Prediction of travel time using deep learning algorithms to solve TDVRPTW

by

Zahra Ejabati Emanab

A Thesis submitted to the Faculty of Graduate Studies of

the University of Manitoba

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Management

Asper School of Business

University of Manitoba

Copyright © 2023 by Zahra Ejabati Emanab

Table of Contents

List of Tables

List of Figures

Abstract

Vehicle routing problem (VRP) is a well-known optimization problem in logistics and operations research. This problem aims to find the optimal routes to meet customers' demands in different locations with a specific number of vehicles. Time-dependent vehicle routing problem with time window (TDVRPTW) is a type of traditional VRP that incorporates the consideration of urban traffic conditions. Classic VRP plans routes without considering the dynamic nature of travel times, assuming that the roads are free-flowing, and travel times are constant. However, in real-world scenarios, many factors can affect the travel time like traffic congestion and varying travel speeds. To overcome this limitation, TDVRPTW introduces the concept of time-dependent travel times. In TDVRPTW, travel times can vary based on the time of day, traffic congestion, road conditions, and other factors. Moreover, TDVRPTW considers time windows, which are specific time intervals during which customers must be serviced.

The accuracy of travel time estimation plays a crucial role in the performance of TDVRPTW solutions. There are different ways to predict the travel time between two locations, including model-based and data-driven methods. Data-driven methods are classified into three groups: statistical methods, basic machine learning methods, and deep learning methods. In this project, we apply deep learning algorithms to predict travel time accurately. By leveraging the power of Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Temporal Convolutional Networks (TCN) architectures, our proposed algorithm demonstrated superior performance compared to existing methods. We achieved a significant improvement of 2.9% in travel time prediction accuracy.

In addition to utilizing deep learning algorithms to predict travel time, this thesis explores the use of genetic algorithms, local search, and large neighborhood search to solve TDVRPTW. Genetic algorithms are inspired by natural selection and iteratively evolve a population of potential solutions using genetic operators, to enhance the quality of the solutions over generations. To effectively manage the intensification (exploitation) and diversification (exploration) aspects in metaheuristics, local search, and large neighborhood search techniques are utilized.

In conclusion, this thesis aims to address the challenges posed by realistic and dynamic transportation systems by combining deep learning algorithms for travel time prediction and metaheuristic algorithms, in the context of the time-dependent vehicle routing problem with time windows (TDVRPTW). By integrating these techniques, we propose a comprehensive and effective approach to optimize route planning and meet customer demands efficiently.

Acknowledgment

I would like to express my heartfelt thanks to my advisors, Dr. Gajpal and Dr. Appadoo. Their guidance and support throughout this research project have been invaluable. I am grateful for their expertise and mentorship, which have played a crucial role in shaping this thesis. Their feedback and availability for discussions have been greatly appreciated.

I am also grateful to the Graduate Program Advisor and Coordinator, Ms. Claire Venevongsa, for their continuous support and responsiveness. Their guidance and assistance in addressing my concerns have been greatly appreciated.

To Dr. Gajpal and Dr. Appadoo, thank you for the opportunity to work under your guidance. Your support has been truly appreciated.

Sincerely,

Zahra Ejabati

Dedication

To my loving family and my devoted husband, who have been my pillars of strength throughout this journey.

To my parents, your belief in my abilities, countless sacrifices, and endless encouragement have been the foundation upon which I have built my academic success. Your unwavering love, guidance, and faith in me have given me the strength to overcome challenges and pursue my dreams. This achievement would not have been possible without your constant support, and I am forever grateful.

To my siblings, your constant support and inspiration have played a vital role in my success. Thank you for always pushing me to strive for excellence.

And to my dear husband, Alireza, your countless hours of assistance, meaningful discussions, and unwavering encouragement have been invaluable. Your patience, understanding, and support during this project have been the bedrock of my progress. Thank you for being my partner in both life and academia.

This thesis is a heartfelt tribute to my family, whose love, support, and guidance have propelled me to reach new heights.

With deepest gratitude,

Zahra Ejabat

Chapter 1

Introduction

In recent years, many researchers focus on applying data-driven models like machine learning (ML), deep learning (DL) algorithms to solve real-world problems due to the increasing availability of big data and processing powers. One of the main areas enhanced by the field of AI is urban transportation systems (Yao et al., 2019). In intelligent transportation systems, the travel time estimation of a given path plays a major role in navigation, route planning, traffic management, etc. The vehicle routing problem (VRP) and its variants are among the significant problems in transportation planning which have been widely studied and solved using diverse algorithms.

1.1 Vehicle Routing Problem

Vehicle Routing Problem is a problem that involves finding the optimal routes to visit customers located at different positions and deliver goods or provide services using a fleet of vehicles. Each customer has a specific demand to be served. All vehicles should start from the depot and after visiting some customers return to the depot. The problem can be formulated with either a single depot or multiple depots. In the single depot VRP, there is only one central location from which all vehicles begin and end their routes. On the other hand, in the multiple depot VRP, the problem involves having more than one depot, with each depot serving as the starting and ending point for a subset of vehicles. The objective of the VRP can vary based on the specific requirements of the problem. It can include minimizing the total cost, which can encompass factors such as vehicle operating costs, fuel costs, or labor costs. Other common objectives include minimizing the total

traveled distance, total time taken, or the number of vehicles used. The problem requires deciding which customers each vehicle should visit, in what sequence, and how the overall routes can be optimized to achieve the desired objective. The classic Vehicle Routing Problem (VRP) is commonly referred to as the Capacitated Vehicle Routing Problem (CVRP), where each fleet of vehicles has a predefined maximum capacity that must not be exceeded. A visual representation of VRP with 13 customers, one depot, and 3 vehicles are shown in Figure 1.

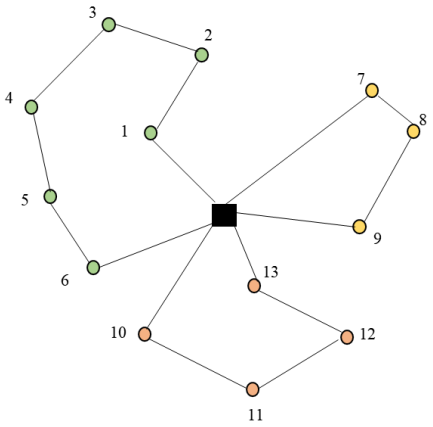

Figure 1: visual diagram of VRP

1.2 Vehicle Routing Problem with Time Windows.

VRP also can have time window constraints, which means every node should be visited between specific time ranges. This type of VRP is called vehicle routing problem with time window (VRPTW). In other words, each customer must be visited in desired time windows. These time windows represent the availability or preferences of customers for delivery or service. The objective of VRPTW is to find optimal routes for a fleet of vehicles that satisfy the time windows of all customers while minimizing total costs, distance traveled, or other relevant metrics. Incorporating time windows in VRP results in more complexity to the problem, since it requires

careful consideration of time-related factors to ensure that deliveries or services are performed within the specified time windows. In VRPTW, hard time windows refer to strict time constraints that must be adhered to. It means vehicles should arrive at customer locations within their specified time windows. Any violations from these time windows are not allowed and lead to infeasible solutions. In contrast, soft time windows consider some flexibility, where slight deviations may be permissible without significant penalties (Islam & Gajpal, 2021). The selection between hard and soft time windows depends on the specific requirements and operational context of the VRPTW problem.

1.3 Time-Dependent Vehicle Routing Problem

Time-dependent vehicle routing problem (TDVRP) is a type of VRP in which, traveling time between nodes may vary based on departure time (Donati et al., 2008). This variation is influenced by factors such as speed levels and traffic congestion along the route connecting the origin and destination. As a result, TDVRP necessitates the calculation of time estimates that consider these dynamic factors. While incorporating time-dependent factors enhances the accuracy and efficiency of routing plans in real-world transportation and logistics scenarios, it also introduces additional complexity to the problem. The consideration of dynamic factors such as traffic congestion and varying travel times in TDVRP increases the difficulty of finding optimal solutions. Addressing these challenges requires the development of advanced algorithms and techniques that can handle the time-dependent nature of the problem.

The traditional Time-Dependent Vehicle Routing Problem (TDVRP) is challenging due to the unpredictable fluctuations in travel times caused by factors such as speed and traffic conditions. Discrete and continuous speed functions are two distinct approaches used in time-dependent speed

estimation for the Vehicle Routing Problem. Discrete speed functions involve dividing time into predefined intervals and assuming a constant speed within each interval. This simplifies the calculations but may overlook the finer variations in speed. On the other hand, continuous speed functions consider speed as a continuous variable, allowing for more precise estimations by capturing the dynamic nature of speed changes over time. Trigonometric functions or other continuous functions are commonly employed to model these variations accurately.

To illustrate, consider a delivery vehicle traveling from Node A to Node B, a distance of 50 kilometers.

If we assume a constant speed of 40 km/h, without considering time-dependent factors, we might naively estimate the total travel time as follows:

$$Travel\ time\ =\ Distance\ /\ Speed\ =\ 50\ /\ 40\ =\ 1.25\ hours\ =\ 30\ min$$

Using a discrete speed function with two time intervals due to rushing hour, the vehicle maintains a speed of 60 km/h for the first 30 minutes and then reduces to 40 km/h for the next 1 hour. The estimated travel time would be:

$$First\ 30\ min\ traveled\ distance\ =\ Speed * Time\ =\ 60 * (30/60)\ =\ 30\ km\ <\ 50km$$

$$Time1\ =\ 30\ min$$

$$remaining\ distance\ =\ 50\ -30\ =\ 20$$

$$Time2\ =\ Distance/Speed\ =\ 20\ /40\ =\ 0.5\ hour\ =\ 30\ min\ <\ 1hour$$

$$Travel\ time\ =\ Time1 + Time2\ =\ 30 + 30\ =\ 60min\ =\ 1\ hours.$$

In contrast, using a continuous speed function, the vehicle's speed gradually decreases from 60 km/h to 40 km/h over the entire distance. The estimated travel time would be calculated using integrals or numerical methods, resulting in a more precise estimation based on the continuous variation of speed over time.

This example demonstrates the difficulty in the traditional TDVRP, as travel times are not constant and can significantly impact the efficiency and accuracy of routing plans. Accounting for time-dependent factors becomes crucial to accurately estimate travel times and optimize routes for real-world transportation scenarios.

When considering time-dependent factors in the Vehicle Routing Problem (VRP), the use of discrete or continuous speed levels to estimate travel times has limitations.

Using discrete speed levels, such as predefined speed categories (e.g., low, medium, high), may not capture the nuanced variations in travel times. Traffic conditions can change rapidly, and discrete speed levels may not accurately reflect the actual speed experienced at different times of the day or on specific road segments.

On the other hand, employing continuous speed levels, where the speed is treated as a continuous variable, offers more flexibility than discrete levels. However, estimating travel times accurately using continuous speed levels can be challenging due to the complex relationships involved. Factors such as time of day, traffic congestion, and road conditions contribute to the variation in travel times. Continuous speed models must account for these multiple variables, which can make the estimation process complex. Moreover, these models may struggle to generalize well to different scenarios or accurately capture non-linear variations in speed. These functions also require specific assumptions and parameter choices, making them sensitive to selection.

Additionally, one of the limitations of traditional approaches using discrete or continuous speed levels is the lack of consideration for spatial features of roads, such as traffic lights, turnings, and speed limits. These spatial features can have a significant impact on travel times and should be taken into account for accurate estimation.

1.4 Deep Learning Models.

Deep learning models are a type of artificial neural network that aim to learn and extract complex patterns and representations from data. The purpose of deep learning models is to solve tasks such as classification, regression, and prediction by automatically learning and adapting to the underlying patterns in the data. A simplified overview of the steps involved in training a basic neural network, which is the simplest deep leaning model are shown in Algorithm 1.

Algorithm 1: Basic Neural Network Algorithm

Input: Training data (input features and corresponding target outputs)
Output: Trained neural network model

1. Initialize the weights and biases of the neural network randomly or using specific initialization techniques.
2. Specify the architecture of the neural network, including the number of layers, number of neurons in each layer, and the activation functions to be used.
3. Set the learning rate, which determines the step size for updating the weights during training.
4. Set the number of iterations or epochs, representing the number of times the entire training dataset will be used for training.
5. For each iteration in the training process:
    a. Perform forward propagation: Compute the weighted sum of inputs and apply the activation function to calculate the output of each neuron in each layer.
    b. Calculate the loss or error between the predicted output and the true output.
    c. Perform backward propagation: Calculate the gradients of the loss with respect to the weights and biases using the chain rule.
    d. Update the weights and biases using the gradients and learning rate.
6. Repeat steps 5a-5d until convergence is achieved or the desired accuracy is reached.
7. The trained neural network model is ready for inference or making predictions on new, unseen data.

The provided pseudo-algorithm is specifically for a basic version of a neural network, which is a fundamental component of deep learning models. Deep learning models encompass various architectures, such as convolutional neural networks (CNNs) for image processing, recurrent neural networks (RNNs) for sequential data, and transformers for natural language processing. While the specific implementation details may vary across different deep learning architectures, the general concept of training through forward and backward propagation remains consistent.

1.5 Solution Methodology for VRP: Exact, heuristic, and metaheuristics

Solving the VRP typically involves employing different algorithms. When it comes to solving the VRP, there are three broad groups of algorithms: exact, heuristic, and metaheuristics. Exact algorithms explore all possible solutions, resulting in finding the optimal solution to the VRP. Examples of exact algorithms include branch and bound, dynamic programming, and integer programming. These algorithms are guaranteed to find the best solution but can be computationally expensive and impractical for large-scale instances of the VRP (Gajpal et al., 2019). Heuristic algorithms apply rules or strategies that prioritize certain aspects of the problem in order to quickly find good-quality solutions. Examples of heuristic algorithms for the VRP include the nearest neighbor algorithm, Clarke and Wright savings algorithm, and sweep algorithm. These algorithms are faster than exact algorithms but do not guarantee to find the optimal solution. However, they often provide reasonable solutions within a reasonable amount of time. Metaheuristic algorithms are high-level problem-solving strategies that guide the search for solutions by combining and modifying various heuristic techniques. They are designed to explore the solution space efficiently and often yield good-quality solutions for the VRP. Examples of metaheuristic algorithms include

genetic algorithms, ant colony optimization, simulated annealing, and tabu search. These algorithms provide a balance between solution quality and computational efficiency, making them suitable for larger VRP instances (Laporte, 2009).

Genetic algorithm is one of the common metaheuristics for solving VRP, which is inspired by natural selection. There are three main operators in genetic algorithms, including selection, crossover, and mutation which are used iteratively until a stopping condition is met. The steps of the genetic algorithm are summarized in Algorithm 2.

Algorithm 2: Genetic

---

1. Sort the population based on the fitness value, in ascending order.
2. Apply elitism to keep the best individual among direct survivors.
3. Select the rest of the direct survivors from the population using the selection process.
4. Select parents for crossover based on the selection process.
5. Perform crossover.
6. Generate mutations in the offspring by applying mutation operators.
7. Combine the direct survivors and mutated individuals to form a new generation.

1.6 Proposed Model:

Numerous enterprises, ranging from food delivery platforms to logistics companies, are investing in innovative solutions to meet the diverse needs of their customers. For instance, popular food delivery services like Uber Eats, DoorDash, or Deliveroo require efficient routing plans to ensure timely deliveries within specified time windows. Similarly, express delivery companies like FedEx, DHL, or UPS face the challenge of optimizing their delivery routes while adhering to time constraints set by customers. In fact, when the problem includes time window or time-dependent constraints, there is a need to estimate the travel time between different nodes. An ineffective travel time prediction in route planning is an important barrier to applications. In other words, planning

routes and on-time delivery requires an accurate estimation of travel time for each path at different times in order to be useful for real-world situations.

Data-driven predictions specifically deep learning and machine learning models have shown a greater performance in terms of accuracy and time rather than model-based ones. This is because machine learning or deep learning algorithms as a subset of artificial intelligence (AI), are automated processes that generate rules and patterns from the data like human learning behavior (Mostafavi & Sadighi, 2021).

To make it clearer, most of the time-dependent VRP papers, considered average speed, and distance to estimate travel time, while different factors like spatiotemporal characteristics, weather, driver behavior, etc. can influence the travel time of a path. Deep analysis of these factors by applying machine learning or deep learning algorithms leads to more accurate predictions (Wang et al., 2018). Deep learning models are highly effective in estimating travel times in time-dependent scenarios. These models excel at learning complex patterns and relationships from extensive data without explicit mathematical rules. By training on historical data that includes speed, time, road conditions, and other relevant features, deep learning models can capture the dynamic variations in travel times caused by factors like traffic congestion and road conditions. Deep learning's strength lies in its ability to handle non-linear relationships and adapt to different scenarios, making it ideal for accurately estimating time-dependent travel times in the Vehicle Routing Problem. Recently, recurrent neural networks (RNN) methods like long short-term memory (LSTM) and gated recurrent unit (GRU) have been successfully applied to traffic and travel time predictions. A large amount of global positioning system (GPS) trajectory data enables these deep-learning models to achieve better performance (Qiu et al., 2020; Ran et al., 2019).

According to literature reviews, an accurate and reliable travel time estimation is still demanding (Wang et al., 2018). So, there are two main purposes for this project. First, we aim to explore existing deep learning (DL) algorithms and work towards enhancing their accuracy in predicting travel times. Second, we apply the accurate travel time prediction using DL to solve the time-dependent vehicle routing problem with time window (TD-VRPTW) more practically and based on dynamic data.

Overall, many studies have focused on prediction algorithms in machine learning and deep learning and tried to improve the accuracy and time of predictions. On the other hand, there are lots of papers from the past until now analyzing and solving problems mathematically and with some simplifying assumptions. Combining deep learning models and traditional solutions is a constructive approach that can be used to improve existing traditional solutions. In other words, applying data-driven and deep learning prediction models offered unprecedented opportunities for researchers to improve traditional solutions and make the solutions more practical and effective in real circumstances.

The structure of the rest of the report is as follows: Chapter 2 provides a comprehensive review of the related scientific literature, focusing on travel time estimation, time-dependent vehicle routing problems with time windows, and the novelty of combining these two areas. Chapter 3 delves into the details of travel time estimation and the model architecture. Chapter 4 evaluates the performance of the travel time estimation model through numerical analysis, covering the dataset, parameter settings, and results. In Chapter 5, the focus shifts to addressing the time-dependent vehicle routing problem with time windows (TDVRPTW). Chapter 6 presents our hybrid metaheuristic algorithm designed to solve the TDVRPTW. The effectiveness of our metaheuristic

approach is analyzed in Chapter 7. Chapter 8 explores the integration of travel time estimation and the TDVRPTW. Finally, Chapter 9 provides a conclusion summarizing the key findings and contributions of the thesis.

Chapter 2

Literature review

2.1 Travel time estimation

Travel time estimation (TTE) has been studied extensively and can be divided into two groups based on the input query: route-based and origin-destination based. In other words, the input query can be a trajectory (sequence of locations) or only two points (origin and destination locations) (H. Wang, Tang, Kuo, Kifer, & Li, 2019). There are two approaches to estimating the travel time of a route: road segment and entire path method (D. Wang et al., 2018).

The classification of travel time estimation papers is shown in Figure 1.

In the road segment-based prediction, every route is divided into several road-segments and the goal is calculating the travel time of each road segment. Then, the travel time is simply the summation of the estimated travel time of different segments (Asif et al., 2014; de Fabritiis et al., 2008; Lv et al., 2015; Pan et al., 2012; Zhong, 2001). As the correlation between road segments affects the travel time of a path, some papers considered the relationships among adjacent road segments using Hidden Markov Model (Yang et al., 2013) or Predictive Regression Tree (PR-Tree) and Spatial-Temporal Probabilistic Graphical Model (STPGM) (D. Wang et al., 2016).J. Wang et al., 2016 implemented an error-feedback recurrent convolutional neural network (eRCNN) to accurately estimate the traffic speed of each road segment, using spatiotemporal information of neighboring road segments as input. The delay time at intersections is calculated using an interpolation method (Jensen & Larsen, 2010), a joint probability model (Hofleitner & Bayen, 2011), or a dynamic Bayesian network (Hofleitner et al., 2012) to concatenate road segment

travel times more accurately. Jenelius & Koutsopoulos, 2013 divided the travel time of a route into two parts: the individual travel time of segments and delay time due to intersections, traffic signals, turns, etc. Although these studies considered the time spent on the connection of different road segments, their main focus is the accurate estimation of individual road segment travel time or speed. Furthermore, another issue of road segment-based prediction is that the travel time error of the whole path can acquire a big number after summing up individual errors if the path consists of many road segments (D. Wang et al., 2018).

According to the weakness of the individual road segment-based TTE method, some researchers predicted the travel time of a route by mining historical data and calculating the average travel time of extracted frequent patterns (Han, Pei, & Yin, 2000; Luo, Tan, Chen, & Ni, 2013; Rahmani, Jenelius, & Koutsopoulos, 2013; Song, Sun, Zheng, & Zheng, 2014). This method also suffers from two issues: first, the historical average-based estimation may not be very accurate. Second, the historical data can not definitely include any or sufficient information for the searched path. Sometimes there is no trajectory passing the entire given path. This is called a data sparsity problem. To enhance the path-based TTE model and low sampling rate problem, Y. Wang et al., 2014 proposed a model called PTTE. They combined frequent sub-path travel times while optimizing the trade-off between the length of a sub-path and the number of historical trajectory data traveling it (i.e., support). Yuan et al., 2010, 2013 studied data sparseness and coverage using landmark graph. A landmark is defined as the top-k frequently traversed road segment based on historical data. They estimate the travel time between two landmarks whenever the historical data for each road segment is insufficient. However, this method can not be used for solving the data sparsity issue of roads with few traveled data since the landmarks are chosen from frequently traveled roads (D. Wang et al., 2018).

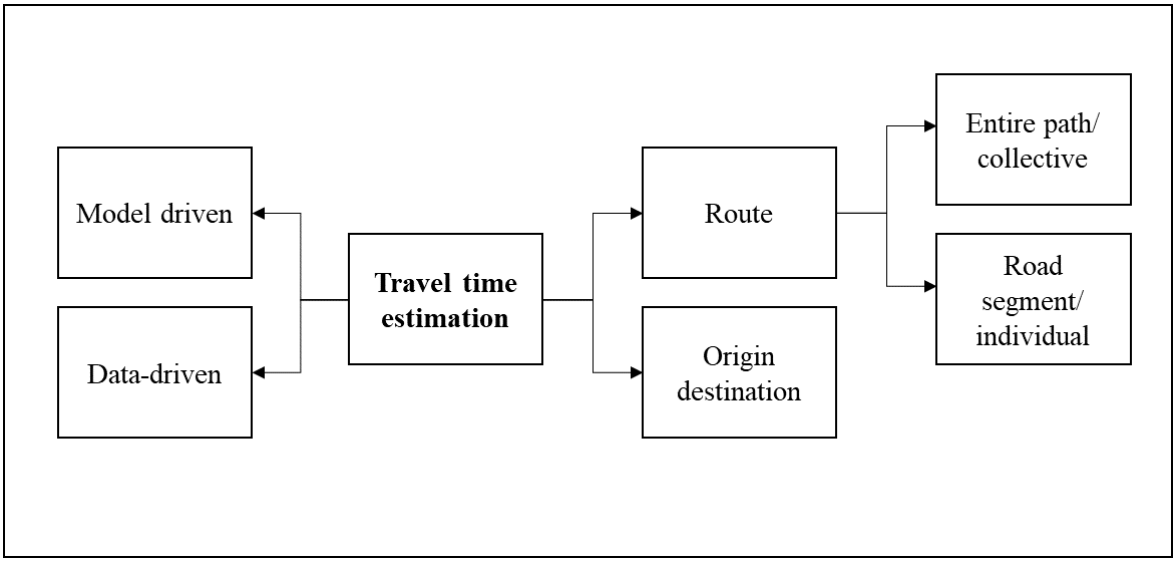

Figure 2: the classification of travel time estimation research

TTE models can be further classified into two types: model-based and data-driven (Bai et al., 2018). The model-based methods are built on a set of assumptions about the underlying relationship between the input and output variables like queuing theory (Ben-Akiva, Bierlaire, Burton, Koutsopoulos, & Mishalani, 2001) and Cell Transmission Model (Wan, Gomes, & Horowitz, 2014; Xiong et al., 2014) while data-driven methods rely solely on the input data to make predictions. The data-driven methods are categorized into three groups: statistical methods, basic machine learning methods, and deep learning methods (Pu et al., 2022).

Statistical models apply mathematical models and statistical assumptions for the prediction of traffic conditions such as ARIMA (Billings & Jiann-Shiou, 2006; F. Xu et al., 2016), Linear Regression (Kwon et al., 2000; Sun et al., 2003), Gaussian process (Y. Xu et al., 2019; Zhao & Sun, 2016), hidden Markov model (Luo et al., 2020; Zheng et al., 2018), Bayesian network (Y. Xu et al., 2014) and Kalman filter (Xie et al., 2007; T. dong Xu et al., 2012). However, the limitation

of statistical models including difficulty in handling large complex datasets and the non-linearity of spatial-temporal correlation features in traffic data motivated researchers to explore the machine learning models such as the k-nearest neighbor algorithm (KNN) (Murni et al., 2020), support vector machine (SVM) (X. M. Chen et al., 2012; Gao et al., 2016; J. Zhang & Sun, 2011), and artificial neural network (ANN) (Park & Rilett, 1999; M. Xu et al., 2019) to predict travel time.

Although these models estimate the traffic conditions more accurately and are relatively suitable for more complex data, they can not deal with the nonlinearity correlation problem. The advancement of deep learning models gives researchers the opportunity to apply deep learning-based methodologies for time series data processing. Specifically, deep learning is more frequently being used in traffic prediction tasks due to the impact of feature extraction on prediction accuracy and the power of deep learning models in extracting the spatiotemporal correlation characteristics. Some recent deep learning models are convolutional neural network (CNN) for feature extraction, recurrent neural network (RNN) and its variants, including long short-term memory neural network (LSTM) (Y. Liu et al., 2018), and gate recurrent unit neural network (GRU) for processing temporal information, graph convolution neural network (GCN) for feature extraction of non-euclidean structured data, attention mechanism for capturing long time span features, etc. Researchers also integrate different deep learning models to achieve higher accuracy in prediction like the DeepTTE model (Wang et al., 2018) which combines CNN and LSTM, the MCT-TTE model (F. Liu et al., 2022), which combines CNN and transformer models, and STGNN-TTE (Jin et al., 2022)which combines multi-stage spatiotemporal GCN and transformer layers.

In conclusion, this literature review highlights the existing approaches for travel time prediction and challenges. To tackle the previously identified difficulties, our research proposes an accurate

travel time estimation using a combination of TCN and LSTM to learn and extract both short- and long-term temporal information. The impact of external factors like time and driver's habits is also considered in this research. Regarding the two mentioned approaches of route-based TTE, we applied an attention mechanism and multitask learning module to predict the travel time for each road segment and entire route simultaneously, which leads to taking advantage of both methods. Finally, our model demonstrates improved accuracy in comparison to other state-of-the-art existing models. This study provides insights into the effective application of deep learning techniques for TTE and opens up avenues for further research in this area.

2.2 Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is a well-known combinatorial optimization problem that has been extensively studied in the literature. Researchers from various disciplines, including operations research, computer science, and transportation, have made significant contributions to addressing VRP and its variants. The first paper published on the Vehicle Routing Problem (VRP) was titled "The Truck Dispatching Problem" in 1959 (Dantzig & Ramser, 1959). In this seminal work, Dantzig and Ramser introduced the concept of the VRP and formulated it as a mathematical optimization problem. Their main objective in the paper was to address the problem of finding the most efficient routes for a fleet of trucks to deliver goods to a set of customers. Their aim was to minimize the total distance traveled by the vehicles in order to optimize the routing process. The Vehicle Routing Problem (VRP) is a generalized version of the Traveling Salesman Problem (TSP), which is a classic combinatorial optimization problem (Abdulkader et al., 2015). While the TSP focuses on finding the shortest Hamiltonian cycle that visits all given cities exactly once, the VRP extends this concept to include additional complexities and constraints related to vehicle

routing. The VRP requires not only determining the optimal sequence of visits to customers but also assigning them to appropriate vehicles while considering capacity limitations. There are different types of VRP, including Capacitated VRP (CVRP), VRPTW, multi-depot VRP (MDVRP), VRP with heterogeneous fleets (HFVRP), time-dependent VRP (TDVRP), Split Delivery VRP (SDVRP), and VRP with Pickup and Delivery (VRPPD). These are just a few examples of the various types of VRP that have been studied in the literature. Each variant introduces unique complexities and constraints, requiring specialized algorithms and solution approaches to address them effectively.

2.3 Vehicle Routing Problem with Time Windows (VRPTW)

The Vehicle Routing Problem with Time Windows (VRPTW) is an extension of the traditional Vehicle Routing Problem (VRP). In the VRPTW, an additional constraint is introduced where the service for each customer must be performed within specified time windows. This means that the delivery or service at each customer location has to be done within a specific time range. This constraint adds an extra level of complexity to the problem, requiring the optimization algorithm to consider both the route optimization and the timing constraints simultaneously (Zhong & Pan, 2007).

The VRPTW has gained attention as a research focus due to the increasing demand for time-sensitive services in various industries (El-Sherbeny, 2010). Examples include food delivery services and express delivery companies that require efficient routing plans to meet specific time windows. The VRPTW serves as a research hotspot to develop effective strategies for optimizing vehicle routes within designated time constraints (Tan et al., 2021).

2.4 Time-dependent vehicle routing problem with time window (TDVRPTW)

Travel times in urban areas can vary considerably throughout the day and week, which has a significant impact on the vehicle routing problem. Time-dependent vehicle routing problem with time window (TDVRPTW) considers the dynamic nature of travel times in real-world transportation systems. This model incorporates time-dependent travel times that vary based on the time of day, day of the week, or other factors. The problem aims to find the optimal set of routes to meet the demands of all customers within their preferred time windows while minimizing the total travel distance, time, or cost. TDVRP was initially proposed in 1992 (Malandraki & Daskin, 1992).

Some researchers consider discrete function for speed based on different time intervals, which assumes constant speed for each time interval (Gmira et al., 2021; Khanchehzarrin et al., 2021; C. Liu et al., 2020). On the other hand, there are some researches that focus on continuous function for vehicle speed changes such as the trigonometric function relation (Fan et al., 2021; Yeh & Tan, 2021).

To address TDVRPTW, many researchers have developed hybrid algorithms that combine multiple optimization techniques. Dabia et al., 2013 develop a branch-and-price algorithm to solve time-dependent vehicle routing problem with time windows TDVRPTW. The algorithm considers time-dependent travel times and dispatch times at the depot with an objective function of minimizing total route duration. Pan et al., 2021a suggest an efficient hybrid adaptive large neighborhood search with a tabu search algorithm to solve the duration-minimizing time-dependent vehicle routing problem with time windows (DM-TDVRPTW). Moreover, Hou et al., 2021 implement a multi-objective time-dependent VRPTW model which simultaneously

minimizes the total travel time and the number of vehicles used. A hybrid algorithm that combines a simulated annealing algorithm with a local search algorithm is applied to solve it. Wu & Wu, 2022 presents a model and algorithm for the time-dependent split delivery green vehicle routing problem with multiple time windows, considering both economic cost and customer satisfaction. Jie et al., 2022 propose a time-dependent vehicle routing problem (TDVRP) model with soft time windows and stochastic factors. To minimize the total cost of distribution and outperforms existing solutions in terms of the number of vehicles used, total transportation distance, and total waiting time, the authors used a hybrid algorithm combining Sweep Algorithm (SA) and Improved Particle Swarm Optimization (IPSO). Overall, the studies reviewed in this section demonstrate the importance of considering time-dependent travel times in the VRPTW and the effectiveness of using hybrid metaheuristic algorithms to solve the problem.

2.5 The novelty of combining travel time estimation using deep learning and VRP (TD-VRPTW)

To clarify that the research goals are innovative, it is essential to review what has been done concerning combining travel time prediction using machine learning (ML) and deep learning (DL) algorithms and the time-dependent vehicle routing problem with time window (TDVRPTW). A systematic literature review can provide a transparent, complete, integrated literature review. A systematic literature review summarizes and assesses all related works based on clear research questions (White & Schmidt, 2005). This literature review aims to specify what has been done in terms of combining the prediction of travel time specifically using deep learning, and solving TDVRPTW to prove the novelty of this study. To conduct a systematic literature review, Scopus is chosen as the database and the papers are selected based on specific keywords in their abstract

and title. The keyword search has been done in December 2022. Different combinations of the main keywords are considered. First, there are two different areas: Vehicle routing problem (VRP) and travel time estimation (TTE). The number of papers containing these two keywords is 11530 and 7298 respectively. Based on the purpose of this research, time-dependent (TD) and time window (TW) are two criteria added to the vehicle routing problem. The number of papers about VRPTW, TDVRP, and TDVRPTW is 3067, 294, and 127. On the other hand, the travel time estimation scope is limited to deep learning or machine learning models for the prediction of travel time, which includes 433 papers. However, the goal of this section is to show the novelty of combining these two primary research areas. To achieve this aim, we first searched for papers including VRP and TTE keywords in their abstract, keywords, and title. The result was only 30 papers. Regarding the combination of deep learning-based TTE and the different mentioned types of VRP (VRP, VRPTW, TDVRP), there are 5, 4, and 1 papers on Scopus. Moreover, there is no paper studying the combination of TDVRPTW and travel time estimation based on deep learning algorithms. It is worth noting that TDVRPTW seems a special rare problem, but in real-world conditions, the routing problem is always dependent on time. So, considering time-dependent and time window constraints can result in solving a more realistic problem. The results of keyword searches are shown in Figure 3. The related papers are reviewed and one of them is a proceeding paper. Y. T. Chen et al., 2021 only mentioned the importance of travel time estimation in VRP, but the focus of the research is predicting travel time using a new machine learning method, which is called the gradient boosting partitioned regression tree (GBPRT). Keskin, 2021 is a book chapter whose full text was unavailable. However, according to the abstract of this book chapter, they predicted travel time using support vector regression, which is an ML algorithm for solving VRPTW. Li et al., 2021 solved a ride-sharing optimization problem that is considered a robust

VRPTW. The travel time prediction is proposed using GRU (deep learning-based) model. Tarapata et al., 2022 estimated the travel time between the origin and destination using simple linear regression (machine learning) for different clusters of departure time intervals. They first portioned a day to different time intervals and applied linear regression for predicting the relationship between geographical distance and travel time. Then, these travel times are used in solving VRPTW based on a modified simulated annealing algorithm called SATM.

However, further research is needed to explore other ways to model the dynamic nature of real-world transportation systems and to develop more efficient algorithms for solving the time-dependent VRPTW. In the current study, the travel time is estimated based on the latest achievements in combining different deep learning-based algorithms for TTE, with high accuracy. This accurate estimation of travel time is applied to solve TDVRPTW, which is a promising solution for many transportation problems.

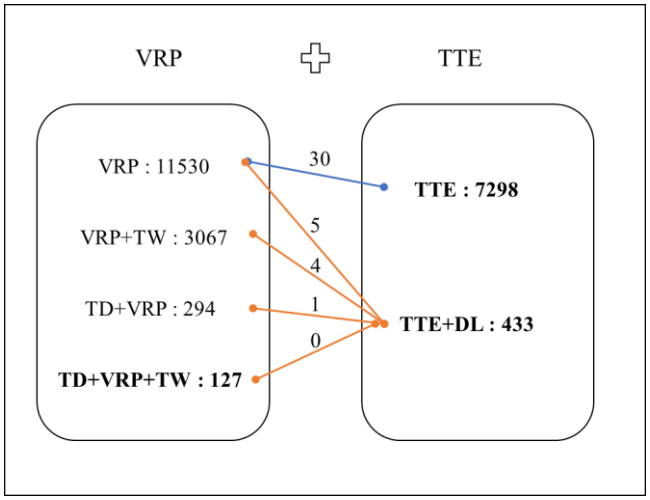

Figure 3: keyword search result (Scopus) about combining TTE and VRP

Chapter 3

Travel time estimation

3.1 Problem definition

In this study, our goal is to estimate travel times for road segments and entire routes based on trajectory data. We are interested in predicting how long it will take for a vehicle to travel between consecutive GPS points along a given trajectory. This information is valuable for various applications such as traffic management, route planning, and transportation optimization. To accomplish this, we utilize trajectory data consisting of latitude, longitude, time, and driver ID information. Each trajectory is composed of a sequence of GPS points that represent the vehicle's movement over time. We define a road segment as the route between three consecutive GPS points. By considering these road segments, we can divide the trajectory into smaller segments and estimate the travel time for each segment. We use different deep learning algorithms to analyze the data and learn from the patterns. This allows us to estimate travel times for individual road segments and the entire route concurrently.

In this section, we define the input and output of the problem.

Input: the input of our problem is a trajectory T, which consists of several consecutive GPS points Each GPS point has latitude, longitude, time, and driver ID elements. The link between three consecutive points is defined road segment.

Output: according to two methods discussed in the literature review, we predict the travel time for each road segment and entire route simultaneously. So, the outputs of our model for each

trajectory are estimated travel time for each road segment and the entire route. The entire route's travel time is calculated by a trainable weighted sum of road segments' travel time.

3.2 Model architecture

Our proposed framework includes three sections: attribute embedding, spatiotemporal learning, and multitask learning section. Figure 4 shows the architecture of our model. The input of the problem is a trajectory with latitude, longitude, time, and driver ID elements. The road segment is defined as the link between $k = 3$ consecutive GPS points. The output of the model is the predicted travel time for each road segment and the entire route.

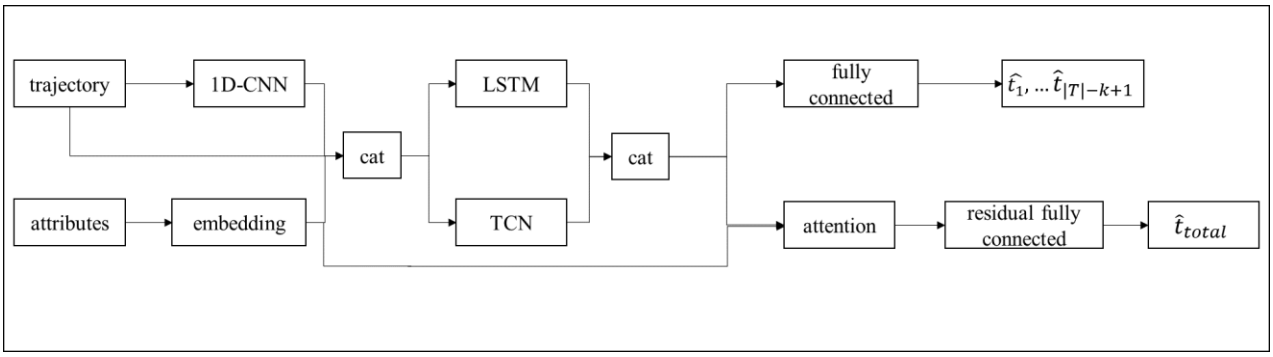

Figure 4: the overview of our proposed model

Attribute embedding: There are several elements that affect travel time like time of day, day of the week, the behavior of the driver, weather, etc. Extracting the pattern of these external factors and considering their impact on travel time using learning algorithms can result in a more accurate estimation. Three external elements are considered in this paper, which are timeID (time of a day: 0 to 1439 min), weekID (0 to 6), and driverID. These factors are categorial attributes that should be converted to numerical vectors using the embedding method.

Spatiotemporal learning: The spatiotemporal learning section aims to learn spatial and temporal correlations of GPS points. In this part, the model learns the special features of GPS data such as traffic lights, speed limit, turning, changing the speed at intersections, etc. Since the theses conditions are dynamic at different time of a day, we also need to capture temporal dependencies. First, to capture the spatial factors precisely, we apply a non-linear mapping and then a CNN layer (Wang et al., 2018). Instead of non-linear Geo-mapping of raw GPS data, some research mapped the GPS coordinates to a two-dimensional grid map and then the features of the grid map are extracted as images (Q. Liu et al., 2016; Wang et al., 2016). However, the spatial information is not accurately represented by the direct mapping of GPS points to the grid cells. So, we apply a non-linear function to map the GPS point to the *traj* vector based on Equation (1).

$$traj_i = \tanh(W_{traj}.(p_i.lat \circ p_i.lng)) \quad (1)$$

In equation (1), p_i indicates i^{th} GPS point in the trajectory $T = \{P_1, \dots., P_{|T|}\}$, the $\circ$ indicates the concatenate operation and W_{traj} is a learnable matrix with a size of $16 * |T|$. $|T|$ is the number of all GPS points in the given trajectory.

A usual approach to capture spatial dependencies is the convolutional neural network (CNN), which is mostly used for image or object processing. Thus, the output of non-linear mapping ($traj_i \in R^{16*|T|}$) is given to 1D-CNN with a kernel size of 3 (k=3) to create the feature map in Equation (2). It is worth mentioning that such input can be considered as a 16-channel input to the Conv layer. The output of the CNN module is a vector with a size of $16 * (|T| - k + 1)$ based on the CNN structure.

$$loc_i = Conv1D(traj_i) \quad (2)$$

As the total distance and other embedded external attributes significantly influence the travel time, they are concatenated to the output of the CNN module, shown in Equation (3). Furthermore, the input of the CNN layer is concatenated with its output through skip connections to keep important information from the input, enabling the network to better learn the relationships between the input and output.

$$loc_f = (loc_i \circ dist \circ attr)^\circ traj_i \quad (3)$$

To capture the temporal dependencies of GPS points, two layers of LSTM and temporal convolutional network are implemented. LSTM is a variant of RNNs, which have three gates, including input, forget, and output gates (Cha et al., 2023; Mostafavi & Cha, 2023). These gates can help the model to memorize important information and forget unimportant ones, allowing the network to capture long-term dependencies. On the other hand, a temporal convolutional network (TCN) is a type of CNNs for processing sequential data. It is suitable for modeling complex temporal patterns in the data since it is able to Capture local and short-term dependencies in the data and learn hierarchical representations of the data. By applying both TCN and LSTM, the model can effectively benefit from the strengths of both methods, which result in a more complete and precise depiction of the time-related information. Therefore, combining these two methods can enhance the accuracy of travel time estimation. After these temporal learning layers, we have the sequence of hidden states at different time steps ($\{h_1, \dots, h_{|T|-k+1}\}$).

$$h_i = LSTM\left(W_h . h_{i-1} + W_c . loc_f\right) \circ TCN_i\left(loc_f\right) \quad (4)$$

Multitask learning: the multitask learning part uses the spatiotemporal learning section's output $\{h_1, \dots, h_{T-k+1}\}$ to predict the travel time for each road segment and the entire route. The segment travel time ($\hat{t}_i$) is predicted through fully connected layers, while the attention layer and residual fully connected layers are used to predict the entire path travel time ($\hat{t}_{total}$). The attention layer is a way to sum all segments' travel time with different weights based on their impact on the entire path. To calculate the weights of hidden states, the attention mechanism considers the external attributes and extracted spatiotemporal features of each segment.

Chapter 4

Numerical Analysis for TTE

4.1 Data

To compare the results of our travel time prediction model and several existing methods one large-scale dataset, the Chengdu dataset (the same as (Wang et al., 2018)), is selected in this research. This dataset is common in many papers based on the literature review, which helps valid comparison. Chengdu Dataset consists of 9737557 trajectories (1.4 billion GPS records) of 14864 taxis in August 2014 in Chengdu, China. However, limited computation resources make it challenging to use all raw trajectory data. We sampled 1,150,000 trajectories to test the performance of different algorithms and compare them. Before a prediction model, it is essential to process raw data since improper data records like outliers, and missing values can negatively influence the accuracy of prediction models (Zheng, 2015). The raw dataset contains driver ID, latitude, longitude, states, and timestamp columns, which are illustrated in Figure 5. The input to the prediction model is GPS data in the form of trajectories. The processing of the data involves converting GPS points into meaningful trajectories and omitting outliers based on time and distance difference of consecutive points, and traffic-controlling criteria. First, the GPS data was sorted based on time and the time and distance difference of each consecutive point are calculated. Then, if the time and distance difference of two consecutive points exceeds 1800 seconds or 1 km, or the driver ID was not the same, these two consecutive points are assigned to two different trajectories.

Second, some traffic-controlling criteria were defined and applied to create more meaningful trajectories and remove outliers. These criteria are set based on typical characteristics of urban

trajectory. The following are some of the criteria: the travel distance greater than 100km or less than 0.5km, the average speed greater than 120km/h or less than 5km/h, and the travel time greater than 7200 seconds or less than 60 seconds (F. Liu et al., 2022). In the end, the remaining trajectories are partitioned randomly (based on Uniform/Normal distribution) to create polylines with lengths of 11 to 128 points.

Driver ID	Lat	Lng	State	Time
1	30.624806	104.136604	1	2014-08-03 9:18:46 PM
1	30.624809	104.136612	1	2014-08-03 9:18:15 PM
1	30.624811	104.136587	1	2014-08-03 9:20:17 PM
1	30.624811	104.136596	1	2014-08-03 9:19:16 PM
1	30.624811	104.136619	1	2014-08-03 9:17:44 PM
1	30.624813	104.136589	1	2014-08-03 9:19:46 PM
1	30.624815	104.136585	1	2014-08-03 9:21:18 PM
1	30.624815	104.136587	1	2014-08-03 9:20:48 PM
1	30.624815	104.136639	1	2014-08-03 9:17:14 PM
1	30.624816	104.136569	1	2014-08-03 9:22:50 PM
1	30.624816	104.136574	1	2014-08-03 9:22:19 PM

Figure 5: raw data of Chengdu dataset

4.2 Parameter setting

In learning algorithms, data should be shuffled in three sets: train, development, and test. Training data is used for learning the model parameters. The development set is for checking the accuracy of different models and comparing them to find the best one, while the test set is unseen data for the best model to evaluate its performance. Considering the huge number of data (more than 1 million), the dataset is divided into 90% training set, 5% dev set, and 5% test set. Other parameters for the experiment are shown in Table 1.

Table 1: detailed parameter setting for TTE

Section	parameter	value
Embedding	Week ID dimension	R^4
	Time ID dimension	R^9
	Driver ID dimension	R^{17}
Spatial_CNN	Number of filters	48
	Activation function	Elu
Temporal_LSTM	Hidden size	128
	Number of layers	2
Temporal_TCN	Activation function	Tanh and sigmoid
	Number of layers	2
Multitask learning_entire route	Activation function	Leaky relu
	Hidden size	64,64,64 with residual connection
Multitask learning_road segment	Activation function	Leaky relu
	Hidden size	64 ,32

The optimization algorithm for training parameters is Adam and the learning rate is scheduled based on an exponential decay formula with an initial learning rate of 0.001 and decay rate of 0.98. The batch size during training is 100 and we train the model for 60 epochs and the best model is selected based on validation loss. The training and evaluation process is conducted on a workstation with a GPU (NVIDIA TITAN V) with 64 GB of available RAM.

4.3 Result and discussion

The objective function to train different models is set to minimize mean absolute percentage error (MAPE), which is a relative loss metric suitable for both long and short paths. The model is

trained by considering a weighted sum of road segment and entire route losses as the loss function. To evaluate our proposed model, we compared it with four models:

- DeepTTE: which uses a Geo-conv layer and LSTM to capture spatial and temporal dependencies (Wang et al., 2018).
- RNNTTE: a streamlined version of DeepTTE, which uses basic RNN instead of LSTM (Wang et al., 2018).
- Transformer: this has the same attribute and multitask learning sections as the proposed model but utilizes a 6-layer encoder transformer with 4 heads to capture temporal information.
- GCN-2TCN: which uses a graph convolutional network instead of 1D-CNN to capture spatial correlations and only two layers of TCN for temporal learning.

Table 2: performance comparison

Model	MAPE (%)
DeepTTE	8.714
RNNTTE	8.899
Transformer	11.093
GCN-2TCN	9.859
Our algorithm	**5.787**

Our algorithm was found to be more accurate than others based on MAPE results provided in Table 2. The most important section for prediction of travel time is learning spatiotemporal features which has been conducted using CNN, LSTM, and TCN in our method. The result of different algorithms with different structures for spatiotemporal correlation extraction shows the importance

of this module for prediction. Taking advantage of extracting short- and long-term dependencies using LSTM and TCN respectively results in significant improvement in estimation's accuracy.

Chapter 5

Time-dependent vehicle routing problem with time window

5.1 Problem definition

The TDVRPTW problem involves a directed graph $G = (V, A)$, where V represents the set of nodes including the depot (node 0) and a set of customers ($V_c = \{1,2, \dots, n\}$). Each node i in V is associated with a demand d_i, a service time s_i, and a time window $[e_i, l_i]$, within which the service must be started. In our problem, vehicles are permitted to arrive at a customer location before the start of the specified time window, denoted as e_i. In such cases, the vehicle is required to wait until e_i to begin the service. However, it is strictly prohibited for the vehicle to arrive at the customer's location after l_i, as late arrivals are not allowed. This constraint ensures that the service starts within the designated time window while allowing for some flexibility in early arrivals and waiting periods. It is worth noting that the waiting time contributes to the overall time duration of the routes.

The depot (node 0) has zero demand and service time. Additionally, time window for the depot $[e_0 = 0, l_0]$ corresponds to a typical workday. This means that the depot is available for service from the start of the workday ($e_0 = 0$) until the end of the workday (l_0). The arc set A consists of arcs (i, j) where i and j are nodes in V, representing the connections between nodes. Each arc (i, j) has a Euclidean distance $dist_{ij}$ and a time-dependent travel time function.

The problem assumes an unlimited fleet of homogeneous vehicles with capacity Q. K is the set of vehicles. The vehicles start from and return to the depot, and their capacity must not be exceeded at any node. The objective of the TDVRPTW problem, also known as DM-TDVRPTW (Duration

Minimizing Time-Dependent Vehicle Routing Problem with Time Windows), is to minimize the total duration of the routes needed to serve all customers while taking into account various constraints. Factors such as the time windows within which each customer must be served, the service times required at each customer location, the time-dependent travel times that vary based on the time of day, and the capacities of the vehicles used for serving the customers are considered in this problem.

To address the problem more conveniently, an additional node $n + 1$ is introduced to represent the returning depot. This creates a new graph $\overline{G} = (\overline{V}, \overline{A})$ where $\overline{V}$ includes V along with node $n + 1$ ($\overline{V} = V \cup \{n + 1\}$), and $\overline{A}$ includes A along with arcs $(i, n + 1)$ connecting each node i in V to the returning depot node ($\overline{A} = A \cup \{(i, n + 1) \mid i \in V_c\}$).

5.2 Model architecture

Mathematical model: According to problem definition, the TDVRPTW problem seeks to optimize the routes of homogeneous vehicles to minimize the total duration required to serve all customers, while considering constraints related to time windows, vehicle capacities, and time-dependent travel times.

The model's decision variables consist of the following:

- x_{ijk}: binary variable, which is 1 if the vehicle travels from the customer i to the customer j using vehicle k, otherwise, it is 0.
- d_dt_k: departure time from the depot for vehicle k
- dt_i: departure time from customer i
- y_{ik}: binary variable, which is 1 if the customer i is served by vehicle k

- z_i: binary variable, which is 1 if there is an early arrival at customer i, otherwise, it is 0.
- a_i: arrival time to customer i
- w_i: waiting time at customer i
- v: number of vehicles

The mathematical formulation for the TDVRPTW is established in the following manner:

$$Min\ z = \sum_{i=1}^{n}\sum_{k=1}^{v} a_{n+1}\ .\ x_{i(n+1)k} - \sum_{k=1}^{v} d\_dt_k \tag{5}$$

$$\sum_{i=0}^{n} x_{ijk} = y_{jk} \qquad \forall j \in V_c, k \in K\ , i \neq j \tag{6}$$

$$\sum_{j=1}^{n+1} x_{ijk} = y_{ik} \qquad \forall i \in V_c, k \in K\ , i \neq j \tag{7}$$

$$\sum_{k=1}^{v} y_{ik} = 1 \qquad \forall i \in V_c \tag{8}$$

$$\sum_{j=1}^{n} x_{0jk} = 1 \qquad \forall k \in K \tag{9}$$

$$\sum_{i=1}^{n} x_{i(n+1)k} = 1 \qquad \forall k \in K \tag{10}$$

$$\sum_{i=1}^{n} d_i\ .y_{ik}\ \leq Q \tag{11}$$

$$a_i \leq l_i \tag{12}$$

$$e_i - a_i \leq M.z_i \tag{13}$$

$$a_i - e_i \leq M.(1 - z_i) \tag{14}$$

$$w_i = z_i.(e_i - a_i) \tag{15}$$

$$dt_j = \sum_{k=1}^{v} d\_dt_k \,.\, x_{0jk} + \sum_{k}^{v}\sum_{i}^{n} x_{ijk} \,.\, (a_i + w_i + s_i) \qquad \forall j \in V_c \tag{16}$$

$$a_j = \sum_{i}^{n}\sum_{k}^{v} x_{ijk} \,.\, \left(dt_i + TT_{ijk,dt_i}\right) \quad \forall j \in \bar{V} \tag{17}$$

$$v \leq \max \_v \tag{18}$$

$$x_{ijk} \in \{0,1\}, y_{ik} \in \{0,1\}, z_i \in \{0,1\} \tag{19}$$

$$d\_dt_k \geq 0, dt_i \geq 0, a_i \geq 0, w_i \geq 0, v \geq 0 \tag{20}$$

Equation (5) is the objective function, which represents minimizing the overall duration of the routes, which is calculated as the time difference between the arrival at the returning depot ($n + 1$) and the departure from the starting depot (0). Equation (6) and (7) ensures that each customer has only one predecessor and one successor. Equation (8) indicates that each customer can be assigned to a single vehicle for service. Equation (9) and (10) makes sure that each vehicle starts from and return to the depot. Equation (11) ensures that the maximum capacity of each vehicle is not exceeded. Equations (12-(17) are time window constraints, including not allowing late arrival, calculation of waiting time if there is any early arrival, arrival time to each customer and departure time from each customer. It is worth noting that in this case, the departure time from depot for each vehicle (d_dt_k) can be any value in working hour of depot, but departure time from each customer is calculated using the summation of arrival time to the customer, waiting time and

service time of the customer. The arrival time for each customer is measured by summing the departure time from the previous customer and the travel time from the previous customer to the current customer. The meaning conveyed by Equation (18) is that the optimization of routing ensures that the number of vehicles utilized does not surpass the maximum available number of vehicles (max_v). Equation (19) shows the binary decision variables and Equation (20) represent the non-negativity constraints of other decision variables.

In the TDVRPTW, the calculation of travel time (TT_{ijk,dt_i}) incorporates a time-dependent approach based on dividing the workday into five distinct and non-overlapping time zones. This division is designed to simulate the morning and evening peak periods commonly observed in real-world transportation. Since peak and off-peak periods generally are the same across different roads, the time interval periods are used for all arcs. However, there are diverse characteristics of various road types. Therefore, three speed profiles are utilized. These profiles represent expressways with fast links, central business districts and city areas with slow links, and all other roads with normal links. The assignment of arcs to specific speed profiles is randomly based on the approach established by Ichoua et al. (2003). In other words, for each pair of nodes (i, j), a cluster index is assigned, representing the speed profile category (slow, normal, fast). This cluster index helps determine the appropriate speed profile for the given arc. The speed profiles and time zones are shown in Table 3.

Using the appropriate speed profile and corresponding time interval for departure time, the function identifies the initial speed for the travel time calculation. This initial speed is then used as the starting point for the calculation of the total travel time. With the initial speed determined, the total travel time is calculated by dividing the traveled distance by the speed within each time

interval. The function iterates through the time intervals. During each iteration, the function evaluates the current time interval and its associated speed profile. It calculates the portion of the distance that can be covered within that interval, taking into account the corresponding speed. This process continues until the entire distance is covered. Ultimately, the function returns the total travel time, enabling efficient vehicle routing solutions that adapt to time-dependent travel speeds and time windows.

Table 3: Travel speed profile for 5 time zones

	Zone1	Zone 2	Zone 3	Zone 4	Zone 5
Time range	0 - 0.2 l_0	0.2 l_0 - 0.3 l_0	0.3 l_0 - 0.7 l_0	0.7 l_0 - 0.8 l_0	0.8 l_0 - ∞
Fast	1.5	1	1.67	1.17	1.33
Normal	1.17	0.67	1.33	0.83	1
Slow	1	0.33	0.67	0.5	0.83

Chapter 6

Solution approach for TDVRPTW

According to the literature review of TDVRPTW, a hybrid algorithm that combines different optimization techniques are mostly used in recent years and have been shown to be effective for solving TDVRPTW. In this paper, we are planning to apply a hybrid Genetic algorithm with Local Search (LS) and Large Neighborhood Search (LNS) to solve TDVRPTW.

6.1 Metaheuristics Background

A genetic algorithm is a type of evolutionary optimization algorithm inspired by the principles of natural selection. First, the genetic algorithm starts with an initial population of candidate solutions or individuals. Each candidate is represented by a set of genes, which can be thought of as variables or parameters of the problem being solved. The genetic algorithms evolve new individuals over successive generations using genetic operators like selection, crossover, and mutation. During the selection process, the individuals from the population are selected for reproduction and further genetic operations. In crossover, two individuals are combined to create a new offspring by exchanging parts of their genes. In mutation, a small random change is applied to the genes of an individual to create a new offspring. The offspring are then replaced within the population, and the process of selection, crossover, and mutation is repeated over successive generations until a stopping criterion is met, such as a maximum number of generations or a satisfactory solution is found. The fitness function is defined for evaluating the individual and quality of the solution (Agrawal et al., 2022).

Local search algorithms, including large neighborhood search algorithms, are optimization methods with the purpose of improving a given solution by iteratively searching the neighborhood of the current solution. In other words, these algorithms apply local search operators to modify the solution in small increments. The objective is to navigate through different solutions within the neighborhood, searching for better alternatives that optimize the given solution. By continuously exploring and evaluating neighboring solutions, local search metaheuristics have the potential to find high-quality solutions to complex optimization problems (Abdulkader et al., 2018; Konstantakopoulos et al., 2022).

Large neighborhood search is a type of local search that explores a wide search space. It starts by partially destroying the current solutions using removal operators, which determine which components should be temporarily removed from the solution. Then, the partial solution is repaired using reinsertion operators, which determine the appropriate manner in which the previously removed components are reintegrated into the solution (Azi et al., 2014; Shaw, 1998; S. Zhang et al., 2018)

6.2 Proposed Hybrid Genetic Algorithm

The proposed algorithm for solving the Time-Dependent Vehicle Routing Problem with Time Windows (TDVRPTW) combines three powerful techniques: Genetic Algorithm (GA), Local Search (LS), and Large Neighborhood Search (LNS). This hybrid approach aims to leverage the strengths of each technique to find high-quality solutions for the TDVRPTW. The framework of our proposed model is illustrated in Figure 6 and then each step is explained in detail.

The genetic algorithm is our main algorithm to solve TDVRPTW. The genetic algorithm proceeds by sorting the population based on the fitness values of each individual. Then, the number of

parents for crossover is determined, ensuring an even number by adjusting if necessary. This is done by calculating a ratio of crossover to the total population size. The number of direct survivors is calculated by subtracting the number of parents from the total population size. To ensure the preservation of the best solution from the previous generation, elitism is applied. This involves selecting the individual with the lowest fitness value, which represents the best solution, from the sorted population. This selected individual is then included among the direct survivors who will be part of the new generation. The rest of the direct survivors are chosen from the population using selection process. Parents are then chosen for crossover, where their genetic information is combined to create offspring. To introduce diversity, mutations are applied to the offspring. The new generation is formed by combining the survivors, offspring, and mutated individuals. This process is repeated for multiple generations, allowing the algorithm to explore the search space and exploit promising solutions, ultimately converging toward an optimal or near-optimal solution.

Moreover, the trade-off between intensification and diversification in optimization algorithms is an essential factor to consider. Intensification emphasizes exploiting current solutions to refine them locally and achieve faster convergence towards improved solutions, while diversification focuses on exploration and the search for new regions to maintain diversity and prevent early convergence to suboptimal solutions. Striking the right balance between the two is crucial to avoid getting trapped in local optima while still making progress towards high-quality solutions. Genetic algorithms are primarily used for both exploiting and exploring the solution space. Local search operators aim to exploit the solutions. LNS can be used differently for intensification, which is called Large Neighborhood Search as Local Search, or for diversification, which is called Large Neighborhood Search as Mutation in this research.

In addition to the mentioned techniques, throughout the algorithm's execution, the three best solution quality achieved so far is tracked. This enables the algorithm to discard inferior solutions and focus on improving the best-known solutions, promoting convergence towards a high-quality solution (Gajpal & Abad, 2009). Moreover, To maintain diversity and introduce new genetic material into the population, the discarding method is utilized to replace similar solutions in terms of the value of fitness function with new individuals.

By combining the genetic algorithm and these techniques, including the initial population generation, discarding similar solutions, the best solution tracking, local search, and large neighborhood search, the algorithm can efficiently navigate the solution space, balancing exploration and exploitation, and ultimately find high-quality solutions for this challenging optimization problem.

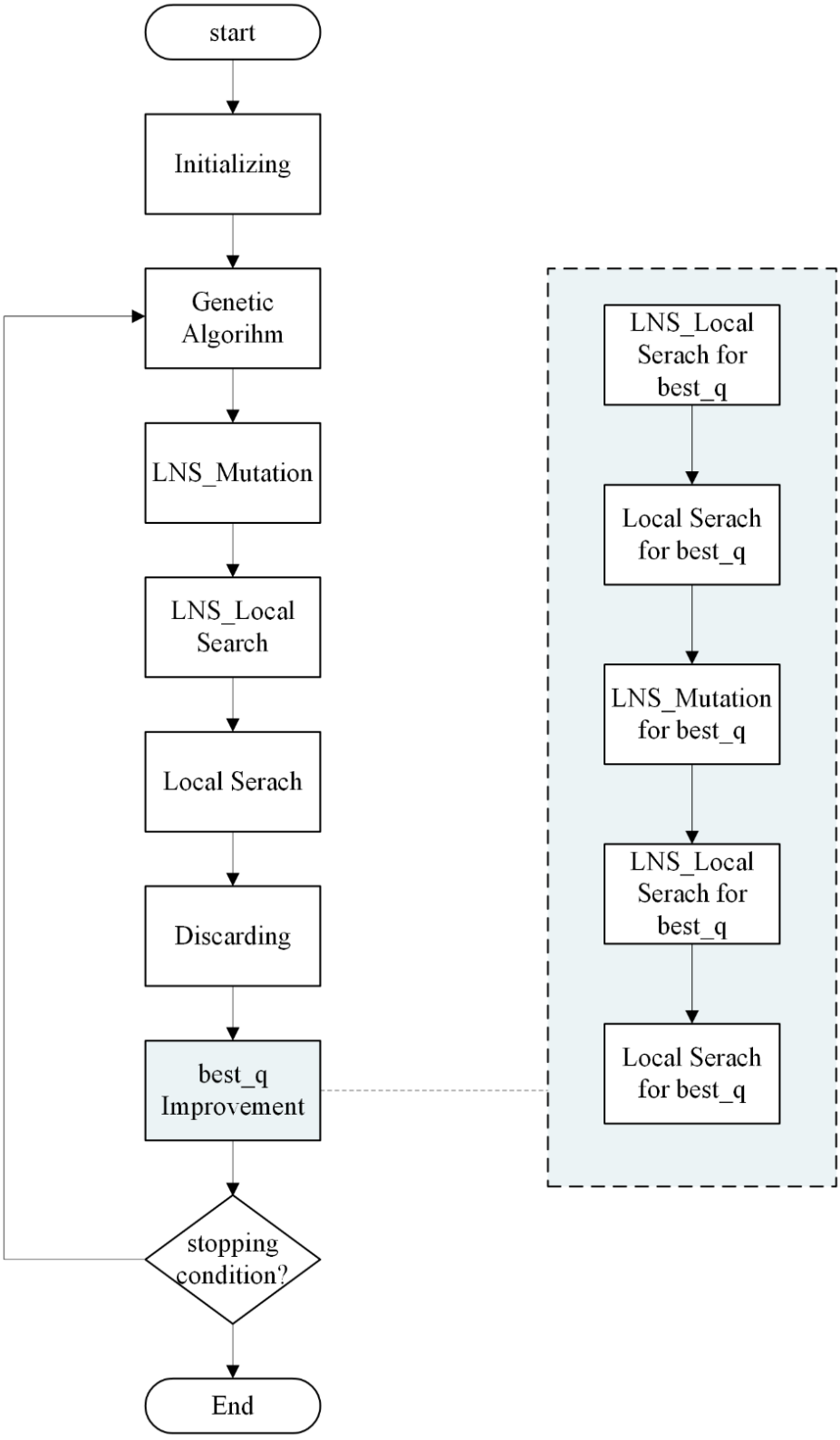
start
Initializing
Genetic Algorihm
LNS_Mutation
LNS_Local Search
Local Serach
Discarding
best_q Improvement
stopping condition?
End
LNS_Local Serach for best_q
Local Serach for best_q
LNS_Mutation for best_q
LNS_Local Serach for best_q
Local Serach for best_q

Figure 6: flowchart of our metaheuristics to solve TDVRPTW

Below are the detailed explanations for each step in our proposed method:

6.2.1 **Initial Population**

The algorithm begins with an initial population of candidate solutions. The initial population is created using various techniques such as random generation or heuristic methods. In this project, two different methods are utilized to create the initial population of chromosomes for a genetic algorithm: random shuffling and k-regret reinsertion.

In the random shuffling approach, chromosomes are created by randomly selecting genes from a set of available genes. These genes represent customers that need to be serviced. The chromosome is constructed by including the depot (0 representing the starting depot and n+1 representing the returning depot) and a random subset of customers. Additionally, departure times from the depot for each route are assigned randomly within the allowed time window limits of the depot, which is a typical workday.

In the k-regret reinsertion approach, instead of randomly selecting customers, customers are inserted based on the k-regret reinsertion heuristic. In order to gain a better understanding of the regret method, it is essential to first delve into the concept of the greedy method. The greedy method is a straightforward approach where, at each step, the next decision is made based solely on the current best option without considering future consequences. In the context of insertion heuristics, the greedy method calculates the minimum insertion cost for each unassigned customer of the solution. The insertion cost is the difference between the total cost of the solution with and without the inserted customer. The customer with the lowest insertion cost is chosen for insertion. However, this method results in delaying the assignment of costly customers, which leads to

limited choices and higher costs toward the end of the process. Regret heuristics are introduced as an enhancement to the greedy method to mitigate this limitation. Regret heuristics take into account not only the minimum insertion cost but also the cost differences of subsequent alternative insertions. To calculate regret value, $k+1$ best routes to place customer i is specified based on insertion cost ($c_{i,1},..,c_{i,k+1}$). The regret value for each customer is calculated based on aggregated differences between the best insertion cost and subsequent k cheapest insertions, which are shown in Equation (21). The customer with the highest regret value is prioritized for insertion. By selecting the customer with the highest regret value, the method aims to make an informed decision for achieving higher solution quality. Considering k best routes and cumulative regret value allows the algorithm to look ahead and select customers based on a broader perspective. The value of k, which determines the level of lookahead, is randomly chosen between 1, 2, or 3 in the construction and repair stages of the algorithm. Overall, the k regret insertion method provides a more sophisticated approach than simple greedy insertion, which leads to better performance on complex TDVRPTW problems (Pan et al., 2021b).

$$RV_i = \sum_{j=2}^{k+1} c_{i,j} - c_{i,1} \qquad (21)$$

The method for generating initial solutions is selected using a specified probability value. This allows for a combination of random shuffling and k-regret reinsertion in the initial population, promoting diversity among the chromosomes. The presence of diverse elements sets the foundation for the genetic algorithm to explore different areas of the search space. This enhances the likelihood of finding optimal or near-optimal solutions during the optimization process.

6.2.2 Selection operators

Three different selection methods are used in this implementation, including tournament selection, random selection, and proportional selection. Tournament selection involves randomly selecting a subset of individuals from the population and comparing their fitness values. Among the selected subset, the best individual in terms of fitness is chosen as a winner. This procedure is iteratively performed for a specified number of times to create a set of selected individuals. Random selection simply chooses a specified number of individuals from the population randomly, without considering their fitness values. The main benefit of this method is adding randomness, which helps maintain diversity in the population. Proportional selection, also known as fitness proportionate selection or roulette wheel selection, is a stochastic selection method that assigns probabilities to individuals based on their fitness values. The higher probability of being selected is assigned to individuals with better fitness. This selection method involves spinning a roulette wheel, where each individual's slice of the wheel corresponds to their probability of being chosen. Multiple spins are conducted to select the desired number of individuals. Based on the same selection process, parents are selected to perform crossover.

6.2.3 Crossover operators

Two different methods are used as crossover operators. The first method employs a single parent and randomly divides it into several pieces, which are then shuffled to create a new ordering, resulting in a single offspring. The second method involves selecting two parents and swapping a middle segment between them. However, to ensure that all customers are visited exactly once, a repair process is applied to the offspring solutions to fix missing and duplicate nodes in the solutions. After the crossover operation, $child1$ is initially a copy of $parent1$, but it may lack

certain genes present in $parent2$. To rectify this, the missing genes from $parent2$ are identified. These missing genes are the ones that exist in $parent2$ but are not present in $child1$. Once the missing genes are identified, the algorithm determines their appropriate positions in $child1$. The appropriate position for each missing gene is the position it occupies in $parent2$. This means that the missing gene from $parent2$ should be inserted into $child1$ at the same index where it exists in $parent2$. This ensures that the genetic information from $parent2$ is accurately incorporated into $child1$. An example of crossover operators with one and two parents are shown in Figure 7 and Figure 8.

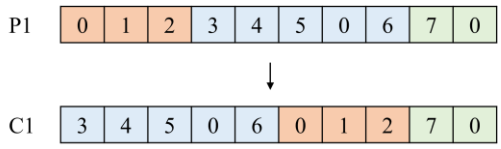

Figure 7: crossover using 1 parent.

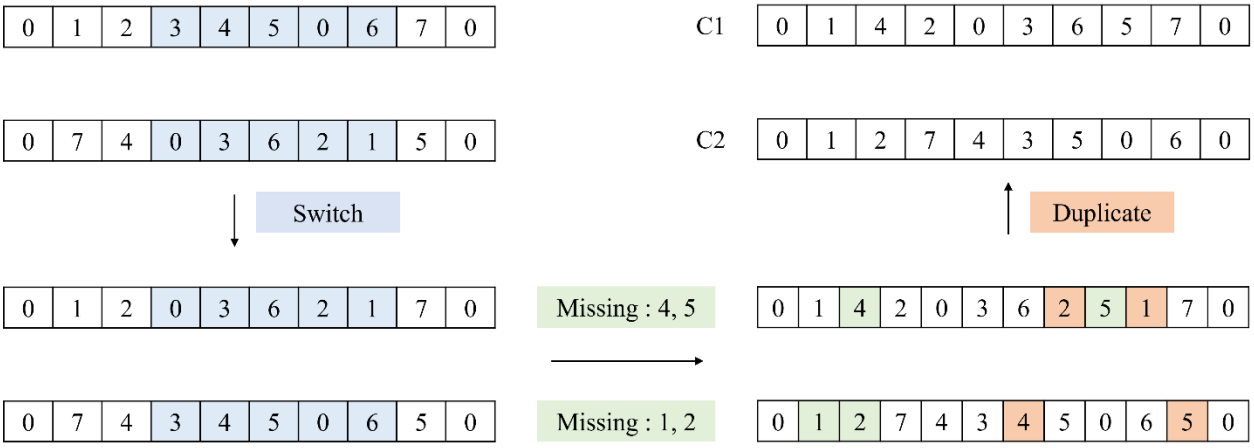

Figure 8: crossover using 2 parents.

In addition to the crossover operators for generating offspring solutions, the optimization algorithm also incorporates the "blend crossover" method to determine the departure time (dt) values, which

are another set of decision variables. The blend crossover operation combines dt values from the parent solutions using a weighted average based on the blending factor alpha. This approach allows for the exploration of different combinations of dt values and inherits favorable characteristics from the parent solutions to potentially improve the offspring solutions' quality. The output of the crossover operation undergoes the mutation process.

6.2.4 **Mutation operator**

The mutation probability is adaptively adjusted based on the average fitness of the population, ensuring individuals with a fitness greater than average fitness have a higher chance of being selected for mutation. Three mutation methods are employed: inversion mutation, swap mutation, and scramble mutation, each with an equal probability. In inversion mutation, a random range within the chromosome is selected and the order of elements within that range is reversed. Swap mutation randomly exchanges the elements at two positions within the chromosome. Scramble mutation shuffles the elements within a randomly selected range. This diverse set of mutation methods promotes exploration and facilitates the discovery of improved solutions by altering the gene order. Additionally, within the mutation process, if a gene represents a depot (identified by a value of 0), there is a 50% chance of modifying its associated departure time (dt) value. The dt value is randomly assigned a new feasible value within a predefined time range for the depot. This stochastic modification of dt values adds variability to the solutions and enables the algorithm to explore different scheduling possibilities, enhancing its ability to find improved solutions.

Finally, the new generation is formed by combining the directs (including the elite individual) with the resulting population after the mutation process.

6.2.5 Local Search

In our approach, we utilize various local search operators, including inter relocate, intra relocate, inter swap, and block insertion for finding the best sequence (Gajpal et al., 2017; Peng et al., 2019), and local search_departure time for optimizing departure time.

Inter Relocate: This operator involves moving a customer from one route to another. It searches for the best route and position to relocate the customer, considering all possible routes except the current one. The goal is to find a relocation that improves the overall fitness of the solution.

Intra Relocate: This operator involves moving a customer within the same route. It searches for the best position to relocate the customer within its current route, aiming to improve the fitness of that specific route.

Inter Swap: This operator involves swapping two customers between two different routes. It searches for the best pair of customers and the best routes to perform the swap, aiming to achieve an improvement in the fitness of both routes.

Block Insertion: This operator involves inserting a consecutive sequence of customers from one route into another route. It searches for the best position and the best route to insert the block of customers, aiming to improve the overall fitness of the solution.

Figure 9 illustrates these local search operators for changing the sequence of visiting customers. Intra relocate function changes the position of selected customer 1 to new position 3 in the same route. The process of inter-relocation involves the movement of customer 1 to another route (for instance route 1 and at the position of 2 in the new route. Inter swap operator includes switching the location of two customers in different routes. Block insertion is the same as inter relocate, with

the only difference that instead of one customer, a block of consecutive customers with 2 or 3 customers is chosen for relocation to the new route.

These operators are specifically designed for intensification, allowing us to refine the solution by making small changes to the routes, such as reordering or relocating customers. Moreover, a dedicated local search is applied to optimize the departure time for each route, further enhancing the overall solution quality.

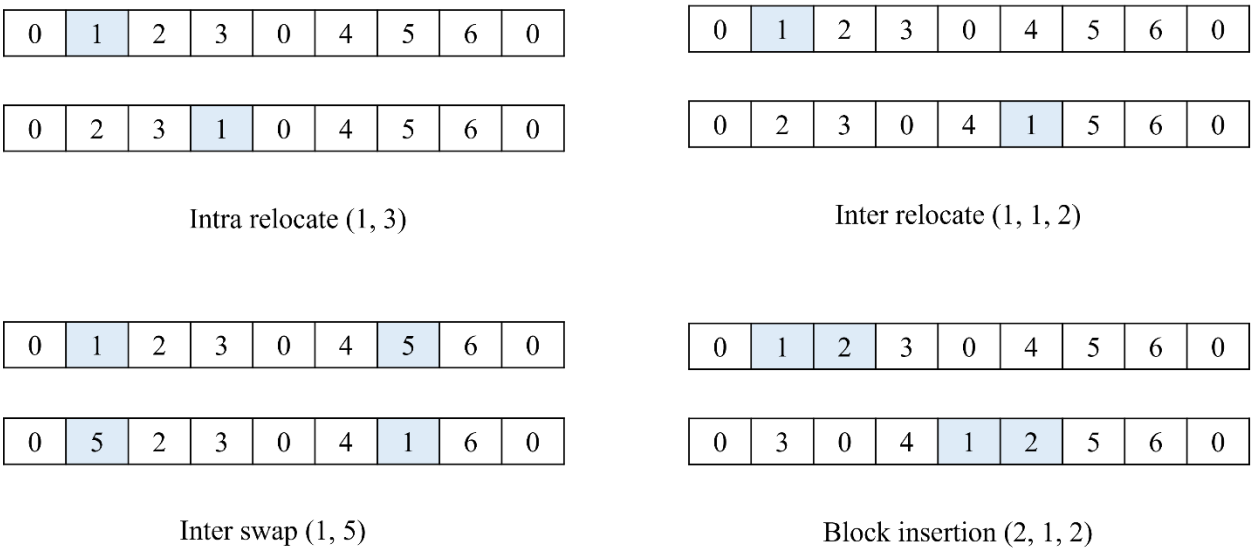

Figure 9: numerical examples for local search moves

We sequentially apply all local search operators, and each operator considers all potential changes by iterating over each route and customer within that route (Islam et al., 2021). For example, the explanation provided refers to the inter-relocate local search function:

The function starts with the current solution and creates a set of routes based on that solution. Each route represents a vehicle's tour to serve a set of customers. The algorithm then calculates the total demand for each route. Then, the algorithm iterates over each route and each customer in that route

(excluding the depot). For each customer, the algorithm evaluates the updated fitness of that route by assuming the removal of the customer from it. In order to reduce computation time, a new fitness function is defined to calculate the total time for a vehicle's tour (one route) and the penalty for violation of time window constraints.

The algorithm then searches for the best route and position to relocate the customer. It considers all other routes (except the current route) and evaluates the fitness improvement that would result from relocating the customer to each possible position in those routes. The best improvement, best route, and best position are tracked.

Once the best route and position are determined, the algorithm updates the routes, demand, and fitness values accordingly if there is any improvement in this relocation. The customer is removed from the current route, and it is inserted into the best route at the best position. The demand and fitness values of the involved routes are updated based on the relocation.

This process is repeated iteratively for each customer in each route, resulting in a sequence of inter-route relocations. The algorithm aims to find relocations that lead to an overall improvement in the solution's fitness. In the end, the fitness of the final solution is calculated, and the updated solution and fitness value are returned.

6.2.6 Large Neighborhood Search (LNS)

In LNS, the algorithm starts with a complete solution and gradually destroys and reconstructs parts of the solution to explore the solution space. Five removal operators are used in large neighborhood search to destroy the solution and select the desired number of customers to be removed (a) based on Equation (22), where If the algorithm does not discover any better solution during an iteration,

the $counter_{no_{improvement}}$ is incremented by 1. The process of removal is carried out in accordance with the research conducted by Pan et al., 2021b.

$$a = \min\left\{a_{max}, a_{min} + a_{rate} * counter_{no_{improvement}}\right\} * num\_cusotmer \qquad (22)$$

The first operator, called Worst Removal (Ropke & Pisinger, 2006), aims to identify and remove customers from a solution that are considered to be misplaced or have a negative impact on the overall fitness of the solution. The objective is to prepare these customers for potential relocation during the reinsertion stage. The removal is based on the customers' insertion cost, which is calculated as the difference in solution cost before and after removing a customer. The customers are then sorted in descending order based on their insertion costs in the ranked list. To prevent the algorithm from repeatedly selecting the same customers for removal, and keeping the selection process diverse, randomization is used as proposed by François et al., 2019. A random value between 0 and 1 is used in the formula mentioned in Algorithm 3 to calculate the index of the customer to be chosen for removal from the ranked list. This process is repeated until the desired number of customers to be removed is selected.

Algorithm 3: Worst removal algorithm:

---

1. Create a ranked list of customers (C) based on insertion cost.
2. While the number of customers selected for removal is smaller than a, do:
    a. Generate a random value v from the interval [0, 1).
    b. Calculate the index i of the customer to be selected as $v^{rf} * |C|$, where rf is a randomization factor parameter.
    c. Remove the customer at index i from the ranked list (C).
    d. Add the selected customer to the list of customers to be removed.
3. End while.

Advanced Shaw Removal selects customers based on their closeness measure and their neighbors (Shaw, 1998). Equation (23) presents the closeness calculation formula, which takes into account various factors such as average travel time during the feasible time window $\bar{t}_{ij}$, waiting time penalties γ_{wt} associated with waiting time when the vehicle departs from customer i at the latest allowable time ($l_i + s_i$), and time window violation penalties γ_{tw} , applied even if the vehicle departs from customer i at the earliest possible time ($e_i + s_i$) but arrived late to customer j. By considering these factors, the operator identifies customers that are closely related or have similar characteristics. Removing customers with dissimilarity may not yield significant improvements since the reinsertion of them may be restricted to their original positions or unfavorable positions (Pan et al., 2021b; Ropke & Pisinger, 2006). The selected customers based on closeness are added to L1 and their neighbors are added to L2. Randomization is used to select the number of customers and their neighbors according to Algorithm 4.

$$CL_{ij} = \bar{t}_{ij} + \gamma_{wt} * \max\{0, e_j - l_i - s_i - TT(i, j, l_i + s_i)\} + \gamma_{tw} * \max\{0, e_i + s_i - TT(i, j, e_i + s_i) - l_j\} \quad (23)$$

Algorithm 4: Advanced Shaw Removal

---

1. Initialize empty lists L1 and L2.
2. While the number of unique customers in L1 and L2 is less than a, do:
    a. If either L1 is empty or there are no unprocessed customers in L1:
        i. Select a random customer from the list of customers.
        ii. Determine the neighbor of the customer (neighbor) based on its position in the list.
        iii. Append customer to L1 as unprocessed.
        iv. Append neighbor to L2 as unprocessed.
    b. End If
    c. Select the first unprocessed customer (uc) in L1.
    d. Select 1 or 2 customers (sc_1, sc_2) based on the highest closeness measure to uc.
    e. Mark the selected customer (uc) as processed in L1.

f. Append customers selected in step c (sc_1, sc_2) as unprocessed to L1 if it is not already present in L1 or L2.
g. Determine the neighbor of the selected customer in step c and append them as unprocessed to L2 if it is not already present in L1 or L2.

3. End While
4. Select the first a customers from combined L1 and L2 to be removed.

In the Route Based Shaw Removal operator, we evaluate the effectiveness of each route by considering the total cost of the route divided by the total demands served. This metric helps us measure how well a route performs in terms of cost per unit of demand. Then, we normalize the effectiveness values to create a probability distribution to select a route. The customers in this route are used as initial L1 and from here, we follow the same steps as in the Advanced Shaw Removal algorithm, using the closeness measure and two removal lists to remove a customers.

Mixed Removal is a combination of advanced Shaw removal and route-based Shaw removal, in which the L1 is initialized using a selected route based on the route-based Shaw removal method and a customer from another random route.

Random Removal selects a customers randomly. This means that each customer in the solution has an equal chance of being selected for removal.

The reinsertion process of LNS is based on k-regret reinsertion which is explained in creating the initial population.

6.2.7 Large Neighborhood Search as Local Search

This function as an intensification function aims to find better solutions. After the removal and reinsertion process of LNS, it compares the fitness of a newly generated solution with the fitness of the current solution. It only accepts better solutions with equal or lower fitness to be replaced

with the current solution, resulting in an improvement. This step focuses on exploiting the search space by consistently selecting better solutions.

6.2.8 Large Neighborhood Search as Mutation

This section aims to introduce diversification in the optimization process to avoid being stuck in local optima. After removal and reinsertion of a customers, if the new solution has a higher fitness value than the current solution (indicating a lack of improvement), there is a possibility of accepting the new solution based on a probability threshold. This result in introducing randomness into the process and helps to explore different areas of the solution space. Consequently, it promotes diversification by actively seeking out different areas and increasing the chances of finding globally optimal or near-optimal solutions.

6.2.9 Discarding

The "discarding" function is responsible for implementing a diversification strategy in the population of individuals. It addresses the scenario where the fitness values of individuals in the population are very close to each other, indicating a lack of diversity. In such cases, the function removes one of the close individuals in terms of fitness from the population and replaces it with a new individual generated through the same process as creating the initial population.

By integrating intensification techniques, our metaheuristic approach aims to strike a balance between exploiting local improvements and exploring diverse solutions, leading to improved solution quality and effectiveness in solving the optimization problem at hand.

6.2.10 Best-q improvement

In this process, best_q represents a list that stores the q best solutions found so far in the optimization process. The variable q determines the number of best solutions to keep track of.

Throughout the iterations and after local search functions which create better solutions, the best_q list is updated to include the best solutions found so far (S. Zhang et al., 2019). The goal is to improve these solutions further using local search and Large Neighborhood Search (LNS) techniques, which provide both intensification and diversification. The detailed steps of best q improvement are shown in Figure 6.

Chapter 7

Numerical Analysis for Time Dependent VRP

7.1 Data

Solomon's dataset for the Time-Dependent Vehicle Routing Problem with Time Windows (TDVRPTW) is frequently applied as a benchmark for evaluating the performance of routing and scheduling algorithms. Dataset instances are divided into three groups of geographic customer repartitions (clustered [C], random [R], and random clustered [RC]). The dataset contains the following columns: Customer/Depot/Vehicle ID, X-Coordinate, Y-Coordinate, Demand, Ready Time, Due Date, and Service Time, which are shown in Table 4.

Table 4: 10 first customers of Solomon dataset (C101 instances)

C101 (Vehicle_number:25, Vehicle_capacity:200)						
Cust No	X_Coordinate	Y_Coordinate	Demand	Ready_Time	Due_Time	Service_Time
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90

7.2 Parameter setting

These are some parameters we should consider when setting up our metaheuristic algorithms for TDVRPTW. The following parameters are tuned using different experiment on several instances of Solomon dataset and shown in Table 5.

Table 5: detailed parameter setting for TDVRPTW

Section	parameter	value	Section	parameter	value
overall	$population\_size$	25	LNS	$ALNS\_LS\_max$	5
	$iteration$	100		$ALNS\_m\_max$	1
	$penalty\_capacity$	10000		$ALNS\_m\_prob$	0.8
	$penalty\ TW$	150000		a_{min}	0.12
best_q	q	3		a_{max}	0.6
dicarding	$dicarding\_gap$	100		a_{rate}	0.05
initial population	$prob\_regret$	0.2		rf	4
genetic	$cross\_ratio$	0.7		γ_{wt}	0.5
	$mutation\_prob$	0.3,0.4		γ_{tw}	0.85
	$selection\_k$	5	LS	LS_max	1000

7.3 Result and discussion

The code was executed on a computer equipped with an 11th Gen Intel(R) Core(TM) i5-11300H processor running at 2.61 GHz, 12 GB of RAM. The operating system used was Windows 11, and the code was implemented in Python 3.9.13.

Our algorithm is compared with optimal solution based on total time as objective function. The results are shown for best solution after 1 time running for 25, 50, and 100 customers in Table 6, Table 6, and Table 7 respectively. In some cases, our algorithm successfully reached the optimal solution, demonstrating its effectiveness. However, it is important to note that for the majority of cases, our algorithm achieved solutions that were near-optimal. The average gap between our solution and the optimal solution for instances with 25, 50, and 100 customers is found to be 1.44%, 3.81%, and 5.68%, respectively. These results indicate that our algorithm consistently achieves solutions that are very close to the optimal solution, with only a small percentage difference on average. This demonstrates the high level of accuracy and effectiveness of our algorithm in tackling the given problem instances.

Table 6: comparison of best fitness of our method and optimal solution_25 customers

instance	optimal	best	gap	instance	optimal	best	gap
C101	24713.63	25093.06	1.54%	C201	25616.94	25698.84	0.32%
C102	24535.70	24672.61	0.56%	C202	24763.79	24763.79	0.00%
C103	24398.88	24522.51	0.51%	C203	24516.91	24549.76	0.13%
C104	24251.69	24319.26	0.28%	C204	24155.16	24225.96	0.29%
C105	24701.96	24932.47	0.93%	C205	25093.61	25206.41	0.45%
C106	24713.63	25111.13	1.61%	C206	24963.49	25050.94	0.35%
C107	24529.24	24549.85	0.08%	C207	24906.52	24960.66	0.22%
C108	24422.83	24462.83	0.16%	C208	24778.02	24823.42	0.18%
C109	24341.81	24414.47	0.30%				
instance	optimal	best	gap	instance	optimal	best	gap
R101	9084.66	9354.89	2.97%	R201	7933.63	8225.64	3.68%
R101	7502.89	7535.82	0.44%	R202	7121.79	7216.96	1.34%
R102	6558.70	6759.56	3.06%	R203	6696.01	6909.09	3.18%
R103	5943.64	5995.48	0.87%	R204	5801.33	5808.87	0.13%
R104	7217.56	7283.29	0.91%	R205	6444.65	6614.14	2.63%
R105	6542.48	6694.49	2.32%	R206	6126.01	6341.26	3.51%
R106	5995.62	6220.19	3.75%	R207	5708.00	5835.10	2.23%
R107	5685.13	5819.5	2.36%	R208	5347.48	5427.00	1.49%
R108	6139.76	6349.58	3.42%	R209	5560.28	5740.16	3.24%
R109	5944.82	6008.91	1.08%	R210	6507.65	6559.15	0.79%

R110	6060.44	6185.54	2.06%	R211	4921.15	4993.45	1.47%
R111	5634.71	5688.07	0.95%				
R112	9084.66	9354.89	2.97%				
instance	optimal	best	gap	instance	optimal	best	gap
RC101	7046.40	7217.53	2.43%	RC201	8388.23	8498.86	1.32%
RC102	6194.98	6335.15	2.26%	RC202	7437.67	7437.67	0.00%
RC103	5454.11	5605.71	2.78%	RC203	6196.00	6196.00	0.00%
RC104	5208.80	5309.76	1.94%	RC204	5709.00	5745.70	0.64%
RC105	6838.89	6855.01	0.24%	RC205	7632.31	7897.24	3.47%
RC106	5625.96	5695.6	1.24%	RC206	6610.71	6655.88	0.68%
RC107	5034.48	5095.69	1.22%	RC207	5359.45	5462.20	1.92%
RC108	4923.51	4974.8	1.04%	RC208	4355.80	4505.80	3.44%

Table 7: comparison of best fitness of our method and optimal solution_50 customers

instance	optimal	best	gap	instance	optimal	best	gap
C101	49251.04	51057.30	3.67%	C201	49698.45	50410.08	1.43%
C102	48553.41	49200.00	1.33%	C202	48745.57	49388.78	1.32%
C103	48258.21	48902.94	1.34%	C203	48496.92	49296.32	1.65%
C104	47988.22	48521.24	1.11%	C204	47,911.45	48557.79	1.35%
C105	48818.40	49974.85	2.37%	C205	49150.57	49682.15	1.08%
C106	48687.39	49856.04	2.40%	C206	49041.43	49505.76	0.95%
C107	48488.94	49349.99	1.78%	C207	49070.27	49682.57	1.25%
C108	48227.15	48600.63	0.77%	C208	48813.23	49112.10	0.61%
C109	48181.30	48501.33	0.66%				
instance	optimal	best	gap	instance	optimal	best	gap
R101	16115.18	16540.75	2.64%	R201	14059.17	14844.79	5.59%
R101	13454.48	13904.3	3.34%	R202	12586.95	13143.59	4.42%
R102	11704.47	12342.95	5.46%	R203	11289.47	11895.36	5.37%
R103	10300.72	10792.3	4.77%	R204	9,550.79	10054.7	5.28%
R104	12828.75	13391.05	4.38%	R205	11785.81	12344.75	4.74%
R105	11806.53	12162.47	3.01%	R206	10864.98	11363.98	4.59%
R106	11047.45	11400.34	3.19%	R207	10274.73	10524.52	2.43%
R107	10026.79	10345.48	3.18%	R208	9,140.49	9431.06	3.18%
R108	11322.23	11812.98	4.33%	R209	10126.69	10836.34	7.01%
R109	10772.25	11251.36	4.45%	R210	11112.05	12052.85	8.47%
R110	10826.73	11163.76	3.11%	R211	8928.65	9161.45	2.61%
R111	10193.11	10427.05	2.30%				
R112	16115.18	16540.75	2.64%				
instance	optimal	best	gap	instance	optimal	best	gap

RC101	14007.50	15143.55	8.11%	RC201	14132.29	15776.22	2.41%
RC102	12451.40	13183.54	5.88%	RC202	13284.98	14145.41	6.48%
RC103	11338.09	11887.05	4.84%	RC203	11,882.96	12703.76	6.91%
RC104	9851.30	10382.11	5.39%	RC204	9,585.57	10052.14	4.87%
RC105	13036.64	13775.53	5.67%	RC205	13984.05	15002.77	7.28%
RC106	11846.66	12497.12	5.49%	RC206	12044.50	12778.89	6.10%
RC107	10623.55	11160.8	5.06%	RC207	9734.90	10699.23	9.91%
RC108	10289.56	10453.07	1.59%	RC208	8369.90	8747.15	4.51%

Table 8: comparison of best fitness of our method and optimal solution_100 customers

instance	optimal	best	gap	instance	optimal	best	gap
C101	104253.20	6352.09	6.49%	C201	98480.06	2364.24	2.46%
C102	101532.44	3869.84	3.96%	C202	99041.57	3116.72	3.25%
C103	100230.39	2857.04	2.93%	C203	99102.15	3388.89	3.54%
C104	98571.81	1459.34	1.50%	C204	99108.25	3487.50	3.65%
C105	102494.80	4665.55	4.77%	C205	97689.30	1925.63	2.01%
C106	101810.08	4117.57	4.21%	C206	98751.81	3135.64	3.28%
C107	99863.57	2394.49	2.46%	C207	97763.77	2150.66	2.25%
C108	98732.06	1518.28	1.56%	C208	97342.20	1790.77	1.87%
C109	98525.06	1520.46	1.57%				
instance	optimal	best	gap	instance	optimal	best	gap
R101	27019.58	154.60	0.58%	R201	25219.88	2774.60	12.36%
R101	24744.71	1312.98	5.60%	R202	23659.40	2947.37	14.23%
R102	22324.52	1600.16	7.72%	R203	20496.13	1872.25	10.05%
R103	19645.27	1427.49	7.84%	R204	17450.80	915.61	5.54%
R104	23658.56	1774.08	8.11%	R205	19988.67	603.81	3.11%
R105	21986.11	1577.93	7.73%	R206	19369.23	1034.83	5.64%
R106	20461.20	1664.01	8.85%	R207	18109.23	819.11	4.74%
R107	18946.68	1358.48	7.72%	R208	16789.87	851.30	5.34%
R108	20278.68	968.37	5.01%	R209	19001.76	1870.92	10.92%
R109	19296.06	558.00	2.98%	R210	20358.67	1793.94	9.66%
R110	19497.33	1041.54	5.64%	R211	16044.20	637.90	4.14%
R111	18411.08	586.80	3.29%				
R112	27019.58	154.60	0.58%				
instance	optimal	best	gap	instance	optimal	best	gap
RC101	26623.07	1832.55	7.39%	RC201	27959.51	3513.07	14.37%
RC102	24856.11	2251.24	9.96%	RC202	25401.26	3042.61	13.61%
RC103	22083.85	1561.29	7.61%	RC203	20995.45	1373.89	7.00%
RC104	20130.95	770.28	3.98%	RC204	18297.28	824.90	4.72%

RC105	25082.59	1672.10	7.14%	RC205	24109.64	465.72	1.97%
RC106	22810.16	1454.57	6.81%	RC206	21700.65	993.63	4.80%
RC107	21510.48	1453.36	7.25%	RC207	18367.89	506.50	2.84%
RC108	20158.27	787.98	4.07%	RC208	16956.64	992.04	6.21%

To check the effect of hybridizing different methods, we solved the problem in different settings, including the complete version, without LNS as mutation, without local search, and without best_q improvement. Since during best_q improvement, we used the local search function, to evaluate the performance of the algorithm without any local search, another setting is defined. This setting solves the problem without both local search and any local search functions during the best_q improvement process.

7.4 Sensitivity Analysis

Table 9, Table 10, and Table 11 shows the performance of the algorithm in 5 different scenarios for 6 instances with 25, 50, and 100 customers. We tried to fix the CPU time to make a fair comparison, so the number of iterations varies in different settings. The metric for evaluating the performance of each setting is the average fitness (total duration) of 6 instances. When a genetic algorithm is combined with a local search scheme, the percentage improvement in average fitness is calculated by comparing the average fitness obtained by the genetic algorithm without applying the local search scheme. This calculation measures the relative enhancement achieved by incorporating the local search into the genetic algorithm.

7.4.1 Effect of local search

When comparing the solution quality without local search (No LS) to the solution quality with the addition of local search, the improvement is relatively small. Instances with 25 and 50 customers

show a mere 0.2% and 0.4% enhancement respectively, while instances with 100 customers exhibit a 0.4% decrease in solution quality.

This limited improvement can be attributed to the fact that both scenarios still utilize the best_q improvement process, which incorporates the local search operator for the three best solutions found so far. As a result, even without explicitly applying local search, the best_q improvement process already generates solutions that are very close to the complete version.

The inclusion of local search as a separate operator in addition to the best_q improvement process may not yield significant additional improvements. This suggests that the impact of local search alone, when used alongside the best_q improvement process, is limited in terms of further enhancing solution quality.

7.4.2 Effect of best_q improvement

The inclusion of the best_q improvement process in the genetic algorithm has a mixed impact on the solution quality when compared to the complete version of the algorithm. In instances with 25 and 50 customers, the improvement is substantial, resulting in an average fitness enhancement of approximately 0.1% for both cases. However, for instances with 100 customers, the addition of the best_q improvement process leads to a slight decrease in solution quality, with an average fitness reduction of approximately 0.5%.

The discrepancy in results can be attributed to the computational time required by the best_q improvement process. While it improves the quality of solutions, it also increases the overall CPU time required to solve the problem. In instances with a larger number of customers, computational time becomes a crucial factor. Therefore, in a fixed CPU time setting, the best_q improvement process may not have sufficient time to find better solutions. This limitation leads to negligible

improvements or even worsens the solution quality for large instances within the given time constraints.

7.4.3 Effect of local search and best_q improvement (intensification)

Since local search operators are used in both local search and best_q improvement process, to assess the effect of hybridizing local search with the genetic algorithm (GA), it is necessary to remove both the local search operator and the best_q improvement process and compare the results.

According to the results, the combination of local search and the best_q improvement process has a profound effect on solution quality and fitness function enhancements. The results show that when the genetic algorithm is combined with both local search and best_q improvement, the average fitness is significantly improved. Instances with 25, 50, and 100 customers demonstrate fitness enhancements of approximately 27%, 30%, and 36% respectively.

It is clear that the impact of local search and the improvement in the best_q metric is associated with intensification, which can lead to significant enhancements in the fitness function. The combination of these techniques leverages the intensification effect of local search and the quality-driven exploration of the genetic algorithm, leading to substantial improvements in the overall fitness and solution quality.

7.4.4 Effect of LNS as mutation

However, incorporating Large Neighborhood Search (LNS) as a mutation operator does not necessarily enhance the quality of solutions. This is primarily due to the constrained CPU time, which prevents the algorithm from getting stuck in local optima. The significance and

effectiveness of using LNS as a mutation function are observed when the solution requires more diversification to avoid being trapped in local optima.

Table 9: The effect of hybridization for 25 customers

Degree of hybridization	average fitness	# of iterations	CPU Time	Improvement %
complete	13437.19	31	607.77	27.7
No best_q improvement	13461.64	81	602.66	27.6
No LS	13474.70	49	606.64	27.5
No LNS_mutation	13442.63	32	606.10	27.7
No LS and best_q LS	18596.85	1081	600.29	0.0

Table 10:The effect of hybridization for 50 customers

Degree of hybridization	average fitness	# of iterations	CPU Time	Improvement %
complete	26791.19	7	656.87	30.1
No best_q improvement	26823.21	17	615.02	30.0
No LS	26931.3	10	622.02	29.7
No LNS_mutation	26805.66	7	645.35	30.1
No LS and best_q LS	38328.02	340	601.13	0.0

Table 11: The effect of hybridization for 100 customers

Degree of hybridization	average fitness	# of iterations	CPU Time	Improvement %
complete	58218.62	3	1394.89	36.5
No best_q improvement	57761.578	7	1288.13	37.0

No LS	58600.65	5	1290.76	36.1
No LNS_mutation	57899.06	3	1370.13	36.8
No LS and best_q LS	91674.32	90	1206.54	0

Chapter 8

Proposed TDVRPTW model considering real travel time

8.1 Model architecture

After training deep learning algorithms for travel time estimation, the learned weights of the model are saved in a text file. These weights can be later loaded and used during the testing phase or for other applications. For applying the estimation in TDVRPTW, the input of the TTE module is a route and starting time from the origin node. The route is generated using Google Direction API (*Getting Directions through the Directions API | Google Developers*, n.d.) based on desired origin and destination of the TDVRPTW problem. Figure 10 illustrated the architecture of combining the TTE module and TDVRPTW, highlighting the difference between the common process used in VRPTW, and TDVRPTW.

In VRPTW, there is a fixed travel time matrix for different combinations of nodes. In this context, many research studies employ a distance matrix as a representation of travel time. This distance matrix contains the distances between different pairs of nodes in the problem. By using the distance matrix, researchers can estimate the travel time based on the assumption that travel time is directly proportional to distance. The distance matrix provides a convenient way to calculate travel times in VRPTW, as it allows for efficient computation of the distances between nodes without the need for real-time data or complex modeling of factors such as traffic congestion. However, it is important to note that using a distance matrix as a proxy for travel time assumes a constant travel speed and does not account for dynamic factors that may affect actual travel times.

In the TDVRPTW, travel times are not constant throughout the day as they are in the traditional VRPTW. Instead, travel times vary depending on the time of day. To account for this time dependency, a function is used to map the travel time based on the starting time of a route.

In our proposed method, as the TTE module needs a route for capturing spatial correlations, a route is generated based on the origin and destination nodes using Google API. Before running the metaheuristic algorithms to solve TDVRPTW, all possible combination of two nodes is considered. Then, for every two different nodes, the Google API is used to generate a route, which is a list of GPSs between the origin and destination. Figure 11 indicates how Google API converts VRP nodes locations to a route matrix. The route generated by the Google API provides a spatial representation of the path between the nodes, incorporating geographical details and road network information. This route serves as input for the TTE module, which applies the saved weights of the trained model to estimate the travel time for each road segment and the entire route. In other words, according to the input route and departing time, the TTE module utilizes the saved weights of the trained model to estimate the travel time.

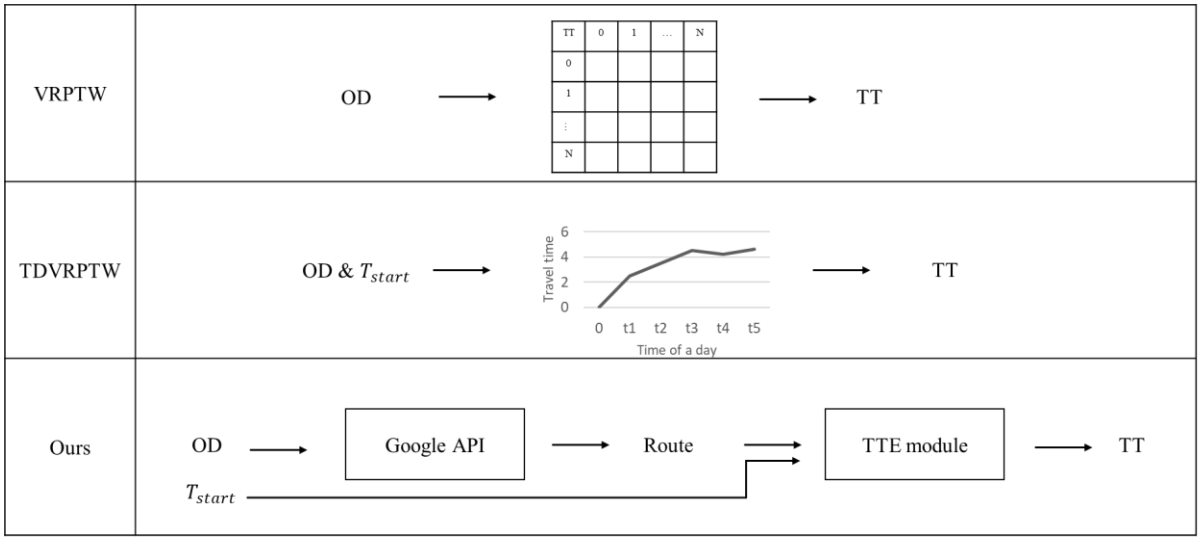

Figure 10: combining TTE and TDVRPTW, compared with regular VRPTW, and TDVRPTW

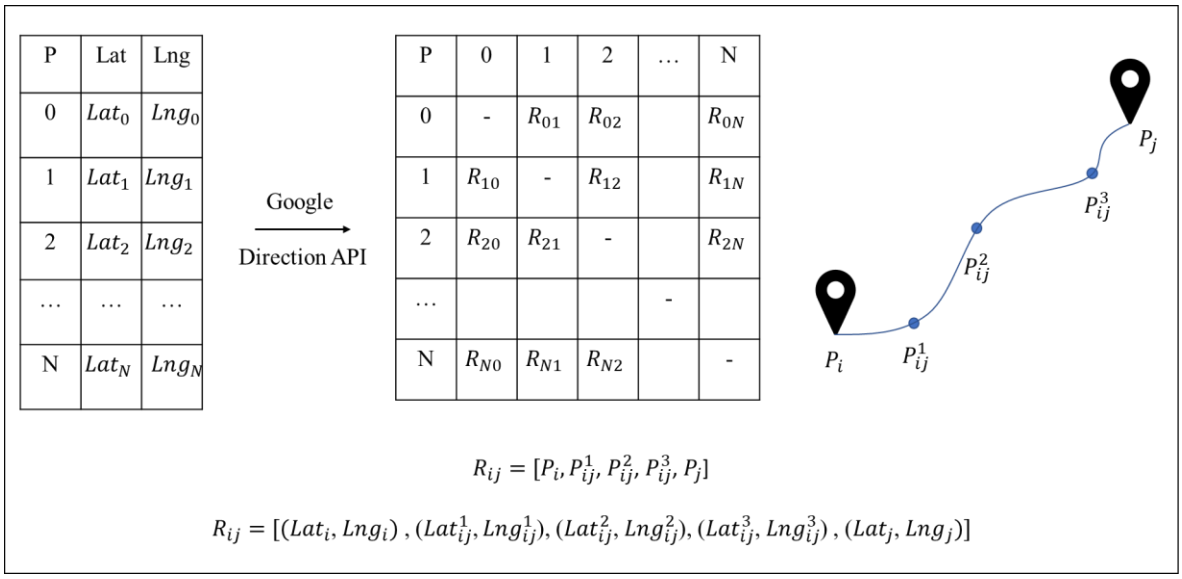

Figure 11: overview of the use of Google API for generating routes

8.2 Data

To check the possibility of integrating VRP and TTE module, 5 random sample datasets with 10 customer node and one depot is generated based on the latitude and longitude of Chengdu city and in the same format as the Solomon dataset. It is important to highlight that when using a deep learning model and weights specifically trained on Chengdu city, the input route data for interfacing and utilizing the TTE module should have a similar distribution. To ensure compatibility, random values were generated within the GPS area of Chengdu city to create customer locations that align with the distribution used in training the model. This allows for consistent and accurate predictions when utilizing the TTE module with the generated customer locations in Chengdu City. Table 12, illustrates the location, demand, time window, and service time for each node based on one of the generated sample data. Figure 12 also shows the locations of 10 customers and one depot on the Chengdu city map.

Table 12: sample TDVRPTW dataset in Chengdu city (based on Solomon dataset format)

cust No	lat	lng	Demand	Ready time	Due date	Service time
0	30.66222	103.9689	0	0	1350	0
1	30.49057	104.0428	30	75	662	90
2	30.64639	103.6759	30	845	929	90
3	30.60393	104.1847	25	893	965	90
4	30.76584	103.8906	35	155	575	90
5	30.63099	103.7576	5	906	1106	90
6	30.56116	104.1196	10	630	835	90
7	30.75416	104.2211	25	145	310	90
8	30.46597	103.9542	15	0	979	90
9	30.57917	103.9728	20	734	870	90
10	30.5222	103.8307	20	222	461	90

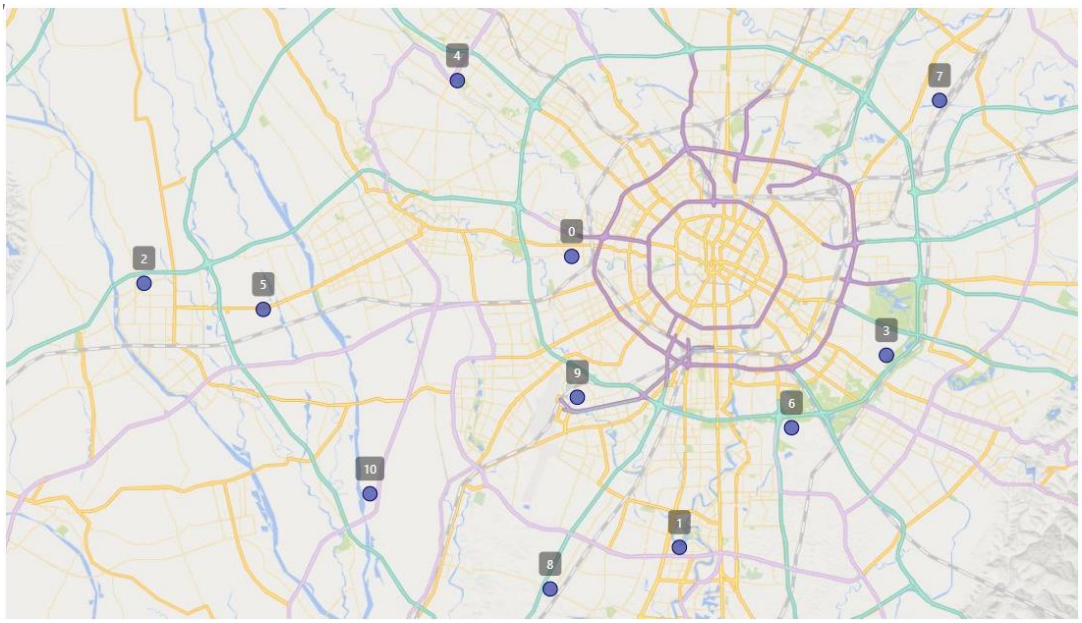

Figure 12: locations of generated customers on Chengdu map

8.3 Result and discussion

As there are currently no existing papers that apply a combination of deep learning for travel time estimation and a metaheuristic algorithm for solving TDVRPTW, our result provides a basis for comparison with future work in the field. To evaluate the possibility of our approach, we conducted experiments using five different datasets, each representing a specific generated dataset in the area of Chengdu city with 10 customers. Since estimating travel time using deep learning has more computations, it takes more time to calculate travel time even using saved weights of trained models. Therefore, to solve TDVRPTW, we applied two settings, first only the genetic algorithm, and second the combination of the genetic algorithm and one of the local search (only inter-relocate) to solve TDVRPTW. The summarized results of these two settings are as follows:

Table 13:sample solution for TDVRPTW using TTE module_10 customers_genetic algorithm

instance	route	departure time	fitness
1	0, 8, 4, 10, 9, 2, 5, 11	809.24	18780.99
	0, 7, 1, 6, 3, 11	795.76	
2	0, 5, 2,11	347.23	16593.12
	0, 1, 7, 3, 6, 9, 11	740.25	
	0, 10, 8, 4, 11	540.90	
3	0, 8, 9, 2, 5, 10, 11	621.09	19201.87
	0, 4, 3, 1, 6, 7, 11	288.43	
4	0, 10, 1, 11	565.38	10532.44
	0, 3, 5, 9, 11	450.29	
	0, 8, 2, 7, 4, 6, 11	301.54	
5	0, 3, 9, 6, 2, 11	735.69	15472.03
	0, 1, 7, 8, 5, 4, 10, 11	491.00	

Table 14: sample solution for TDVRPTW using TTE module_10 customers_genetic algorithm combined with inter relocate LS.

instance	route	departure time	fitness
1	0, 8, 2, 1, 9, 3, 5, 11	791.32	15581.76
	0, 7, 10, 6, 4, 11	705.60	
2	0, 10, 2, 4, 11	407.33	13403.54
	0, 8, 7, 5, 6, 3, 11	760.02	
	0, 1, 9, 11	408.90	
3	0, 2, 4, 6, 10, 11	560.11	17119.65
	0, 8, 3, 1, 5, 9, 7, 11	198.75	
4	0, 9, 6, 1, 11	555.94	9065.03
	0, 3, 4, 11	422.78	
	0, 7, 2, 8, 5, 10, 11	353.04	
5	0, 7, 9, 6, 5, 2, 11	625.65	13787.35
	0, 10, 8, 3, 4, 1, 11	405.77	

Hybridizing a genetic algorithm with local search can indeed lead to improved solutions, but it can also increase the computational time required to solve the problem. This is mainly due to the additional computation needed for the local search phase, especially when calculating travel times using deep learning weights, which can be more time-consuming compared to simpler discrete speed function methods. The local search phase involves exploring all possible movements to improve the solution. However, each movement requires calculating travel times using deep learning weights, which can accumulate and result in significantly longer computation times. For instance, for a problem with just 10 customers, the time required to find a solution might be as long as half a day. One approach to address this issue is to employ a reverse method, focusing on optimizing the deep learning model to find an efficient speed function. Moreover, by concentrating on relevant variables in the deep learning model, it might be possible to reduce the number of

weights, leading to faster computation times for the TDVRPTW using the deep learning-based TTE module.

Chapter 9

Conclusion

In conclusion, this thesis has successfully addressed the challenges posed by realistic and dynamic transportation systems by combining accurate travel time estimation using deep learning algorithms and the application of metaheuristic algorithms, including genetic algorithms, local search, and large neighborhood search, to solve the time-dependent vehicle routing problem with time windows (TDVRPTW).

The novelty of this research lies in three significant contributions. Firstly, the integration of deep learning-based travel time estimation (TTE) with the time-dependent vehicle routing problem with time windows (TDVRPTW), which had not been done before, introduces a novel approach that addresses the limitations of traditional routing optimization methods. By considering real traffic conditions and speed flow, this combination enables a more realistic and accurate representation of travel times, leading to improved routing optimization. Secondly, the development of a new deep learning-based model utilizing a large real-world GPS dataset has demonstrated superior performance, achieving higher accuracy in travel time estimation compared to existing models. This enables the development of more efficient and effective transportation systems that can adapt to the dynamic nature of travel times. Thirdly, the application of hybrid algorithms, combining genetic algorithms, local search, and large neighborhood search techniques, addresses the intensification and diversification aspects of metaheuristic algorithms. Genetic algorithms, inspired by natural selection, iteratively evolve potential solutions to enhance their quality over generations. The inclusion of local search and large neighborhood search techniques effectively

manages the exploration and exploitation aspects, allowing for efficient exploration of the solution space and refinement of the solutions. Additionally, the focus on the three best solutions found so far and applying local search to them results in the discovery of near-optimal solutions.

To summarize, this research has yielded valuable insights into the integration of traditional problem-solving methods with innovative artificial intelligence and data-driven approaches, leading to enhancing transportation systems and enabling them to adapt to the dynamic nature of travel times. These advancements have wide-ranging implications, including improved efficiency, reduced operational costs, and an overall enhancement in the quality of life within urban areas. As for future research, the integration of additional factors such as weather data could be considered. This would further enhance the accuracy and realism of travel time estimation, allowing for more precise routing optimization in response to changing weather conditions. Moreover, in our current research, we have primarily focused on utilizing a single route generated between each pair of nodes using the Google Direction API to address the combination of TDVRPTW and Travel time estimation. However, in real-world scenarios, multiple routes exist between an origin and destination based on different departure times, leading to varying travel times. Time can be considered as the third dimension alongside origin and destination for generating different routes and selecting the best route in terms of minimum time for future research. Lastly, By optimizing the deep learning model's architecture and reversed method to find a simplified speed function according to the results of deep learning, it might be feasible to find a more efficient speed function, making the hybrid algorithm with local search more time-effective while still delivering improved solutions for the TDVRPTW. By continuously advancing the integration of artificial intelligence, data-driven approaches, and real-time data utilization, researchers can unlock further

potential for optimizing transportation systems and developing sustainable and efficient logistics networks.

.

Reference

Abdulkader, M. M. S., Gajpal, Y., & Elmekkawy, T. Y. (2015). Hybridized ant colony algorithm for the Multi Compartment Vehicle Routing Problem. *Applied Soft Computing Journal*, *37*.

Abdulkader, M. M. S., Gajpal, Y., & ElMekkawy, T. Y. (2018). Vehicle routing problem in omni-channel retailing distribution systems. *International Journal of Production Economics*, *196*.

Agrawal, A. K., Yadav, S., Gupta, A. A., & Pandey, S. (2022). A genetic algorithm model for optimizing vehicle routing problems with perishable products under time-window and quality requirements. *Decision Analytics Journal*, *5*.

Azi, N., Gendreau, M., & Potvin, J. Y. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Computers and Operations Research*, *41*(1).

Billings, D., & Jiann-Shiou, Y. (2006). Application of the ARIMA Models to Urban Roadway Travel Time Prediction-A Case Study. Systems, Man and Cybernetics, 2006. SMC'06. *IEEE International Conference On*.

Cha, Y. J., Mostafavi, A., & Benipal, S. S. (2023). DNoiseNet: Deep learning-based feedback active noise control in various noisy environments. *Engineering Applications of Artificial Intelligence*, *121*.

Chen, X. M., Gong, H. B., & Wang, J. N. (2012). BRT vehicle travel time prediction based on SVM and Kalman filter. *Jiaotong Yunshu Xitong Gongcheng Yu Xinxi/Journal of Transportation Systems Engineering and Information Technology*, *12*(4).

Chen, Y. T., Sun, E. W., Chang, M. F., & Lin, Y. B. (2021). Pragmatic real-time logistics management with traffic IoT infrastructure: Big data predictive analytics of freight travel time for Logistics 4.0. *International Journal of Production Economics*, *238*.

Dabia, S., Ropke, S., Van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science*, *47*(3).

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*(1).

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, *185*(3).

El-Sherbeny, N. A. (2010). Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University - Science*, *22*(3).

Fan, H., Zhang, Y., Tian, P., Lv, Y., & Fan, H. (2021). Time-dependent multi-depot green vehicle routing problem with time windows considering temporal-spatial distance. *Computers and Operations Research*, *129*.

François, V., Arda, Y., & Crama, Y. (2019). Adaptive large neighborhood search for multitrip vehicle routing with time windows. *Transportation Science*, *53*(6), 1706–1730.

Gajpal, Y., & Abad, P. (2009). An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers and Operations Research*, *36*(12), 3215–3223.

Gajpal, Y., Abdulkader, M. M. S., Zhang, S., & Appadoo, S. S. (2017). Optimizing garbage collection vehicle routing problem with alternative fuel-powered vehicles. *Optimization*, *66*(11).

Gajpal, Y., Roy, V., & Sahay, B. S. (2019). Vehicle routing for a mid-day meal delivery distribution system. *Heliyon*, *5*(1).

Gao, P., Hu, J., Zhou, H., & Zhang, Y. (2016). Travel time prediction with immune genetic algorithm and support vector regression. *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, *2016-September*.

Getting directions through the Directions API | Google Developers. (n.d.). Retrieved December 13, 2022, from https://developers.google.com/maps/documentation/directions/get-directions

Gmira, M., Gendreau, M., Lodi, A., & Potvin, J. Y. (2021). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research*, *288*(1).

Hou, D., Fan, H., Ren, X., Tian, P., & Lv, Y. (2021). Time-dependent multi-depot heterogeneous vehicle routing problem considering temporal–spatial distance. *Sustainability (Switzerland)*, *13*(9).

Islam, M. A., & Gajpal, Y. (2021). Optimization of conventional and green vehicles composition under carbon emission cap. *Sustainability (Switzerland)*, *13*(12).

Islam, M. A., Gajpal, Y., & ElMekkawy, T. Y. (2021). Hybrid particle swarm optimization algorithm for solving the clustered vehicle routing problem. *Applied Soft Computing*, *110*.

Jie, K. W., Liu, S. Y., & Sun, X. J. (2022). A hybrid algorithm for time-dependent vehicle routing problem with soft time windows and stochastic factors. *Engineering Applications of Artificial Intelligence*, *109*.

Jin, G., Wang, M., Zhang, J., Sha, H., & Huang, J. (2022). STGNN-TTE: Travel time estimation via spatial–temporal graph neural network. *Future Generation Computer Systems*, *126*.

Keskin, F. D. (2021). Multi-criteria decision making with machine learning for vehicle routing problem. In *Handbook of Research on Decision Sciences and Applications in the Transportation Sector*.

Khanchehzarrin, S., Shahmizad, M., Mahdavi, I., Mahdavi-Amiri, N., & Ghasemi, P. (2021). A model for the time dependent vehicle routing problem with time windows under traffic conditions with intelligent travel times. *RAIRO - Operations Research*, *55*(4).

Konstantakopoulos, G. D., Gayialis, S. P., & Kechagias, E. P. (2022). Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Operational Research*, *22*(3), 2033–2062.

Kwon, J., Coifman, B., & Bickel, P. (2000). Day-to-day travel-time trends and travel-time prediction from loop-detector data. *Transportation Research Record*, *1717*.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*(4).

Li, X., Gao, J., Wang, C., Huang, X., & Nie, Y. (2021). Ride-Sharing Matching under Travel Time Uncertainty through A Data-Driven Robust Optimization Approach. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, 2021-September*.

Liu, C., Kou, G., Zhou, X., Peng, Y., Sheng, H., & Alsaadi, F. E. (2020). Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. *Knowledge-Based Systems*, *188*.

Liu, F., Yang, J., Li, M., & Wang, K. (2022). MCT-TTE: Travel Time Estimation Based on Transformer and Convolution Neural Networks. *Scientific Programming*, *2022*.

Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). Predicting the next location: A recurrent model with spatial and temporal contexts. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*.

Liu, Y., Wang, Y., Yang, X., & Zhang, L. (2018). Short-term travel time prediction by deep learning: A comparison of different LSTM-DNN models. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, *2018-March*.

Luo, Z., Shi, H., Liu, W., & Jin, Y. (2020). HMM-Based Traffic Situation Assessment and Prediction Method. *CICTP 2020: Transportation Evolution Impacting Future Mobility - Selected Papers from the 20th COTA International Conference of Transportation Professionals*.

Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science*, *26*(3).

Mostafavi, A., & Cha, Y. J. (2023). Deep learning-based active noise control on construction sites. *Automation in Construction*, *151*.

Mostafavi, A., & Sadighi, A. (2021). A Novel Online Machine Learning Approach for Real-Time Condition Monitoring of Rotating Machines. *9th RSI International Conference on Robotics and Mechatronics, ICRoM 2021*.

Murni, Kosasih, R., Fahrurozi, A., Handhika, T., Sari, I., & Lestari, D. P. (2020). Travel Time Estimation for Destination in Bali Using kNN-Regression Method with Tensorflow. *IOP Conference Series: Materials Science and Engineering*, *854*(1).

Pan, B., Zhang, Z., & Lim, A. (2021a). A hybrid algorithm for time-dependent vehicle routing problem with time windows. *Computers and Operations Research*, *128*.

Pan, B., Zhang, Z., & Lim, A. (2021b). A hybrid algorithm for time-dependent vehicle routing problem with time windows. *Computers and Operations Research*, *128*.

Park, D., & Rilett, L. R. (1999). Forecasting freeway link travel times with a multilayer feedforward neural network. *Computer-Aided Civil and Infrastructure Engineering*, *14*(5).

Peng, B., Zhang, Y., Gajpal, Y., & Chen, X. (2019). A memetic algorithm for the green vehicle routing problem. *Sustainability (Switzerland)*, *11*(21).

Qiu, J., Du, L., Zhang, D., Su, S., & Tian, Z. (2020). Nei-TTE: Intelligent Traffic Time Estimation Based on Fine-Grained Time Derivation of Road Segments for Smart City. *IEEE Transactions on Industrial Informatics*, *16*(4).

Ran, X., Shan, Z., Fang, Y., & Lin, C. (2019). An LSTM-based method with attention mechanism for travel time prediction. *Sensors (Switzerland)*, *19*(4).

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4).

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1520*.

Sun, H., Liu, H. X., Xiao, H., He, R. R., & Ran, B. (2003). Use of Local Linear Regression Model for Short-term Traffic Forecasting. *Transportation Research Record*, *1836*.

Tan, F., Chai, Z. yi, & Li, Y. lun. (2021). Multi-objective evolutionary algorithm for vehicle routing problem with time window under uncertainty. *Evolutionary Intelligence*.

Tarapata, Z., Kulas, W., & Antkiewicz, R. (2022). Machine learning algorithms for the problem of optimizing the distribution of parcels in time-dependent networks: the case study. *Archives of Transport*, *61*(1), 133–147.

Wang, D., Cao, W., Xu, M., & Li, J. (2016). ETCPS: An effective and scalable traffic condition prediction system. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9643*.

Wang, D., Zhang, J., Cao, W., Li, J., & Zheng, Y. (2018). When will you arrive? Estimating travel time based on deep neural networks. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

White, A., & Schmidt, K. (2005). Systematic literature reviews. *Complementary Therapies in Medicine*, *13*(1), 54–60.

Wu, D., & Wu, C. (2022). Research on the Time-Dependent Split Delivery Green Vehicle Routing Problem for Fresh Agricultural Products with Multiple Time Windows. *Agriculture (Switzerland)*, *12*(6).

Xie, Y., Zhang, Y., & Ye, Z. (2007). Short-term traffic volume forecasting using Kalman filter with discrete wavelet decomposition. *Computer-Aided Civil and Infrastructure Engineering*, *22*(5).

Xu, F., Lin, Y., Huang, J., Wu, D., Shi, H., Song, J., & Li, Y. (2016). Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach. *IEEE Transactions on Services Computing*, *9*(5), 796–805.

Xu, T. dong, Hao, Y., Peng, Z. ren, & Sun, L. jun. (2012). Real-time travel time predictor for route guidance consistent with driver behavior. *Canadian Journal of Civil Engineering*, *39*(10).

Xu, Y., Kong, Q. J., Klette, R., & Liu, Y. (2014). Accurate and interpretable bayesian MARS for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *15*(6).

Xu, Y., Yin, F., Xu, W., Lin, J., & Cui, S. (2019). Wireless Traffic Prediction with Scalable Gaussian Process: Framework, Algorithms, and Verification. *IEEE Journal on Selected Areas in Communications*, *37*(6).

Yao, H., Tang, X., Wei, H., Zheng, G., & Li, Z. (2019). Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference,*

IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019.

Yeh, W. C., & Tan, S. Y. (2021). Simplified swarm optimization for the heterogeneous fleet vehicle routing problem with time-varying continuous speed function. *Electronics (Switzerland)*, *10*(15).

Zhang, J., & Sun, J. (2011). Prediction of urban expressway travel time based on SVM. *Jiaotong Yunshu Xitong Gongcheng Yu Xinxi/Journal of Transportation Systems Engineering and Information Technology*, *11*(2).

Zhang, S., Gajpal, Y., Appadoo, S. S., & Abdulkader, M. M. S. (2018). Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International Journal of Production Economics*, *203*.

Zhang, S., Zhang, W., Gajpal, Y., & Appadoo, S. S. (2019). *Ant Colony Algorithm for Routing Alternate Fuel Vehicles in Multi-depot Vehicle Routing Problem*.

Zhao, J., & Sun, S. (2016). High-Order Gaussian Process Dynamical Models for Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems*, *17*(7).

Zheng, Y. (2015). Trajectory data mining: An overview. In *ACM Transactions on Intelligent Systems and Technology* (Vol. 6, Issue 3).

Zheng, Y., Li, Y., Own, C. M., Meng, Z., & Gao, M. (2018). Real-time predication and navigation on traffic congestion model with equilibrium Markov chain. *International Journal of Distributed Sensor Networks*, *14*(4).

Zhong, Y., & Pan, X. (2007). A hybrid optimization solution to VRPTW based on simulated annealing. *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2007*.