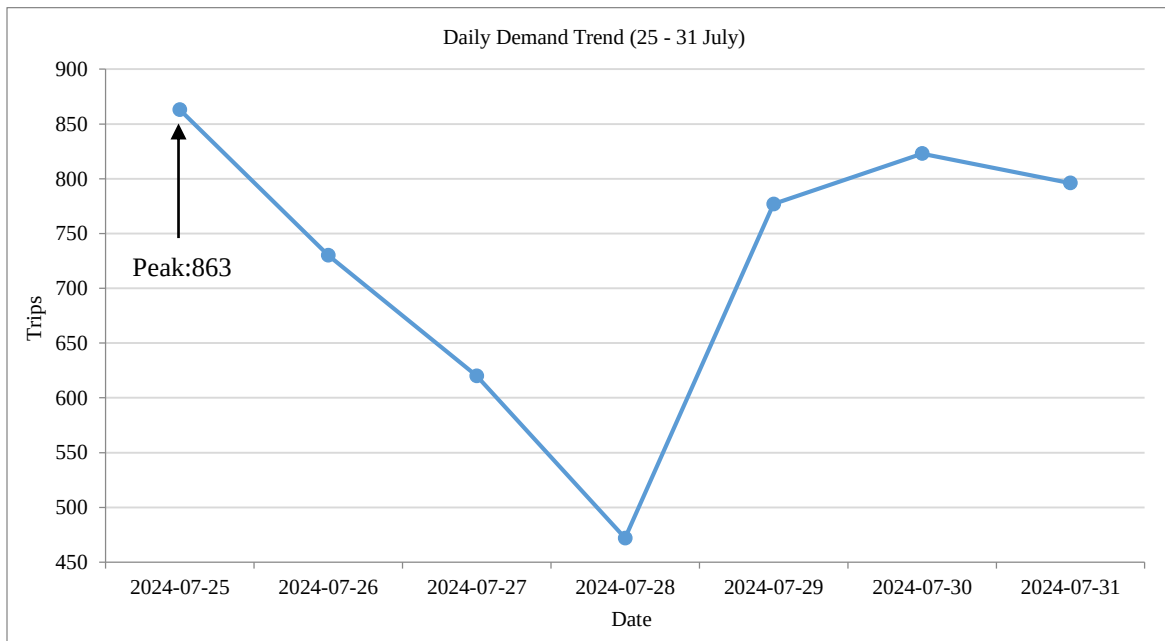


Fig. 3 Daily demand trend for 25 to 31 July, 2024

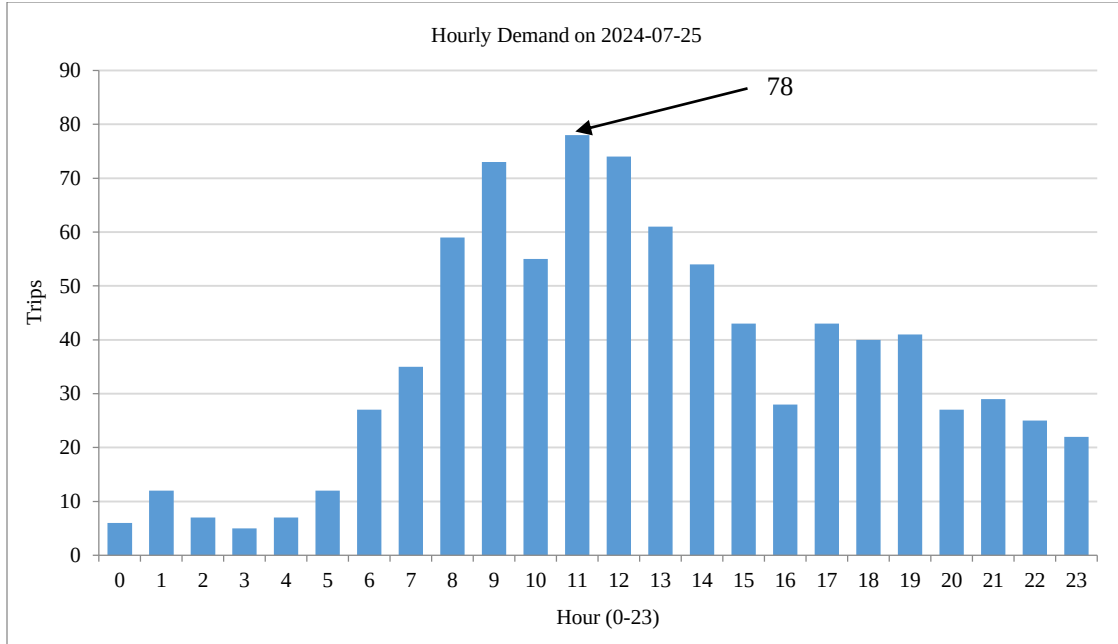


Fig. 4 Peak hour identification

```

TARGET DATE: 2024-07-25
one of Peak traffic Hour: 11:00
Total Trips on this day: 863
Trips at this hour: 78
Active Flash Cabs at peak hour: 57
Total Fleet Available on 2024-07-25: 280
    
```

Fig. 5 Peak hour cab usage

The DSSA metaheuristic module directly operates on the Chicago Taxi trips dataset for training and testing purposes. It evaluates the candidate solutions across population search matrices over 22,346 valid July Taxi trips.

The training runs for 200 epochs with an early stopping of 20 epochs, yielding the same global best fitness score. Its performance in minimising the 3 objectives was shown in Figure 6.

DSSA OPTIMIZATION REPORT				
Trips Optimized	:	78		
Total Fleet Available	:	165 cabs		
cabs Used Before	:	57		
Cabs Used After	:	66		
Deadhead Reduction	:	11.43% (↓ 3700.69 meters 2.30 miles)		
Original Deadhead	:	32373.59 m (20.12 miles)		
Optimized Deadhead	:	28672.90 m (17.82 miles)		
Total Runtime	:	0.76s		
◆ UPDATED DSSA STABILIZATION TABLE				
Run	Cabs_Used	Deadhead_Reduction	Total_Trips	Algorithm_Runtime
1	66	11.431205	78	0.760022

Fig. 6 DSSA optimisation report

An increase in utilisation of cabs from 57 to 66 shows that such cabs were either idle or cruising for pickups in the wrong region, have been utilised, and the trip requests were evenly

distributed; thereby, the income is balanced and reduces the over-usage of the same cab. It further extends that the cruising time, i.e., the “Deadhead” distance travelled, is also reduced.

To assign cabs for each request made, the overall execution time of the DSSA is also recorded and portrayed in Figure 6. Since the requests are assigned seamlessly, the waiting time for each passenger is also reduced. This module, therefore, integrates discrete swarm intelligence, adaptive fleet chaining, chaos-based entropy injection, and cryptographic security into a unified DSSA optimisation layer

suitable for real-time intelligent mobility systems. As a side effect, an average of 11.4% reduction in CO₂ is achieved by this model. The time taken to pick up the passengers is also reduced to a total of 3.70 km for the spatiotemporal parameters taken. The results are discussed based on the comparison of this model with the baseline model i.e., the default method, and is displayed in Figure 7.

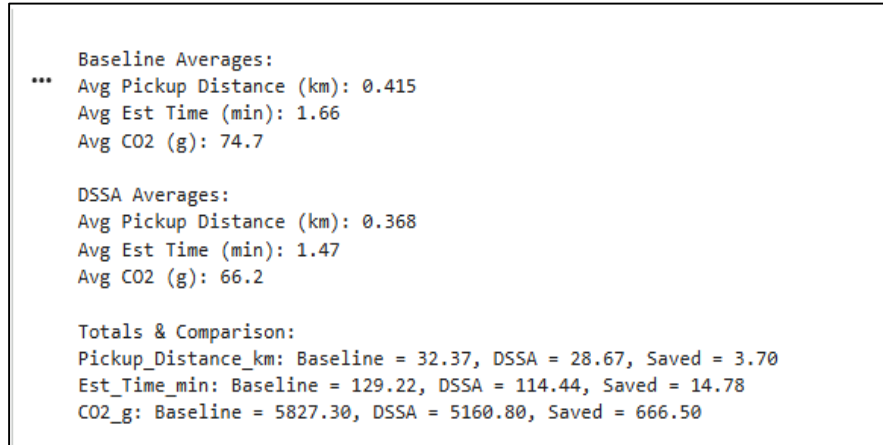


Fig. 7 DSSA versus baseline

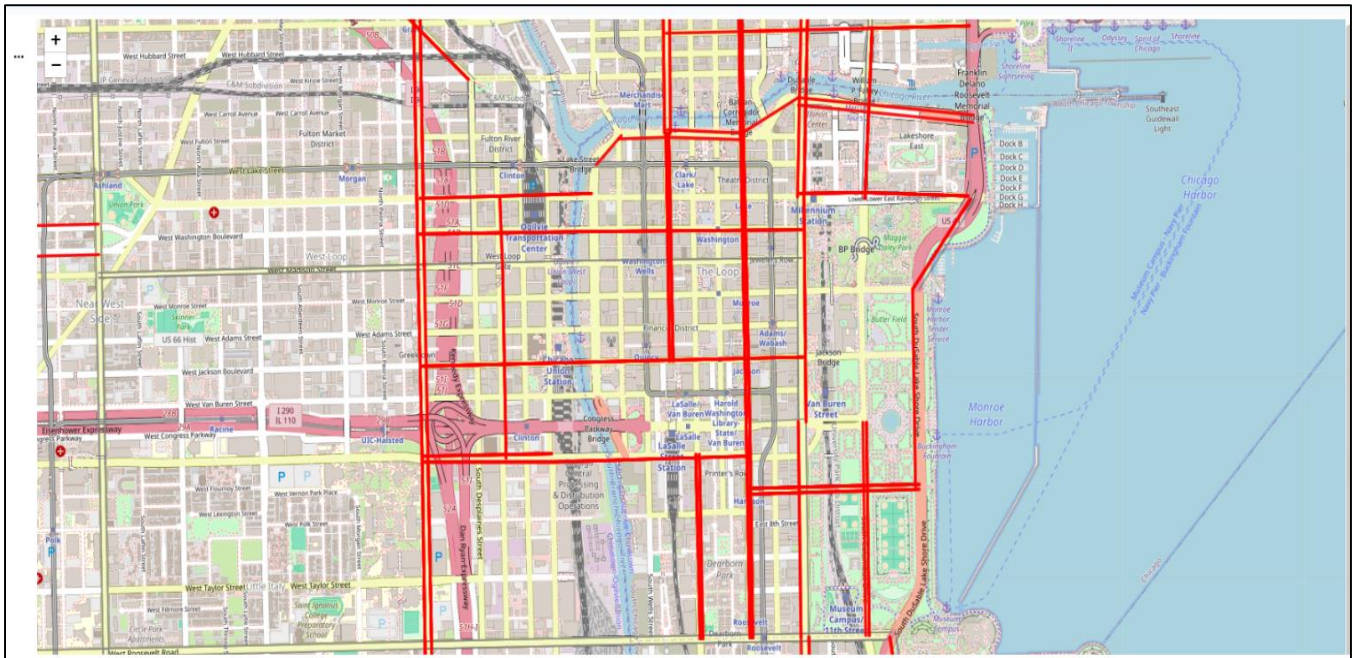


Fig. 8 Chicago congested road network

While the DSSA optimised the mapping of riders to drivers, the next phase contributed its part by forecasting the hotspots and the traffic zones to be crossed between each trip source and destination. It is common to have congestion on roads during peak hours, but it can be avoided if forecasted early. Hence, with the help of an LSTM that was trained with the Chicago Traffic Dataset, the prediction operation was performed, and Figure 8 depicts the forecasted traffic

segments in the city for the testing date and time. Congestion on the road occurs when more people share the road segment, and hotspots will attract a hectic crowd.

To forecast the Hotspots that create congestion, another LSTM is trained, and around 146 Hotspots were predicted; a few of them have been marked for easy understanding in Figure 9 during the peak hour on 25th, July 2024.

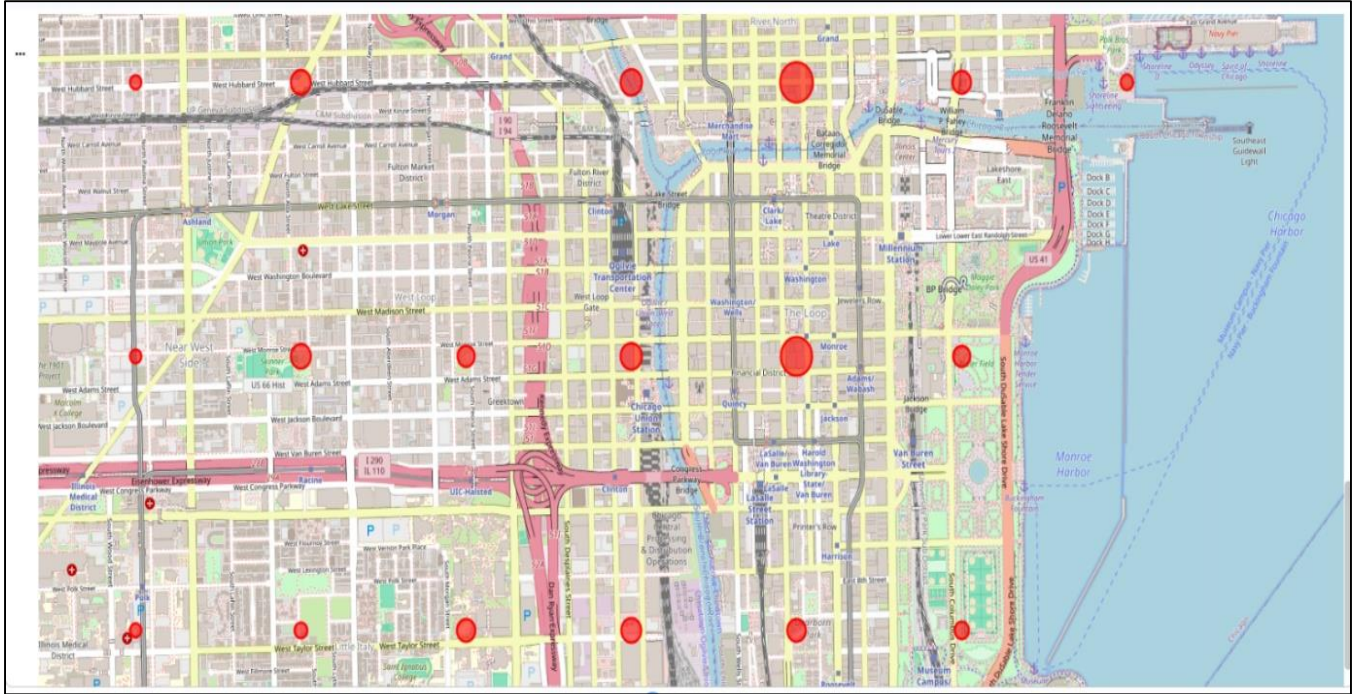


Fig. 9 Predicted hotspots

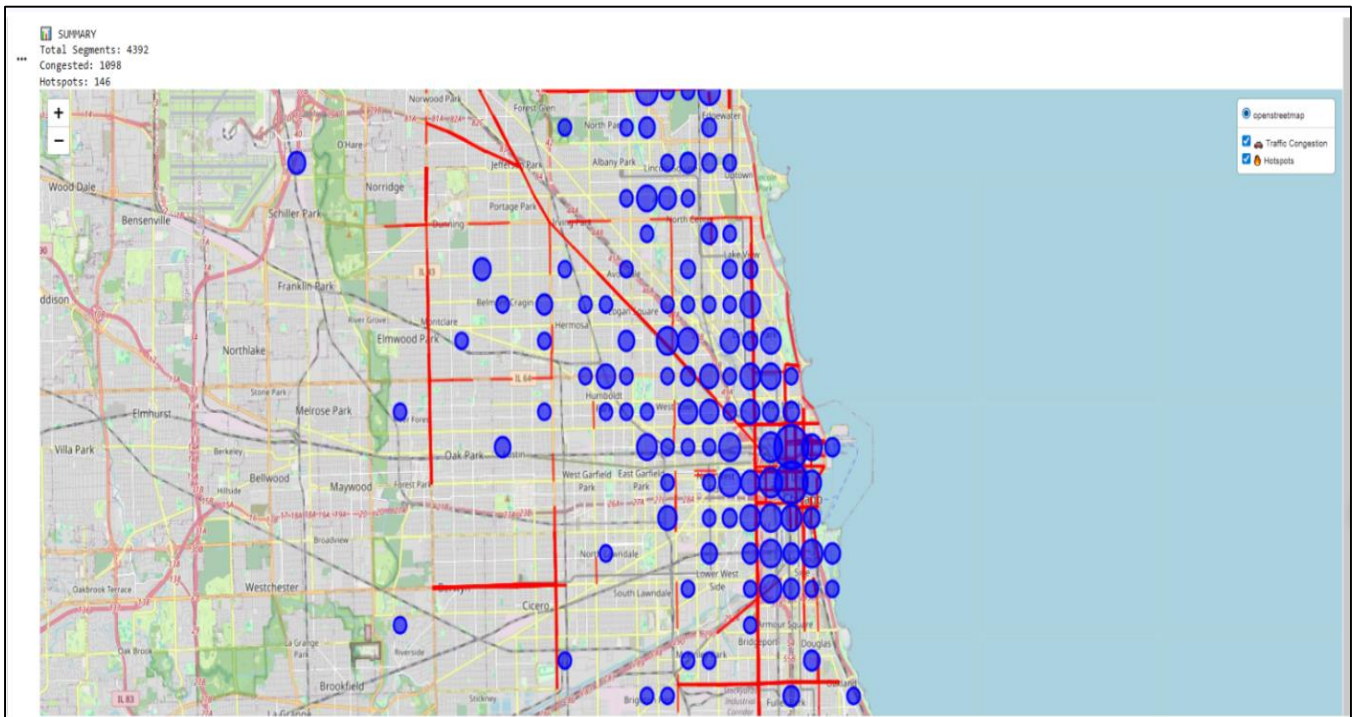


Fig. 10 Traffic segments and hotspots

The combined Traffic segments and the hotspots together were marked in Figure 10. The total road segments displayed are 4392, out of which 1098 segments were marked as congested lanes, and 146 hotspots were available, which attracted the crowd. Since the DSSA optimised the reassignment of ride requests from 57 to 66 cabs, most of the

cabs that were close to pick-up locations were connected, and a few cabs that travelled to reach the customer location were marked in Figure 11, with deadhead distance to be crossed to pick-up marked in purple coloured dashed lines. Graph neural networks such as GATv2, STGCNN, and the DCRNN have been trained with the Chicago Taxi Trip dataset and the

Chicago Traffic dataset, and these models were in a position to suggest alternate paths when traffic segments and hotspots were given as inputs. The estimated kilometres to be travelled and the time taken to compute the path for all 78 trips were recorded.

Figure 12 displays the number of hotspots and the congested zones to be crossed by the GATv2, which were reduced when compared to the baseline path to be travelled. A snapshot of the GATv2 alternate path is also shown in Figure 13.

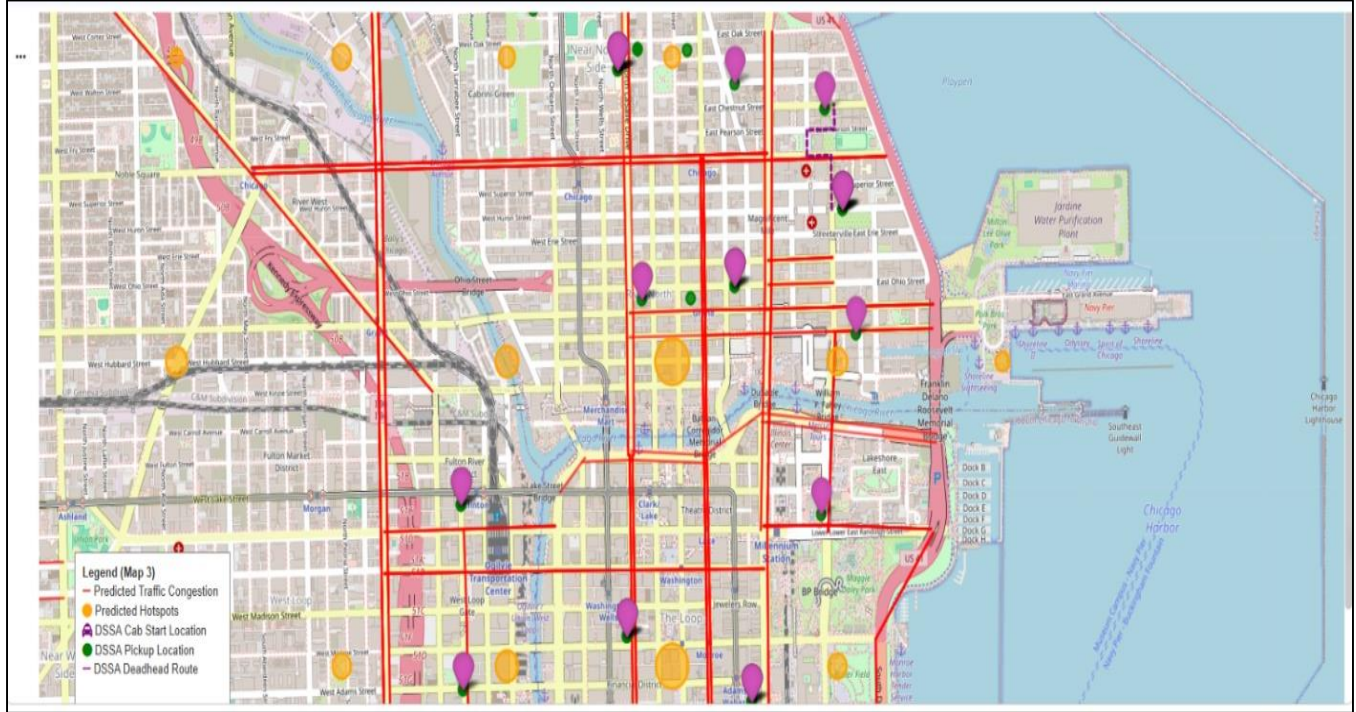


Fig. 11 A snapshot of the DSSA cabs to reach the corresponding customer

DSSA_Taxi_ID	User_Pickup_Lat	User_Pickup_Lon	User_Dropoff_Lat	User_Dropoff_Lon	Baseline Dist (km)	GATv2 Dist (km)	Distance_km	Baseline Time (min)	GATv2 Time (min)	Time_min	Hotspots Baseline	Hotspots GATv2	Hotspot Change (GATv2 - Baseline)	Traffic Baseline	Traffic GATv2	Traffic Change (GATv2 - Baseline)	GATv2 Time (s)
12829774801528933cdce279a9ebc9dbab805ccd5e5b52...	41.878866	-87.625192	41.986712	-87.663416	13.14	13.43	-0.29	52.43	53.59	-1.16	4	0	-4	213	172	-41	6.099
afbc704ce13c65a59e682bde0c6abff92e67c79a034fe...	41.986712	-87.663416	41.878866	-87.625192	13.17	13.20	-0.03	52.56	52.70	-0.14	1	0	-1	164	164	0	2.187
ad3406bedb910f12bb5e1f1f7236e83379fd7b58c64...	41.877406	-87.621972	41.890922	-87.618068	1.55	1.55	0.00	6.19	6.19	0.00	2	2	0	85	85	0	6.522
07009b8e9b32f36656567f7809b970af38e59dbb58704a...	41.812949	-87.617860	41.761578	-87.572782	8.83	8.83	0.00	35.24	35.24	0.00	0	0	0	22	22	0	1.632
a394089d397e0007d9e4154fa76936aba7bd6052e892e...	41.900266	-87.632109	41.877406	-87.621972	3.42	3.46	-0.04	13.65	13.79	-0.14	3	1	-2	137	134	-3	1.386
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
f19677875bae72431e9e35abbf3fd29aed79e11f2104...	41.885300	-87.642808	41.891972	-87.612945	3.05	3.05	0.00	12.18	12.18	0.00	3	3	0	104	104	0	1.050
ad3406bedb910f12bb5e1f1f7236e83379fd7b58c64...	41.895033	-87.619711	41.877406	-87.621972	1.86	1.86	0.00	7.44	7.44	0.00	2	2	0	103	103	0	2.259
f7bc9f63826f85b491e1e945a6c7ae000f9827ec23f...	41.899156	-87.626211	41.879255	-87.642649	3.59	3.61	-0.02	14.33	14.42	-0.09	2	3	1	104	100	-4	0.734
35a442d7913c9d9dbcd75d97179667ee556898b8d8b55...	41.986712	-87.663416	41.947792	-87.683835	6.17	6.18	-0.01	24.64	24.69	-0.05	2	1	-1	19	18	-1	2.247
90738dbca5b9c7c2de984b7e96a81569178364ca8cd29...	41.965812	-87.655879	41.899602	-87.633308	7.97	7.98	-0.01	31.83	31.84	-0.01	2	1	-1	83	72	-11	0.790

Fig. 12 Baseline travel distance versus GATv2 travel distance (in Kms) and other parameters

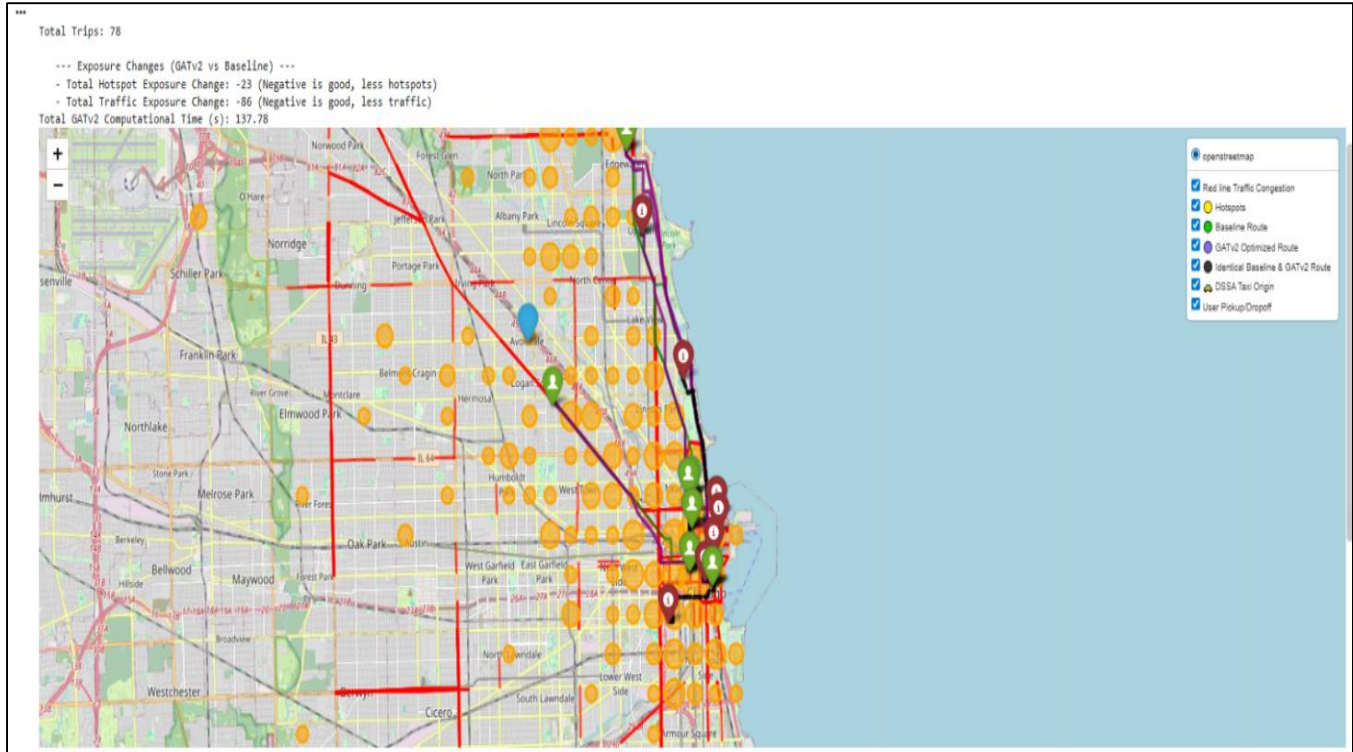


Fig. 13 GATv2 computational time to generate alternate paths

...

* GATv2 REDUCED EXPOSURE TRIPS:

	Trip_ID	Hotspot Change (GATv2 - Baseline)	Traffic Change (GATv2 - Baseline)	Baseline Dist (km)	GATv2 Dist (km)	Distance_km	Baseline Time (min)	GATv2 Time (min)	Time_min
0	8ff4cd37db22a6c8c12d932aac8ac54ad3bb0	-4	-41	13.14	13.43	-0.29	52.43	53.59	-1.16
1	79cd4fe9531233c3b9e82ace31f1653c2c4d8997	-1	0	13.17	13.20	-0.03	52.56	52.70	-0.14
4	e9fcd596796758013a82c37542d8bae4b55d48c	-2	-3	3.42	3.46	-0.04	13.65	13.79	-0.14
7	39f617427e4d269635e4e4dcb5448a51db6edbd	-1	-14	8.69	8.73	-0.04	34.68	34.84	-0.16
8	f3cc858740879a54065942025e06cfb7aebc086	-1	-11	7.97	7.97	0.00	31.82	31.83	-0.01
14	05a7adb71cf5afb2f9a60937a6716128401636db	0	-16	1.28	1.28	0.00	5.12	5.13	-0.01
18	af85275805680cd53c9b85de120ea6a0d083db	-1	-11	10.40	10.41	-0.01	41.52	41.53	-0.01
20	e0dd2cd661191e420891f7756d4977c75f6f14e	0	-4	1.34	1.35	-0.01	5.36	5.40	-0.04
22	7257f6c0a6704e91aeb0242dbd0fa998f3aa74	-1	-12	13.98	14.05	-0.07	55.79	56.09	-0.30
24	981bca1625929b32c59618384aa0b97efd97ab30	-1	-7	3.27	3.28	-0.01	13.07	13.09	-0.02
25	0c9dc130369d5af0b38c82800b9e7f7f7cbc9738	0	-11	3.40	3.42	-0.02	13.58	13.66	-0.08
27	3d23b436e6d1e125f13a196c1c196a0c428c5b7	0	-16	4.27	4.39	-0.12	17.04	17.54	-0.50
28	88ea2262ea312e6bd7f66371d35ebdf0d3f2160	-1	0	2.80	2.85	-0.05	11.16	11.36	-0.20
30	63f08e9b4b38f97f451aacbea855b30ce9f542	0	-1	15.16	15.16	0.00	60.51	60.52	-0.01
34	886f0fd911da85bec4b1824402d87b06785afe	-1	-13	2.47	2.47	0.00	9.86	9.88	-0.02
41	088402e96b697269e7684721454262ac196502f	-1	14	1.79	1.87	-0.08	7.15	7.46	-0.31
43	5191d3d90b5ed16584d6e2cb9962a50e757c7104	-1	-20	10.87	11.15	-0.28	43.37	44.49	-1.12
53	c53579fc3f36bfb9c3fd3fee948c85fe4d7bd	-3	4	8.50	8.53	-0.03	33.92	34.03	-0.11
55	64b9ac1bae45104310b1867c1803982535ad3417	-2	6	6.28	6.44	-0.16	25.08	25.69	-0.61
59	0e131afe9d25fd2e59c5ddc351d5bd2074ecf02d	-1	0	16.07	16.07	0.00	64.15	64.16	-0.01
60	cb0abafcb8e7b1aac524b624edc59a7c1b46f0b7	-1	6	3.77	3.87	-0.10	15.06	15.44	-0.38
63	3ac4e8dfc79ad28e7e7dfc7f64f19084f30193	-1	-7	3.27	3.28	-0.01	13.07	13.09	-0.02
71	03b90fc629157e9b65d4a51f6e0f03a4df61bd	-1	1	2.84	2.88	-0.04	11.32	11.51	-0.19
75	021001b17848c9c8c9c2a11641cb7a55f0bc91d7	1	-4	3.59	3.61	-0.02	14.33	14.42	-0.09
76	5b025fca886e0522b18c736d999b179099d7dd3	-1	-1	6.17	6.18	-0.01	24.64	24.69	-0.05
77	5884a968648dbed6e20ea9391f6ebcb722ba	-1	-11	7.97	7.98	-0.01	31.83	31.84	-0.01

--- Summary for Reduced Exposure Trips ---

Total Trips with Reduced Exposure: 26

- Total Hotspot Exposure Change: -26

- Total Traffic Exposure Change: -172

Fig. 14 GATv2 reduced the hotspots and the traffic-congested zones for 26 trips

The overall performance of the GATv2 model is recorded for the 26 trips for which it has generated the path; the hotspots

to bypass, and the traffic sector it avoided are shown in Figure 14. The Computational time of GATv2 is 137.78 seconds,

whereas the Spatio-Temporal Graph Convolutional Neural Networks (STGCNN) performed well in generating routes for the 78 trips in 129.40 seconds, which is slightly faster, but avoided less traffic and hotspots when compared to GATv2.

Figure 15 and Figure 16 portray the computational time taken by the model and a snapshot of the routes generated, and Figure 17 shows exactly how the 23 trips effectively reduced the traffic and hotspots, respectively.

DSSA_Taxi_ID	User_Pickup_Lat	User_Pickup_Lon	User_Dropoff_Lat	User_Dropoff_Lon	Baseline Dist (km)	STGCNN Dist (km)	Distance_km	Baseline Time (min)	STGCNN Time (min)	Time_min	Hotspots Baseline	Hotspots STGCNN	Hotspot Change (STGCNN - Baseline)	Traffic Baseline	Traffic STGCNN	Traffic Change (STGCNN - Baseline)	STGCNN Time (s)
29774801528933cdca279a9ebc9dbab805ccd5e5b52...	41.878866	-87.625192	41.986712	-87.663416	13.14	13.14	0.00	52.43	52.46	-0.03	4	3	-1	213	201	-12	1.521
bc704ce13c65a59e682bde0c6abfd92e67c79a034fa...	41.906712	-87.663416	41.878866	-87.625192	13.17	13.21	-0.04	52.56	52.72	-0.16	1	0	-1	164	164	0	2.347
d3406bedb9f10f12fbb5e1f1f7236e883379fd7b58c64...	41.877406	-87.621972	41.890922	-87.618868	1.55	1.58	-0.03	6.19	6.29	-0.10	2	2	0	85	85	0	0.751
109b8e9b32c366565677809b970af30e59dbb58704a...	41.812949	-87.617860	41.761578	-87.572782	8.83	8.83	0.00	35.24	35.24	0.00	0	0	0	22	22	0	2.326
4089d397e0007d9a4154fa7f6936aba7bd6052e892a...	41.900266	-87.632109	41.877406	-87.621972	3.42	3.44	-0.02	13.65	13.73	-0.08	3	2	-1	137	157	20	0.828
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19677875bae72f431e9e35abb3fd29aedd79e11f2f04...	41.885300	-87.642808	41.891972	-87.612945	3.05	3.06	-0.01	12.18	12.21	-0.03	3	3	0	104	117	13	0.755
d3406bedb9f10f12fbb5e1f1f7236e883379fd7b58c64...	41.895033	-87.619711	41.877406	-87.621972	1.86	1.86	0.00	7.44	7.44	0.00	2	2	0	103	103	0	0.686
f7bcb9f63026f85b491e1e945a6c7ae0009b827ec23f...	41.899156	-87.626211	41.879255	-87.642649	3.59	3.61	-0.02	14.33	14.42	-0.09	2	2	0	104	106	2	2.501
1442c7913c9ddfbcd75d97179667ee556898e8d8b55...	41.906712	-87.663416	41.947792	-87.683835	6.17	6.18	-0.01	24.64	24.66	-0.02	2	2	0	19	15	-4	0.769
38dbca5b9c7c2de984bf7e96a81569178364ca8cd29...	41.965812	-87.655879	41.899602	-87.633308	7.97	7.98	-0.01	31.83	31.84	-0.01	2	1	-1	83	72	-11	2.558

Fig. 15 STGCNN performance

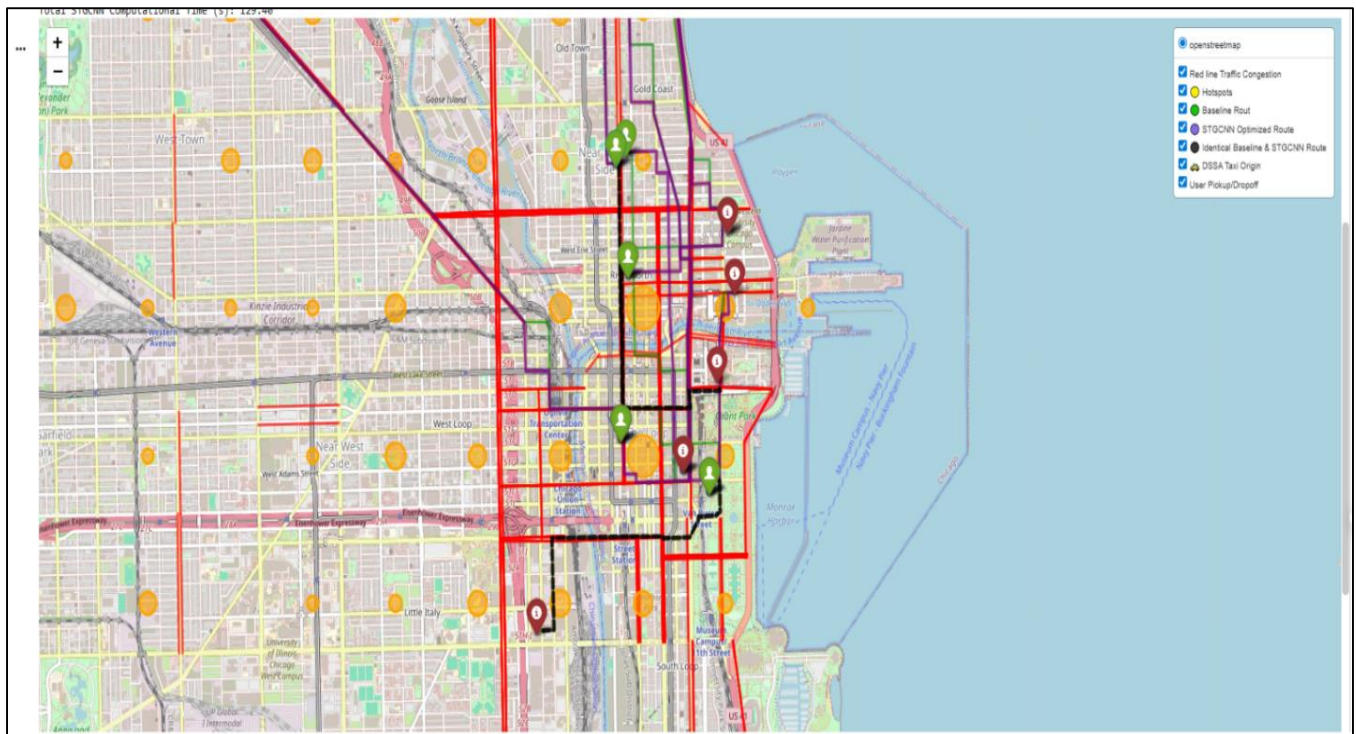


Fig. 16 A few trips of STGCNN-generated routes

...

• STGCNN REDUCED EXPOSURE TRIPS:

	Trip_ID	Hotspot Change (STGCNN - Baseline)	Traffic Change (STGCNN - Baseline)	Baseline Dist (km)	STGCNN Dist (km)	Distance_km	Baseline Time (min)	STGCNN Time (min)	Time_min	
0	8ff48cd37db22a6c8c12d932aac8ac54a5d3cbb0		-1	-12	13.14	13.14	0.00	52.43	52.46	-0.03
1	79cd4fe9531233c3b9e82ace31f1653c2c448997		-1	0	13.17	13.21	-0.04	52.56	52.72	-0.16
4	e9fcf5967f9675f013a82c375429d8ae4b55d48c		-1	20	3.42	3.44	-0.02	13.65	13.73	-0.08
7	39f617427e64269635e4e4dcb5d4a51db6edbd		0	-12	8.69	8.71	-0.02	34.68	34.77	-0.09
8	f3cc858740879a54065942025e86cfb7aebc086		-1	-11	7.97	7.97	0.00	31.82	31.83	-0.01
14	05a7adb7fc5af2f9a68937a6716128401836db		0	-16	1.28	1.28	0.00	5.12	5.13	-0.01
18	a1852758050680cd53c9b85de120ea6a00d083db		-1	-11	10.40	10.41	-0.01	41.52	41.53	-0.01
22	7257f6c0a6704e91aeb0242dbd0fa998ef3aa74		-1	12	13.98	14.07	-0.09	55.79	56.16	-0.37
24	981bca1625929b32c69618384aa0b97efd97ab30		-1	13	3.27	3.29	-0.02	13.07	13.12	-0.05
25	0c9dc130369d5a#0b38c82800b9e7f7f7cbc9738		0	-11	3.40	3.42	-0.02	13.58	13.66	-0.08
27	3d23b436e6d1e125ff13a196c1c196a0c428c5b7		1	-3	4.27	4.37	-0.10	17.04	17.43	-0.39
28	88ea2262ea312efb7fc66371d35e8dfd03f2160		-2	-2	2.80	2.89	-0.09	11.16	11.52	-0.36
34	886f0fd911da85bec4b18244002d7fb06785afe		-1	-13	2.47	2.47	0.00	9.86	9.88	-0.02
41	088402e6b697269e7684721454262cac196502f		-1	14	1.79	1.87	-0.08	7.15	7.46	-0.31
53	c53579fc3f36bfb9c3fd3fe948c85fefed47bd		-3	25	8.50	8.54	-0.04	33.92	34.09	-0.17
55	64b9ac1bae45104310b1867c188398253ad3417		-2	3	6.28	6.44	-0.16	25.08	25.69	-0.61
56	9b246e48b16a1da7b736a6368cbca48cfd70ad		0	-45	12.09	12.31	-0.22	48.24	49.15	-0.91
59	0e131afe9d25d2e59c5ddc351d5bd2074ec02d		0	-3	16.07	16.07	0.00	64.15	64.17	-0.02
60	cb0abafcb87b1aac524b624edc59a7c1b46f0b7		-1	3	3.77	3.87	-0.10	15.06	15.44	-0.38
63	3ac4e8dc79ad78e7efdc7f646f19084f30193		-1	13	3.27	3.29	-0.02	13.07	13.12	-0.05
71	03b90fc629157e9b65d4a51fca0cf03ad6f1bd		-1	-12	2.84	2.88	-0.04	11.32	11.50	-0.18
76	5b025fca886fe0522b18c736d99eb179099d7d3		0	-4	6.17	6.18	-0.01	24.64	24.66	-0.02
77	5804a960640d0be6e20ea9f391fbc8f8e22b4a		-1	-11	7.97	7.98	-0.01	31.83	31.84	-0.01

--- Summary for Reduced Exposure Trips ---

Total Trips with Reduced Exposure: 23

- Total Hotspot Exposure Change: -19

- Total Traffic Exposure Change: -63

Fig. 17 STGCNN reduced the exposure to traffic and hotspots

DSSA_Taxi_ID	User_Pickup_Lat	User_Pickup_Lon	User_Dropoff_Lat	User_Dropoff_Lon	Baseline Dist (km)	DCRNN Dist (km)	Distance_km	Baseline Time (min)	DCRNN Time (min)	Time_min	Hotspots Baseline	Hotspots DCRNN	Hotspot Change (DCRNN - Baseline)	Traffic Baseline	Traffic DCRNN	Traffic Change (DCRNN - Baseline)	DCRNN Time (s)
12829774801528933cdce279a9ebc9dbab805cc5e5b52...	41.878866	-87.625192	41.906712	-87.663416	13.14	13.14	0.00	52.43	52.44	-0.01	4	3	-1	213	202	-11	0.975
afbc704ce13c65a59e682bde0c5abff092e67c79a034fe...	41.986712	-87.663416	41.878866	-87.625192	13.17	13.17	0.00	52.56	52.56	0.00	1	1	0	164	164	0	1.343
ad3406bed9f10f12b0b5e11f17236e883379fd7b58c64...	41.877406	-87.621972	41.890922	-87.618868	1.55	1.55	0.00	6.19	6.19	0.00	2	2	0	85	85	0	3.014
07009b8e9b32f366565677809b970af30e59dbb58704a...	41.812949	-87.617860	41.761578	-87.572782	8.83	8.83	0.00	35.24	35.24	0.00	0	0	0	22	22	0	0.845
e394089d397e0007d9a4154fa76936aba7bd6052e892e...	41.900266	-87.632109	41.877406	-87.621972	3.42	3.44	-0.02	13.65	13.73	-0.08	3	2	-1	137	146	9	0.767
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
f19677875bae724f31e9a35abbf3fd29aedd79e11f2f04...	41.885300	-87.642808	41.891972	-87.612945	3.05	3.05	0.00	12.18	12.18	0.00	3	3	0	104	104	0	0.771
ad3406bed9f10f12b0b5e11f17236e883379fd7b58c64...	41.895033	-87.619711	41.877406	-87.621972	1.86	1.86	0.00	7.44	7.44	0.00	2	2	0	103	103	0	2.514
f7bcb9ff6326f0b5d491e1e945a6c7ae00f09827ec23f...	41.899156	-87.626211	41.879255	-87.642649	3.59	3.60	-0.01	14.33	14.38	-0.05	2	2	0	104	117	13	0.776
35a442cf7913c9d0fbc075d97179667e556898e8d8b55...	41.986712	-87.663416	41.947792	-87.683835	6.17	6.18	-0.01	24.64	24.67	-0.03	2	1	-1	19	15	-4	0.919
90738dbca5b9c7c2de984b7e96a81569178364ca8cd29...	41.965812	-87.655879	41.899602	-87.633308	7.97	7.98	-0.01	31.83	31.84	-0.01	2	1	-1	83	72	-11	4.120

Fig. 18: DCRNN travel time and computational time for each trip

The last model is the Diffusion Convolutional Recurrent Neural Network (DCRNN), which has generated the expected routes for all 78 trips in 133.98 seconds, which is a moderate speed by avoiding a considerable number of hotspots, but

suggested routes where more traffic segments needed to be crossed. Figures 18 and 19 display the travel time and other parameters generated by the DCRNN and the output map containing routes for a few trips. Overall, this model has the

ability to suggest an optimal path for 18 trips, as shown in Figure 20. The overall performance of this model is displayed in Figure 21. It is observed that, though the training time for predicting traffic zones has taken a considerable amount of

time in the entire system, it has performed well in re-assigning cabs with passengers, forecasting the traffic and hotspot zones, and generating GATv2-generated paths for all the trips within a short span of time, leaving the other two GNNs.

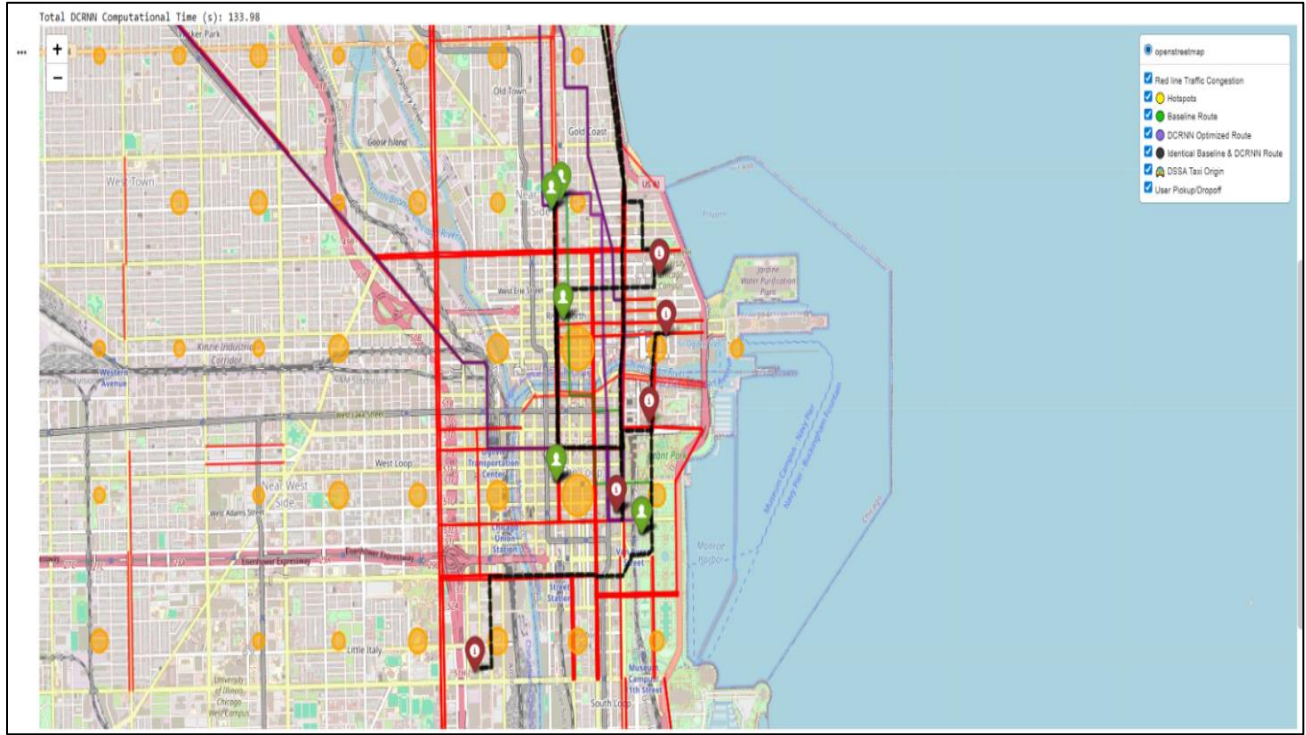


Fig. 19 DCRNN routes generation snapshot

...

• DCRNN REDUCED EXPOSURE TRIPS:

	Trip_ID	Hotspot Change (DCRNN - Baseline)	Traffic Change (DCRNN - Baseline)	Baseline Dist (km)	DCRNN Dist (km)	Distance_km	Baseline Time (min)	DCRNN Time (min)	Time_min
0	8ff48cd37db22a6c8c12d932aac8ac54a5d3cbb0	-1	-11	13.14	13.14	0.00	52.43	52.44	-0.01
4	e9fc5967f9675f013a02c375429d8ae4b55d40c	-1	9	3.42	3.44	-0.02	13.65	13.73	-0.08
8	f3cc858740879a54065942025e86c8fb7aebc086	-1	-11	7.97	7.97	0.00	31.82	31.83	-0.01
18	af85275805680cd53c9b85de120ea6a0d083db	-1	-11	10.40	10.41	-0.01	41.52	41.53	-0.01
22	7257f6c0a6704e91aeb0242ddbe0fa990ef3aa74	-1	13	13.98	13.99	-0.01	55.79	55.83	-0.04
24	981bca1625929b32c69618384aa0b97efd97ab30	-1	13	3.27	3.29	-0.02	13.07	13.11	-0.04
28	88ea2262ea312e6bd7f66371d35e8dfd03f2160	-2	1	2.80	2.87	-0.07	11.16	11.48	-0.32
30	63f08fe9bf4b38f97f451aacba855b30ce9f542	0	-2	15.16	15.16	0.00	60.51	60.52	-0.01
34	886f0fd911da85bec4b182440028d7fb06785afe	-1	-7	2.47	2.47	0.00	9.86	9.88	-0.02
41	088402e96b697269e7684721454262cac196502f	-1	14	1.79	1.87	-0.08	7.15	7.46	-0.31
53	c53579fc3f36bf9e9c3fd3fee948c85fefe47bd	-3	4	8.50	8.53	-0.03	33.92	34.06	-0.14
55	64b9ac1bae45104310b1867c1883982535ad3417	-1	0	6.28	6.29	-0.01	25.08	25.09	-0.01
59	0e131afe9d25fd2e59c5ddc351d5bd2074ecd02d	-1	0	16.07	16.07	0.00	64.15	64.16	-0.01
60	cb0abafcbef87b1aac524b624edc59a7c1b46f0b7	-1	0	3.77	3.78	-0.01	15.06	15.08	-0.02
63	3ac4e8dfc79ad28e7e7dfc7f646f19084f30193	-1	13	3.27	3.29	-0.02	13.07	13.11	-0.04
71	03b90fc629157e9b65d4a51fco0c03a4df61bd	-2	1	2.84	2.88	-0.04	11.32	11.49	-0.17
76	5b025fca886fe0522b18c736d99eb179099d7dd3	-1	-4	6.17	6.18	-0.01	24.64	24.67	-0.03
77	5884a968648d0bed6e20ea9391febcb0fe22b4a	-1	-11	7.97	7.98	-0.01	31.83	31.84	-0.01

--- Summary for Reduced Exposure Trips ---
Total Trips with Reduced Exposure: 18
- Total Hotspot Exposure Change: -21
- Total Traffic Exposure Change: +11

Fig. 20 Traffic and hotspot exposure changes by the DCRNN

...

=====	
COMPUTATIONAL TIME SUMMARY	
=====	
1. DSSA Dispatch Optimization Runtime:	0.0127 minutes
2. Traffic LSTM Training Time:	53.9333 minutes
3. Traffic LSTM Prediction Time:	1.3657 minutes
4. Hotspot LSTM Training Time:	0.4043 minutes
5. Hotspot LSTM Prediction Time:	0.0115 minutes
6. GATv2 Model Routing Runtime:	2.2963 minutes
7. STGCNN Model Routing Runtime:	2.1566 minutes
8. DCRNN Model Routing Runtime:	2.2330 minutes
=====	

Fig. 21 Summary of the entire framework

5. Conclusion

The constructed model DSSA-LSTM-GATv2 can effectively compute routes by avoiding traffic and hotspots, thereby resulting in an overall reduction in drivers' idle time and cruising time, which indirectly maximises the revenue across the cab drivers and reduces the waiting time and travel time of passengers for pickups and drop-offs and improves the

satisfaction of the trip at both ends. Its modularity helps to dynamically update the dataset of any component at ease without disturbing the others, rather than constructing a giant transformer for computation. This model is scalable for any FMS having 200+ cabs, due to its lower time consumption in forecasting the paths. This model is effective in reducing the emission of CO₂ during cruising time and congestion zones.

References

- [1] Shuofeng Wang et al., "Trajectory Analysis for On-Demand Services: A Survey Focusing on Spatial-Temporal Demand and Supply Patterns," *Transportation Research Part C: Emerging Technologies*, vol. 108, pp. 74-99, 2019. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [2] Alireza Nejadettehad, Hamid Mahini, and Behnam Bahrak, "Short-Term Demand Forecasting for Online Car-hailing Services using Recurrent Neural Networks," *Applied Artificial Intelligence*, vol. 34, no. 9, pp. 674-689, 2020. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [3] Jintao Ke et al., "Predicting Origin-Destination Ride-Sourcing Demand with a Spatio-Temporal Encoder-Decoder Residual Multi-Graph Convolutional Network," *Transportation Research Part C: Emerging Technologies*, vol. 122, pp. 1-17, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [4] Zhen Zhang, and Yang Han, "Discrete Sparrow Search Algorithm for Symmetric Traveling Salesman Problem," *Applied Soft Computing*, vol. 118, pp. 1-18, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [5] Boris Medina-Salgado et al., "Urban Traffic Flow Prediction Techniques: A Review," *Sustainable Computing: Informatics and Systems*, vol. 35, pp. 1-16, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [6] R. Yesodha, and T. Amudha, "A Bio-Inspired Approach: Firefly Algorithm for Multi-Depot Vehicle Routing Problem with Time Windows," *Computer Communications*, vol. 190, pp. 48-56, 2022. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [7] Hongguang Wu, and Yuelin Gao, "An Ant Colony Optimization based on Local Search for the Vehicle Routing Problem with Simultaneous Pickup-Delivery and Time Window," *Applied Soft Computing*, vol. 139, pp. 1-16, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [8] Youcef Djenouri et al., "Hybrid Graph Convolution Neural Network and Branch-and-Bound Optimization for Traffic Flow Forecasting," *Future Generation Computer Systems*, vol. 139, pp. 100-108, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [9] Jinwei Qiao et al., "A Modified Particle Swarm Optimization Algorithm for a Vehicle Scheduling Problem with Soft Time Windows," *Scientific Reports*, vol. 13, no. 1, pp. 1-18, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [10] Nicolás Frías, Franklin Johnson, and Carlos Valle, "Hybrid Algorithms for Energy Minimizing Vehicle Routing Problem: Integrating Clusterization and Ant Colony Optimization," *IEEE Access*, vol. 11, pp. 125800-125821, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [11] Xiaohui Huang et al., "Multi-View Dynamic Graph Convolution Neural Network for Traffic Flow Prediction," *Expert Systems with Applications*, vol. 222, pp. 1-14, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [12] Qichao Wu et al., "A Neighborhood Comprehensive Learning Particle Swarm Optimization for the Vehicle Routing Problem with Time Windows," *Swarm and Evolutionary Computation*, vol. 84, pp. 1-16, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]

- [13] Yufeng Wang et al., “Self-Competition Particle Swarm Optimization Algorithm for the Vehicle Routing Problem with Time Window,” *IEEE Access*, vol. 12, pp. 127470-127488, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [14] Ting Liu, and Yuan Liu, “GMTP: Enhanced Travel Time Prediction with Graph Attention Network and BERT Integration,” *AI*, vol. 5, no. 4, pp. 2926-2944, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [15] Dongwei Xu et al., “Hierarchical Spatio-Temporal Graph Convolutional Neural Networks for Traffic Data Imputation,” *Information Fusion*, vol. 106, pp. 1-11, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [16] Tong Liu, and Hadi Meidani, “End-to-End Heterogeneous Graph Neural Networks for Traffic Assignment,” *Transportation Research Part C: Emerging Technologies*, vol. 165, pp. 1-14, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [17] Yu Qian, “Optimization of Logistics Distribution Scheme based on Improved Artificial Bee Colony Algorithm,” *Procedia Computer Science*, vol. 262, pp. 1381-1387, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [18] Mingxuan Zhang, “Research on Vehicle Route Optimization for Half-Open Multi-Energy Urban Distribution Considering Order Priority,” *IEEE Access*, vol. 13, pp. 79007-79029, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [19] Agnieszka Polowczyk, Alicja Polowczyk, and Marcin Woźniak, “Graph Neural Network via Dynamic Weights and LSTM with Attention for Traffic Forecasting,” *Procedia Computer Science*, vol. 257, pp. 119-126, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [20] Hui Chen et al., “Urban Traffic Flow Prediction based on Multi-Spatio-Temporal Feature Fusion,” *Neurocomputing*, vol. 638, pp. 1-13, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [21] Guojun He et al., “Adaptive Spatial-Temporal Dependence Graph Convolution Neural Network for Traffic Flow Prediction,” *Expert Systems with Applications*, vol. 281, pp. 1-11, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [22] Tong Liu, and Hadi Meidani, “Graph Neural Networks for Travel Distance Estimation and Route Recommendation Under Probabilistic Hazards,” *International Journal of Transportation Science and Technology*, vol. 21, pp. 372-383, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [23] Linjie Zhang, and Jianfeng Ma, “A Spatiotemporal Graph Wavelet Neural Network for Traffic Flow Prediction,” *Journal of Information and Intelligence*, vol. 3, no. 2, pp. 173-188, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [24] Venkata Sai Sathvik Rajampalli, Akarsh Reddy Narra, and G.G. Jaffino, “A Spatio-Temporal GNN and Multi-Agent System Framework for Bottleneck Prediction and Signal Optimization in Bangalore,” *Results in Engineering*, vol. 28, pp. 1-15, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [25] Yuyan Annie Pan et al., “Urban Intersection Traffic Flow Prediction: A Physics-Guided Stepwise Framework Utilizing Spatio-Temporal Graph Neural Network Algorithms,” *Multimodal Transportation*, vol. 4, no. 2, pp. 1-20, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [26] Zhiyou Liu et al., “IWOA-RNN: An Improved Whale Optimization Algorithm with Recurrent Neural Networks for Traffic Flow Prediction,” *Alexandria Engineering Journal*, vol. 117, pp. 563-576, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [27] Minghao Tong, Zhenhua Peng, and Qin Wang, “A Hybrid Artificial Bee Colony Algorithm with High Robustness for the Multiple Traveling Salesman Problem with Multiple Depots,” *Expert Systems with Applications*, vol. 260, pp. 1-18, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [28] Shimon Komarovskiy, and Jack Haddad, “Spatio-Temporal Graph Convolutional Neural Network for Traffic Signal Prediction in Large-Scale Urban Networks,” *Transportation Research Interdisciplinary Perspectives*, vol. 32, pp. 1-11, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [29] Xuhui Lin et al., “Field-Theory Inspired Physics-Informed Graph Neural Network for Reliable Traffic Flow Prediction under Urban Flooding,” *Reliability Engineering and System Safety*, vol. 265, pp. 1-22, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [30] Taxi Trips, Chicago Data Portal, 2026. [Online]. Available: https://data.cityofchicago.org/Transportation/Taxi-Trips-2024-ajtu-insnz/about_data
- [31] Chicago Traffic Tracker - Historical Congestion Estimates by Segment - 2024-Current, Data.Gov, 2026. [Online]. Available: <https://catalog.data.gov/dataset/chicago-traffic-tracker-historical-congestion-estimates-by-segment-2024-current>