Developing Environmentally Friendly Solutions for On-Demand Food Delivery Service

December 2022

A Research Report from the National Center for Sustainable Transportation

Peng Hao, University of California, Riverside
Haishan Liu, University of California, Riverside
Yejia Liao, University of California, Riverside
Kanok Boriboonsomsin, University of California, Riverside
Matthew J. Barth, University of California, Riverside

National Center for Sustainable Transportation

UCR | College of Engineering- Center for Environmental Research & Technology

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. NCST-UCR-RR-22-43	**2. Government Accession No.** N/A	**3. Recipient's Catalog No.** N/A
4. Title and Subtitle Developing Environmentally Friendly Solutions for On-Demand Food Delivery Service		**5. Report Date** December 2022 **6. Performing Organization Code** N/A
7. Author(s) Peng Hao, Ph.D., https://orcid.org/0000-0001-5864-7358 Haishan Liu, Ph.D. Student, https://orcid.org/0000-0002-0817-9928 Yejia Liao, Ph.D. Student, https://orcid.org/0000-0003-4997-7528 Kanok Boriboonsomsin, Ph.D., https://orcid.org/0000-0003-2558-5343 Matthew J. Barth, Ph.D., https://orcid.org/0000-0002-4735-5859		**8. Performing Organization Report No.** N/A
9. Performing Organization Name and Address University of California, Riverside Bourns College of Engineering –Center for Environmental Research & Technology 1084 Columbia Avenue, Riverside, CA 92507		**10. Work Unit No.** N/A **11. Contract or Grant No.** USDOT Grant 69A3551747114
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of the Assistant Secretary for Research and Technology 1200 New Jersey Avenue, SE, Washington, DC 20590		**13. Type of Report and Period Covered** Final Research Report (February 2021 – June 2022) **14. Sponsoring Agency Code** USDOT OST-R

15. Supplementary Notes
DOI: https://doi.org/10.7922/G2610XPN
Dataset DOI: https://doi.org/10.6086/D19X1J

16. Abstract
Goods movement accounts for a significant and growing share of urban traffic, energy use and greenhouse gas emissions (GHGs). This project investigated the vehicle miles travelled (VMT) and emissions impact of on-demand food delivery under different COVID-19 pandemic periods and multiple operational strategies, with real-world scenarios set up in the city of Riverside, California. The evaluation results showed that during COVID-19 the total VMT and pollutant emissions (CO_2, CO, HC, NO_x) incurred by eat out demand all decreased by 25% compared with the before-COVID-19 period. The system can achieve substantial reductions in vehicle trips and emissions with higher penetration of on-demand delivery. From the dynamic operation perspective, the multi-restaurant strategy (allow food orders to be bundled from multiple restaurants in one driver's tour) can bring 28% of VMT and and emissions reductions while avoiding introducing additional delay compared to the one-restaurant policy (only allow food orders from the same restaurant to be bundled in one driver's tour). The research results indicate that the delivery platform should provide more reliable service with lower cost to increase the food delivery penetration level, which needs improvement in driver capacity management, eco-friendly delivery strategy, and efficient order allocation system. Meanwhile, to achieve maximum VMT and emissions reduction, the platform should encourage order bundling and employ a multi-restaurant policy to provide higher flexibility to group food orders, especially from restaurants located densely in one shopping plaza or commercial zone.

17. Key Words Shared mobility, On-demand food delivery, Sustainability, Emission evaluation	**18. Distribution Statement** No restrictions.		
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 61	**22. Price** N/A

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

NCST

About the National Center for Sustainable Transportation

The National Center for Sustainable Transportation is a consortium of leading universities committed to advancing an environmentally sustainable transportation system through cutting-edge research, direct policy engagement, and education of our future leaders. Consortium members include: University of California, Davis; University of California, Riverside; University of Southern California; California State University, Long Beach; Georgia Institute of Technology; and University of Vermont. More information can be found at: ncst.ucdavis.edu.

Disclaimer

Acknowledgments

This study was funded, partially or entirely, by a grant from the National Center for Sustainable Transportation (NCST), supported by the U.S. Department of Transportation (USDOT) through the University Transportation Centers program. The authors would like to thank the NCST and the USDOT for their support of university-based research in transportation, and especially for the funding provided in support of this project.

NCST

Developing Environmentally Friendly Solutions for On-Demand Food Delivery Service

A National Center for Sustainable Transportation Research Report

December 2022

Peng Hao, Center for Environmental Research & Technology, University of California, Riverside

Haishan Liu, Center for Environmental Research & Technology, University of California, Riverside

Yejia Liao, Center for Environmental Research & Technology, University of California, Riverside

Kanok Boriboonsomsin, Center for Environmental Research & Technology, University of California, Riverside

Matthew J. Barth, Center for Environmental Research & Technology, University of California, Riverside

NCST

TABLE OF CONTENTS

List of Tables

List of Figures

Developing Environmentally Friendly Solutions for On-Demand Food Delivery Service

EXECUTIVE SUMMARY

The urban freight transportation system serves the daily supply of cities by distributing millions of packages from either warehouses or restaurants and grocery stores to customers scattered in the city. Due to the fast-growing and excessive urban freight demand from international trade and e-commerce, road infrastructures have witnessed the fast growth of traffic demand, potentially leading to increased traffic congestion, excessive energy consumption, greenhouse gas (GHG) emissions and criteria pollutant emissions. Some innovative transportation solutions such as shared mobility services have the potential to improve the way of movement for goods. However, the business models of recent shared mobility service for freight are neither focused on reducing energy use nor optimizing vehicle utilization. Little attention was paid in investigating the environmental sustainability challenge and opportunity from the delivery operation perspective. There is great uncertainty regarding the extent to which these new freight mobility services will impact the environment.

This project proposes to improve vehicle utilization and energy efficiency by modeling and evaluating the innovative shared mobility services for freight, with a specific focus on on-demand food delivery. In this research, we use a well-calibrated CEMDAP model to generate realistic eat-out demand, utilize Riverside traffic network information to quantify the driver travel distance and travel time, and employ an emission model to evaluate the environmental impact of the eat-out incurred trips (including on-demand delivery and dine-in trips). Based on the generated trips, we first assume all information is known beforehand to study a static ODFD problem. A meta-heuristic Adaptive Large Neighborhood Search (ALNS) is proposed to solve the problem efficiently. Multiple scenarios were simulated to evaluate the urban traffic and environmental impact of on-demand food delivery services before and during COVID. The evaluation results show that the total travel distance incurred by eat-out demand during the COVID period decreased by 25% compared with the before-COVID period. Fuel consumption and tailpipe emissions were also reduced by 25% approximately. The benefit of ODFD is significant with the increased penetration rate. If the penetration rate of on-demand food delivery increases from 16% to 50%, then the total travel distance can be reduced by 31%.

We further extend the static assumption to dynamic setting, where food orders and drivers arrive and leave the ODFD system continuously. A rolling horizon optimization approach with ALNS algorithm is proposed to tackle the ODFD dynamism. Two delivery policies are proposed: One-R and Multi-R, which allow orders from one or multiple restaurants to be bundled in one driver's delivery trip, respectively. The system-level evaluation shows that on-demand food delivery has great potential to reduce dining-related VMT, resulting in reductions of fuel consumption and emissions, especially with Multi-R delivery policy. Under 14%, 21%, and 40% delivery penetration rate with the Multi-R policy, the total dining-related VMT can be reduced

by 5%, 10%, 25% respectively, compared to the baseline with no on-demand delivery, and the corresponding environmental impacts were also reduced significantly.

Overall, the shared mobility service has great potential to reduce freight transportation VMT cost and emission. With well-designed delivery policy, the on-demand food delivery can mitigate traffic in the urban city and bring a greener transportation system.

1. Introduction

The urban freight transportation system serves the daily supply of cities. Different from long-haul delivery which moves large quantity goods between two specific points using high-capacity vehicles along the well-designed highway or railway [1], the urban freight system aims to distribute millions of packages from either warehouses or restaurants and grocery stores to customers scattered in the city. Due to the fast-growing and excessive urban freight demand from international trade and e-commerce, road infrastructures will bear with high pressure, potentially leading to increased traffic congestion, excessive energy consumption, greenhouse gas (GHG) emissions and criteria pollutant emissions [2]. According to the U.S. Energy Information Administration, vehicle miles traveled (VMT) of freight trucks are expected to increase from 300 billion miles in 2019 to 415 billion miles in 2050 [3]. Innovative solutions are needed to address the growing freight demand that outpace the rate of expansion in supporting infrastructure.

Shared mobility services, which have revolutionized the way people move, have also broken ground in goods movement, e.g., collaborative shipping, shared delivery, paired on-demand passenger ride and courier services, and crowdsourced shipping. The long-haul delivery problem has been well studied and solved with collaborative shipping [4], which sets up collaboration between multiple less-than-truckload carriers to reduce truck VMT and energy consumption. For the last-mile delivery in urban area, shared delivery is a promising way to tackle the existing tough tasks. Shared delivery can support sustainability by using the excessive availability vehicles for goods deliveries without adding extra trips and the vehicle is shared within the city. In recent years, shared delivery developed rapidly thanks to the surge of internet companies. Any qualified drivers who sign up in the internet platform can deliver groceries, food takeout, or goods using their private vehicles, e.g., cars, bikes or scooters. As a complement or even alternative of freight trucks, shared delivery has the potential to reduce energy consumption and emissions in delivering light weighted goods. Especially, during COVID-19 dine-in closure in 2020, many restaurants switched to delivery or take-out modes to maintain their customers and revenues. Expanding shared delivery services thereafter supported the growing online food delivery need during the pandemic. Similar findings can be seen in the online goods and grocery delivery industry. Meanwhile, conventional delivery companies also have started to use crowdsourcing for parcel deliveries based on internet platforms. However, the business models of recent shared mobility services for freight are not focused on reducing energy use or optimizing vehicle utilization. Especially in shared delivery, little attention was paid in investigating the environmental sustainability challenge and opportunity from the delivery operation perspective. There is great uncertainty regarding the extent to which these new freight mobility services will impact the environment.

This project proposes to improve vehicle utilization and energy efficiency (productive ton-miles per unit of energy) by modeling and evaluating the innovative shared mobility services for freight, with a specific focus on on-demand food delivery. Among all urban freight delivery requests, on-demand food delivery is the most challenging one, in which orders arriving dynamically and urgent for fast delivery. Normally the food orders need to delivered within an

hour when placed and within only certain minutes after the food becoming ready. Meanwhile, the restaurants also scatters in the city, instead of an centerailzed location (depot). The approach that is able to solve the on-demand food delivery problem well can be readily applied to the grocery delivery and parcel delivery only with a minor modification of problem setting.

In this research, we present an On-demand Food Delivery (ODFD) system that is efficient to implement in real time, while considering both practical feasibility for service provider and system benefit for transportation and environment agencies. In an ODFD system, when an order is placed, the customer is provided with an expected delivery time. Then a centralized system would make order dispatching and driver routing decisions in time by coordinating all the new orders with available drivers starting with different locations, working schedules, and delivery capacities. The essential problem is to assign drivers with the optimal sequences of orders that minimize the total cost (i.e., delivery time, travel distance, driver compensation) while maximizing customer satisfaction. We then develop a rolling horizon-based approach with adaptive large neighborhood search (ALNS) to solve this ODFD problem, which leverages both the power of ALNS to optimize the large-scale ODFD problem for a given time step and a rolling horizon framework to deal with the system dynamism. The method is capable of handling both multiple-restaurant policy (i.e., orders from different restaurants have a chance to be bundled) and one-restaurant policy (i.e., only orders from the same restaurant can be bundle, so in a single trip the driver can only visit one restaurant). This optimization problem aims to minimize the weighted sum of both total delivery time overage and vehicle-miles-traveled (VMT) as objectives, which would meet customer satisfaction while mitigating the traffic congestion and environmental burden related to ODFD. A large-scale real-world delivery demand dataset based on City of Riverside network is generated with CEMDAP and BEAM model to validate the effectiveness of the proposed method and study the performance of both delivery policies. CEMDAP model and BEAM model are presented in Section 3. Our project mainly makes the following contributions:

1. Both static and dynamic Pick-up and Delivery Problem with Time Window (PDPTW) models are formulated to study the on-demand food delivery problem.
2. An efficient approach combining rolling horizon framework and ALNS to handle the large-scale meal delivery scenario in real-time manner.
3. Explore the ODFD demand under different COVID-19 period and study its corresponding operational and sustainable impact.
4. Explore different delivery policies to provide insights to the operational practice.
5. Extensive numerical study using delivery demand generated from a calibrated activity-based model and searching for the best route with dynamic traffic network information from an agent-based traffic simulation model.

The rest of this report is organized as follows. A literature review of meal delivery and its general mathematical problem is provided in Section 2. Section 3 shows the steps and results from the delivery demand generation model. Section 4 shows the methodology and results for on-demand food delivery with static information. A rolling horizon optimization framework with ALNS is proposed in Section 4, in which we also show the experimental results to validate

the proposed methods and investigate the impact of ODMD. Finally, some concluding remarks, limitation, and future research interests are presented in Section 6.

2. Shared Delivery Service in Freight Transportation

There are two types of shared mobility services in transportation: passenger (i.e., ridesharing services provided by Uber, Lyft, etc.) and freight (i.e., crowdsourcing services from Instacart, UberEATS, DoorDash, etc.). The shared mobility of people has been well-studied for a decade, which has been shown to have the potential of improving traffic efficiency and reducing VMT. Similar to the delivery of light-weighted freight in urban area, shared delivery can bundle delivery requests from nearby locations, thus having the potential to contribute to operational efficiency and environmental sustainability with optimized delivery strategy. In this project, we mainly focus on the shared delivery of online food orders: on-demand food delivery (ODFD).

2.1 On-demand food delivery (ODFD)

ODFD services have recently gained popularity around the world, especially during the COVID-19 pandemic, because they benefit both consumers and restaurants by providing trouble-free, efficient, and expedient online food ordering and offline food delivery services [5]. According to the report from Statista, by 2021 digital food delivery comprised 14% of the total market and will keep growing steadily in next 5 years [6].

Most conventional meal delivery services are restaurant-operated, i.e., the resturant itself owns a professional delivery fleet [7], where the delivery is reliable but expensive. As the fleet only serves the specific restaurant, the conventional restaurant-operated delivery system fails to scale up to deal with the dynamic large-scale volume of food orders and cannot meet the customer's expectation of fast and low-cost delivery service [8]. In contrast, the on-demand meal delivery, also known as shared delivery or Online-to-Offline (O2O) food delivery, is a service where any qualified driver registered in the delivery platform can deliver cooked food with their private vehicles, i.e., cars, bikes or scooters [9], [10]. The delivery drivers' resources are somehow "shared" between various restaurants, which is suitable for a market that is highly fluctuated and diverse in time. However, to provide meal delivery services efficiently with a flexible driver fleet, many challenges still need to be overcome. The first main challenge is large-scale food delivery demand which puts too much pressure on the traditional optimization methods to gain the optimal solution, i.e, branch and cut, or branch and bound [11]. The solution space tends to grow exponentially with the demand scale. The other concern is the highly dynamic and urgent aspect of the delivery orders, as the orders flow into the system continously and quick response and action are needed to deliver the food order within less than an hour after the food becoming ready. Last, most on-demand meal delivery services today are operated by internet-based companies, which are more focused on commercial perspectives such as profit, incentives, and customer satisfaction, rather than how the deliveries impact the transportation system and the environment. Therefore, the on-demand food delivery has been regarded as the ultimate challenge in last-mile logistics [8].

There are four main stakeholders in ODFD, namely, customers, restaurants, delivery drivers, and service platform. Customers expect fast and lower-cost food delivery service. Restaurants require specialized and reliable service. Drivers want to maximize delivery earnings per order and have a good delivery experience. To meet all the expectations above, the ODFD platform has to coordinate assigning orders to proper drivers with consideration of order location, driver capacity, restaurant food preparation time, etc. And the ODFD platform itself also aims to maximize its own operational benefits by charging the commission fee per order.

The general ODFD process is depicted in Figure 1. The customers first place the order using their mobile phone. After receiving the order request, the delivery platform confirms the request information with the corresponding food providers. Meanwhile, based on a collection of food orders and available drivers, the platform needs to assign each order to the best driver with the consideration of the driver's location, availability, and scheduled deliveries. Then the drivers will pick up orders from the restaurants and deliver them to the customers to finish the task. The order sequence assignment and route planning are critical for the platform because it needs to meet the customer expectation of fast and on-time food delivery and reduce the travel distance as much as possible to ease the driving burden for drivers.

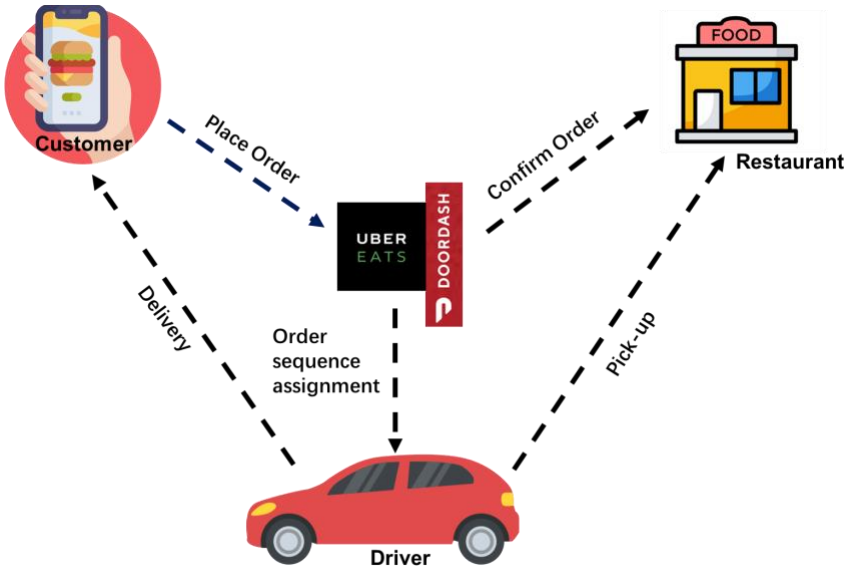

Figure 1. On-demand food delivery system

2.2 Related work of ODFD

On-demand food delivery service is an emerging area—most research results were published in the past five years. Due to the stochastic feature of meal delivery problem, some researchers assume all order and driver information is known beforehand, and simply solve a static meal routing problem. Liu et al. proposed to leverage the taxi resources to deliver food orders either in opportunistic manner or in dedicated manner with the goal of minimizing taxi number and distance cost. They generated 200 orders within Chengdu city and used ALNS algorithm to solve the scenario [12]. Tu et al. developed an online dynamic optimization framework which

includes order collection, solution generation, and sequential delivery. But their approach lacks interactions between each time interval, so this approach is essentially to solve a static small scale meal delivery problem in every time step [13]. Wang et al. presented an insertion-based heuristic to solve a single driver food delivery routing problem along with the geographic information to accelerate the insertion process and an XGBoost to select the order sequencing rules [14].

To study the meal delivery problem in the real-world setting, many researchers also incorporate the ODFD system dynamism in their models. Zhou et al. formulated an online order dispatch system with new order arrival and extended the traditional greedy insertion and regret insertion heuristic to evaluate more orders in one iteration [15]. But they only solve the problem in one time interval without considering the platform update. Reyes et al. studied the meal-delivery routing problem (MDRP) and proposed a rolling-horizon algorithm to solve the dynamic vehicle routing problem and capacity management problem [8]. Yildiz and Savalesbergh further extended Reyes' research and introduced the concept of work-package which allows it to be solved by a column and row generation method [16]. The two research demonstrated their performance with the order instance from Grubhub. Steever et al. studied a scenario where one customer is allowed to place multiple orders in one restaurant and proposed two policies, split policy and non-split policy, for the system to make decisions [17].

However, most research focuses on obtaining the operational optimization result by minimizing order delivery delay and delivery cost in terms of miles travel and compensation to drivers. The evaluation metrics are also the same as the operational objectives. All research mentioned above lack an environmental perspective evaluation of the on-demand food delivery. Meanwhile, most research has focused on developing efficient algorithms to solve the ODFD problem validated with small-scale, generated order demands, with which we cannot draw a conclusion on the city-scale impact of ODFD service. Although the research in [8] claims to be large scale since they solved the instances with 3000 orders, these orders spread over a 800 minute horizon which indicates a lower order intensity in each minute.

To fill the research gaps in ODFD, in this research we study a real-world on-demand food delivery problem with higher order intensity in each time window. We also evaluate delivery-incurred energy consumption and pollutant emissions to provide insights into developing an eco-friendly on-demand food delivery system.

3. Demand Generation for Eat-out Trips

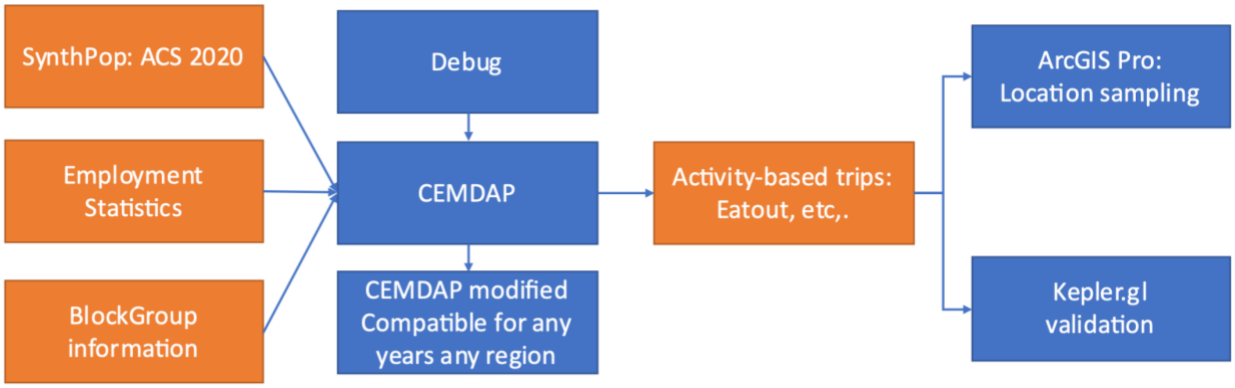

Figure 2. Eat-out trips generation workflow

In this part, we first show the real-world eat-out data generation process. Figure 2 shows the workflow to generate on-demand trips in the transportation system based on up-to-date census population, travel survey, and traffic analysis zone information. The SynthPop [18] is a reimplementation of PopGen [19] using modern scientific Python stack, which provides the ability to synthesize population within small geographies (e.g., census tract, block group). Longitudinal Employer-Household Dynamic (LEHD) Origin-Destination Employment Statistics [20] provides zone-based commuting trips. Block group information, including employment, population, and distance, is the input for CEMDAP. Random points generated by ArcGIS Pro are assigned to each trip and Kepler.gl is the data visualization tool used to do the data validation.

3.1 SynthPop

A key for CENSUS API is needed in SynthPop, and we can get this key by signing up through the link: http://api.census.gov/data/key_signup.html. With the key, the SynthPop can download PUMS/American Community Survey data for a certain year that is specified by the user from SynthPop server that has already preprocessed large files into small files. Then, the SynthPop can be implemented easily for synthesizing populations while matching population controls in small geographies.

In this project, we synthesized the population of the city of Riverside, California. The total population is 353,860, while the household number is 104,758. As a comparison, Google search shows that the total population in the city of Riverside is 327,569. The synthetic population is a little more than the number given by Google search, because the geographies (i.e., block groups) are a little larger than the city of Riverside. Figure 3 indicates the geographies we used to synthesize population, where the bottom right is outside the city of Riverside.

3.2 LEHD Original-Destination Employment Statistics (LODES)

The Longitudinal Employer-Household Dynamics (LEHD) program is part of the Center for Economic Studies at the U.S. Census Bureau. This program creates statistics on employment, earnings, and job flow at detailed levels of geography and industry and for different

demographic groups. In addition, the LEHD program uses this data to create partially synthetic data on workers' residential pattern.

Figure 4 shows a demo job count by home places (e.g., cities) using data from LODES. Dark blue color means a greater job count in this area while white color means a lesser job count. In this project, we extract employment statistics from a more detailed level of geography, which is block level. Around 150,000 work destinations occur in the city of Riverside and these trip origin-destination pairs would combine with the synthetic population we created from SynthPop. We observe that there are some workers who live in Riverside working outside Riverside, while some workers who lived outside Riverside work in Riverside. However, the eat-out trips generated by CEMDAP are based on the workplace (destination) in the Employment statistics data, so it is less vital where the workers come from. To simplify our travel pattern without losing travel demand, we make Riverside an isolated island, which means each worker living in Riverside would only work inside the city and workers outside the city would not come in. As such, the amount of job counts is kept the same as the LODES employment statistics, and sufficient eat-out trips based on the places of jobs are created.

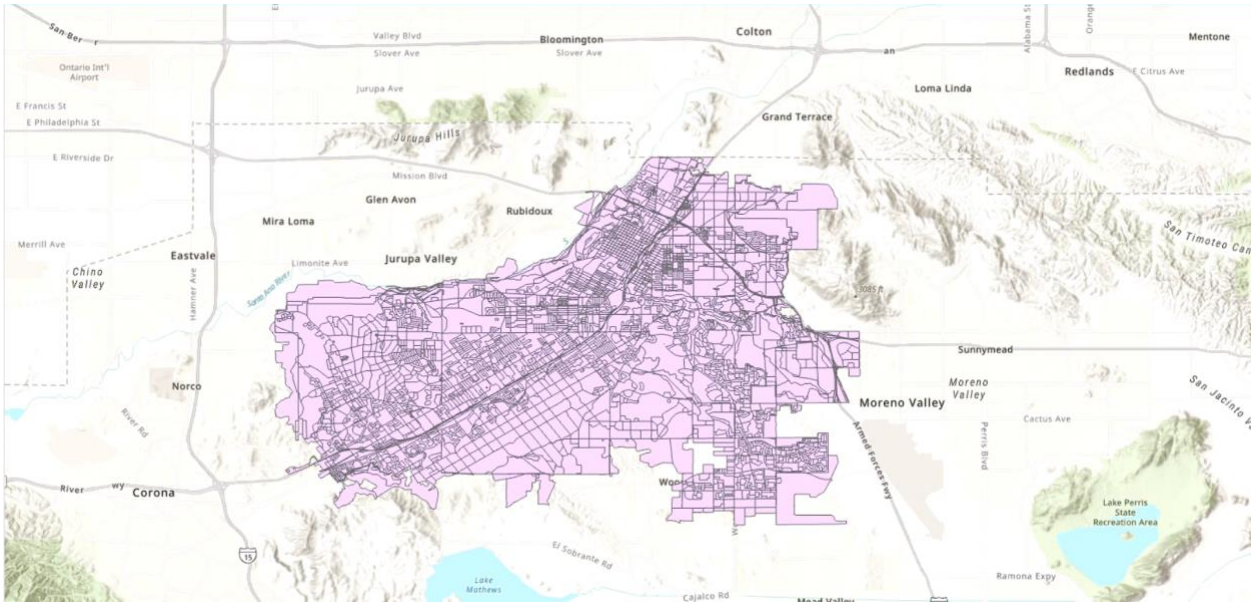

Figure 3. Synthetic geographies, block groups

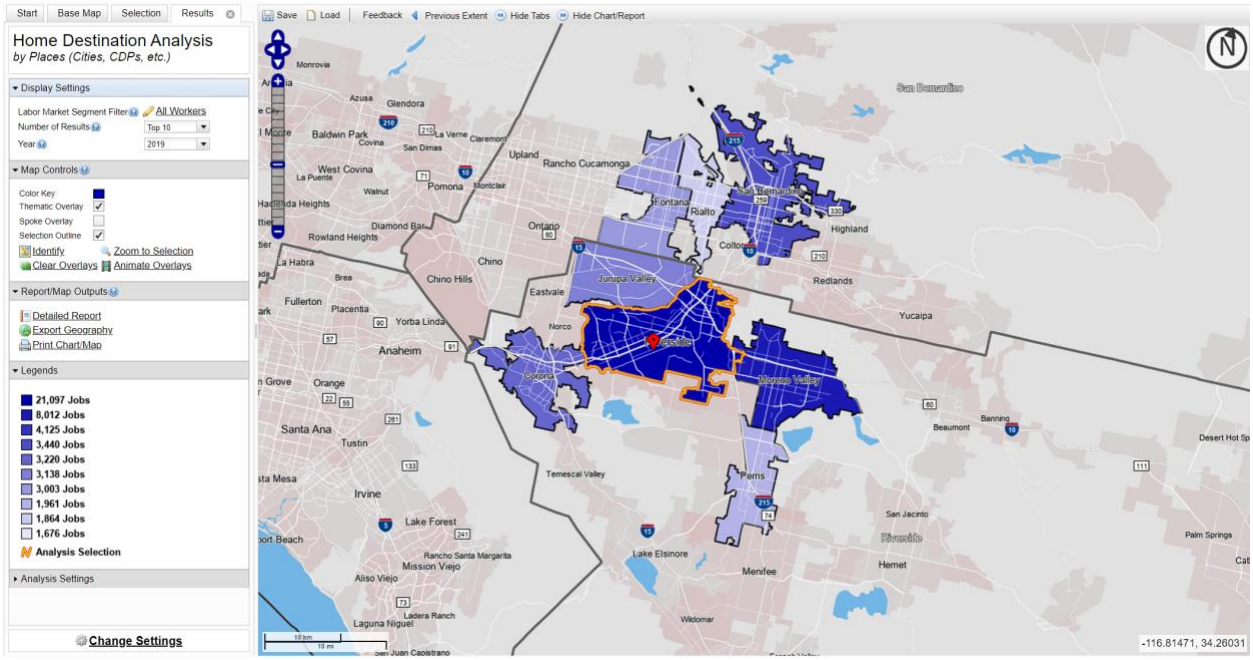

Figure 4. Employment statistics

3.3 Block Group Information

The origin-destination pairs are zone-based (i.e., blocks and block groups). To assign precise workplaces for workers within block/block groups, we use employment data from the Employment Development Department of the State of California [21]. This data provides precise information about employers, including address, category (e.g., retail, restaurant), size, etc. Figure 5 shows the addresses of restaurants within the city of Riverside in Kepler.gl.

Figure 5. Restaurant locations

3.4 CEMDAP

CEMDAP offers a user-friendly environment to simulate the activity-travel patterns of a population by using standard Windows user interface features as shown in Figure 6. Given as inputs various land-use, sociodemographic, activity system, and transportation level-of-service attributes, the system provides as outputs the complete daily activity-travel patterns for each individual in each household of a population [22].

The PostgresSQL and PSQLDBC need to be installed prior to running CEMDAP. Also, the households table, persons table, zone table, zone to zone table, and level of service table should be prepared. More details about operations, database registration, input files, and output files can be found in the CEMDAP user manual. However, users should be aware of some consistency issues in the household table and person table. For instance, total number of persons in household should be equal to the number of adults plus the number of children, otherwise CEMDAP will crash. What is more, the household structure is not clearly defined in the user manual, so we must make assumptions and do the test to match the household structures and labels to avoid crashing when running CEMDAP.

According to the analysis, there are 72 household structures in the synthetic households and population, which is the combination of number of adults and children (e.g., one adult with no

child, two adults with one children). Because the CEMDAP software did not document well which household structure can be processed and which cannot be, we have to test different household structures before they can be processed by CEMDAP. Since there are too many household structures, we prioritize testing the household structures with more population and now we can process 18 out of 72 household structures as shown in Figure 7. These 18 household structures make up 86% of the population, which should be enough to show on-demand trips in the city of Riverside. The remaining 54 out of 72 household structures can be tested but will cost lot of time. All the codes [23] have been uploaded to GitHub and are ready for use for any year and place in the U.S.

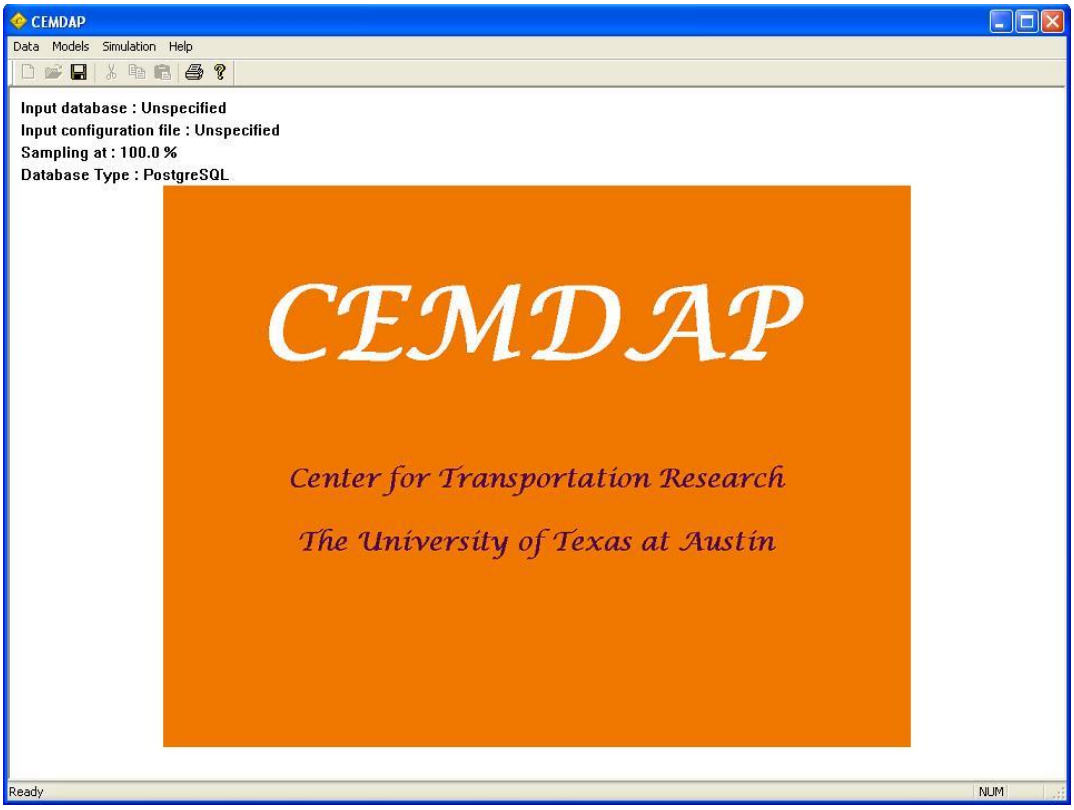

Figure 6. CEMDAP user interface

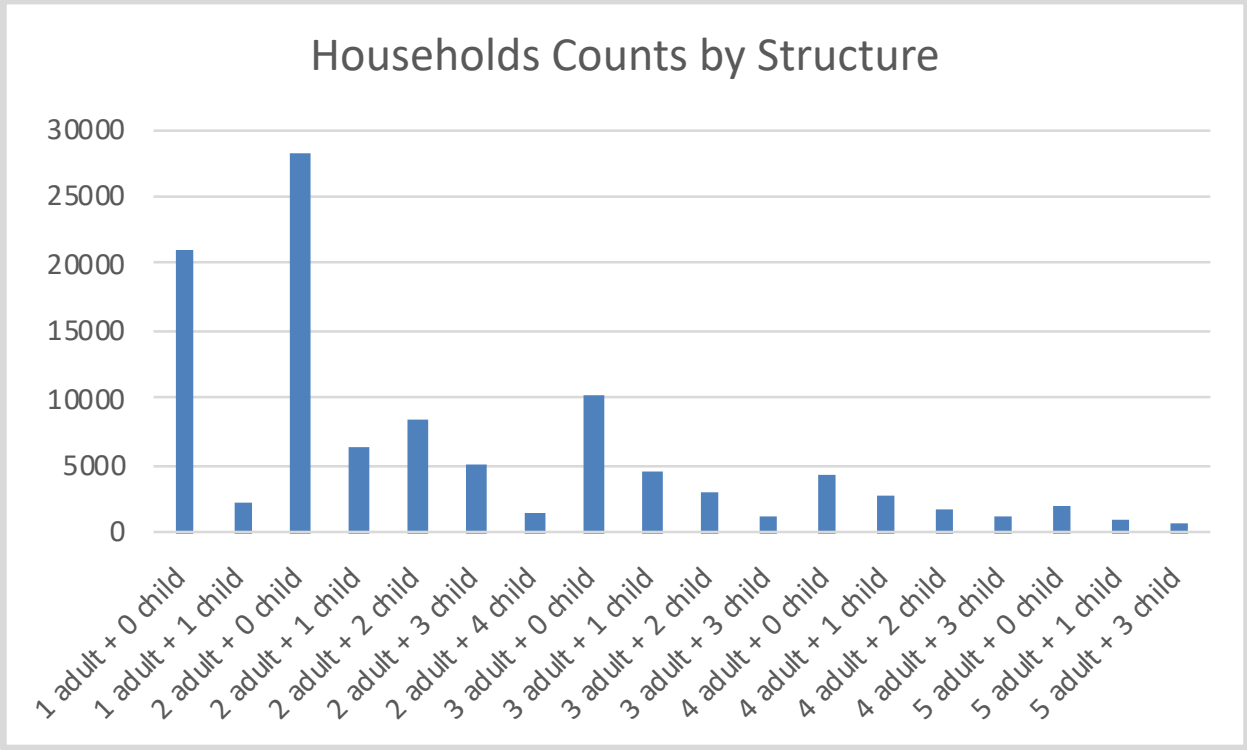

Figure 7. The household structure distribution

3.5 Trips validation and visualization

We use ArcGIS Pro to generate random points within block groups that can be used to be assigned to home locations as shown in Figure 8. Kepler.gl is an online visualization tool. As in Figure 9. The total eat-out trips in City of Riverside

, the red points refer to restaurant location while the green points refer to customer location. The red clusters in the map are around the business centers and restaurants, which suggests a reliable result.

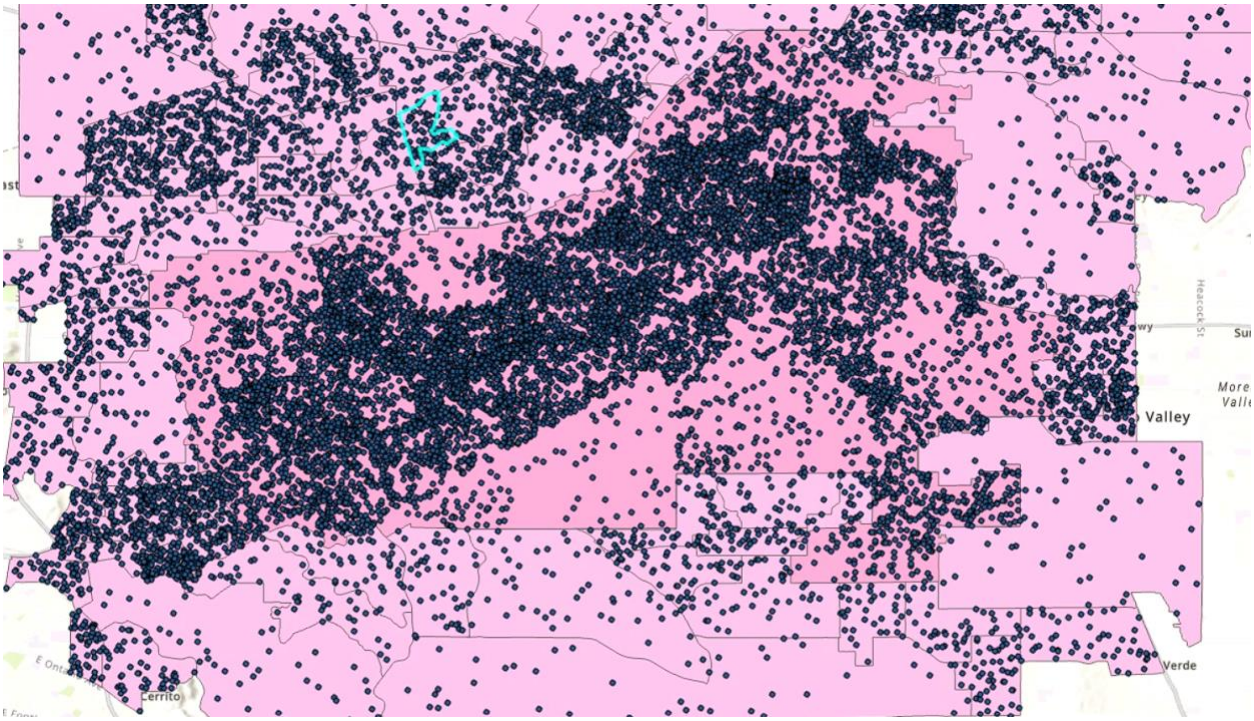

Figure 8. Random points in ArcGIS

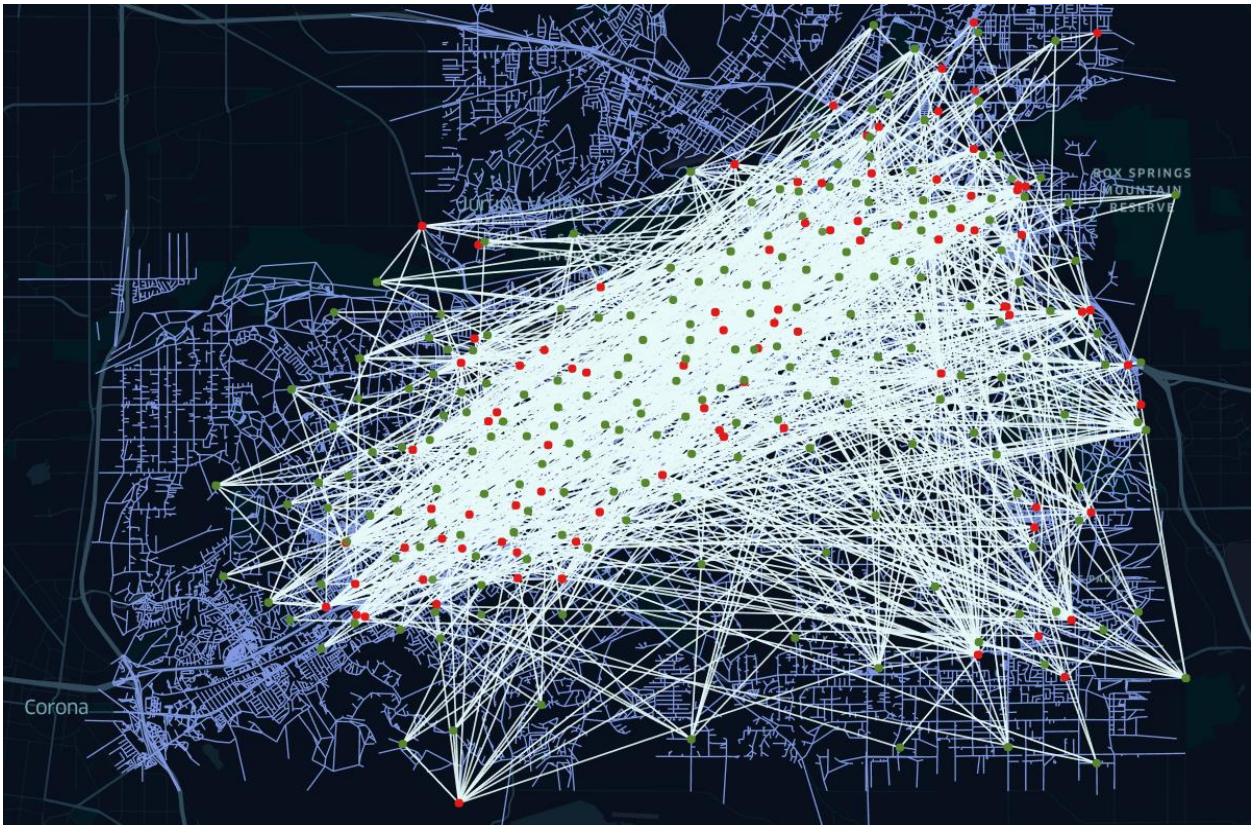

Figure 9. The total eat-out trips in City of Riverside

4. On-demand food delivery with static information

In an ODFD system, when an order is placed, the customer is provided with an expected delivery time. Then a centralized system would make order dispatching and driver routing decision in time by coordinating all the new orders with available drivers starting with different locations, working schedules, and delivery capacities. The essential problem is to assign drivers with the optimal sequences of orders that minimize the total cost (i.e., delivery time, travel distance, driver compensation) while maximizing customer satisfaction. In this section, we first assume that all information is known to the system, including every order placing time, customer location, restaurant location, drivers' shift plan and driver location. In this case, the ODFD problem can be formulated as a static Pickup and Delivery with Time Window (PDPTW) model. The ODFD problem aims to assign orders to proper drivers and deliver the order as quickly as possible with lower delivery cost. Specifically, we have following assumptions regarding our static ODFD problem.

1. All drivers are originally idle around the restaurant zone, thus the first-bound detour of the driver to visit the restaurant is not considered in our problem.
2. Each order must be served. Rejection is not allowed, which may bring challenges to the proposed approach to deal with a large number of orders, especially during peak hours.
3. Each driver has limited capacity. Although one car can carry many food orders, we are still inclined to set a capacity limit. This is because with excessive orders at a given time, the driver will have a bad delivery experience (easy to mess up between different customers) and the food freshness may be sacrificed if the food stays in the vehicle for a long time.

With these assumptions, the ODFD problem can be formulated as in Section 4.1.

4.1 Static PDPTW model

Assuming that there are *n* orders and *m* drivers in the system, then an undirected graph G = (*V*, *E*) can be defined, in which each node represents the location of a customer, a restaurant, or driver, i.e., V =$P \cup D \cup K$, and each arc (E = $V \times V$) represents the movement from one node to another. Specifically, order *i* can be defined as $\langle i, i+n, q_i, q_{i+n}, EPT_i, ETA_i \rangle$, where $q_i + q_{i+n} = 0$. Then we can formulate the on-demand delivery problem as follows (all parameters and variables are listed in Table 1):

Table 1. Parameters and variables definition

Symbol	Description
n	Number of orders
m	Number of drivers
P	Set of restaurants. Pick-up point $\{1, \dots, n\}$
D	Set of customers. Delivery point $\{n+1, \dots, 2n\}$
K	Set of drivers. Initial location $\{2n+1, \dots, 2n+m\}$
q_i	Loads needed to be transported at node i. Positive when i is a pick-up point; negative when i is a delivery node.
Q^k	Capacity of rider k
t_{ij}^k	Travel time of link ij with rider k
s_i	Service time at node i (load/unload)
EPT_i	The earliest pick-up time of restaurant i
ETA_i	The regular expected time of arrival of customer i
d_{ij}	Travel distance of link ij
Decision variable	x_{ij}^k: driver k use link(i,j) Q_i^k: load of vehicle k when leave node i T_i^k: time when vehicle k arrive node i

The static PDPTW model for ODFD problem:

$$\min F = \alpha \sum_{k \in K} \sum_{(i,j) \in E} d_{ij} x_{ij}^k + \beta \sum_{i \in D} max(0, T_i^k - ETA_i)$$

subject to:

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad \forall i \in P \cup D \qquad (1)$$

$$\sum_{j \in P} x_{2n+k,j}^k = 1 \qquad \forall\, k \in K \qquad (2)$$

$$\sum_{i \in D} x_{i,2n+k}^k = 1 \qquad \forall k \in K \qquad (3)$$

$$\sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall j \in P \cup D, \forall\, k \in K \qquad (4)$$

$$\sum_{j \in V} x_{ij}^k - \sum_{j \in V} x_{j,n+i}^k = 0 \quad \forall i \in P, \forall k \in K \qquad (5)$$

$$x_{ij}^k = 1 \Rightarrow Q_j^k \geq Q_i^k + q_j \quad \forall (i,j) \in E, \forall k \in K \qquad (6)$$

$$Q_i^k \leq Q^k \qquad \forall i \in V, \forall k \in K \qquad (7)$$

$$x_{ij}^k = 1 \Rightarrow T_j^k \geq T_i^k + t_{ij} + s_i \quad \forall (i,j) \in E, \forall k \in K \qquad (8)$$

$$T_i^k \geq EPT_i \qquad \forall i \in P \qquad (9)$$

$$T_i^k \leq T_{n+i}^k \qquad \forall i \in P, \forall k \in K \qquad (10)$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K \qquad (11)$$

$$Q_i^k \geq 0 \qquad \forall i \in V, \forall k \in K \qquad (12)$$

$$T_i^k \geq 0 \qquad \forall i \in V, \forall k \qquad (13)$$

The objective of this problem is to minimize total travel distance and total order delay. The order delay is defined as the difference between actual delivery time and predefined expected time of arrival (*ETA*). α and β are weight factors designed to balance the distance and delay. Constraint (1) ensures that any customer or restaurant will be visited once, i.e., all orders in the system will be serviced. Constraint (2) and Constraint (3) define drivers' first stop and last stop of the trip. Constraint (4) guarantees the flow conservation of the route. Constraint (5) ensures that each order should be picked up and delivered by the same driver. The driver capacity change along the path and its limit is defined in Constraint (6) and (7). Constraint (8) states that driver arrival time at node *j* is no less than arrival time of the previous point *i* plus travel time from node *i* to *j* and service time at node *i*. Constraints (9) and (10) allow each driver to pick-up the order no earlier than *EPT*, and then deliver the order to the corresponding customer's location. Decision variables are defined in Constraint (11)-(13). The solution of the on-demand delivery problem is a set of order sequences assigned to multiple drivers.

In this problem, ETA is defined as the regular expected time of arrival of each order assuming regular traffic and delivery demand level. Under peak hour, the congested traffic and high demand may cause unavoidable additional delay to the orders. To accommodate this condition, we set time window as a soft constraint in this problem rather than a hard constraint in the general PDPTW model [24]. If the actual arrival time is later than the regular ETA, this route still feasible but a penalty will be recorded in the objective function.

4.2 Adaptive Large Neighborhood Search (ALNS)

The ODFD problem is a variant of Vehicle Routing Problem (VRP; in this case the driver needs to execute both pickup and delivery task along the route). The VRP possesses the feature of NP hardness. Therefore, it is impossible to find the optimal solution of ODFD problem in acceptable running time. Here, we seek for a meta-heuristic method--Adaptive Large Neighborhood Search (ALNS)—to solve our ODFD problem efficiently. ALNS is a mature search framework in which a number of simple operators are selected in a structured way to improve the current solution. In each iteration, a pair of destroy operator (utilized to remove some undesirable assignments) and repair operator (used to reinsert back the removed tasks into a better position) is selected. The performance of the operators is evaluated with the new solution and will be recorded to

update the weights of them in the future. ALNS has been proven to have several advantages: 1) it can provide high-quality solution, 2) the algorithm is robust, 3) it is self-calibrating to some extent, and 4) it has the capability to solve large-scale planning problems (i.e., routing, scheduling).

4.2.1 Construct Initial solution

The first step of ALNS is to construct an initial solution. We proposed a simple heuristic to generate the initial solution. It consists of two steps: 1) Construct the sorted order sequence with respect to ETA, then assign the first order to each nearest driver. 2) Exploit greedy insertion to plan all remaining orders according to the increasement of objective. The detailed approach is shown in Algorithm 1. Based on the sorted order queue (line 1), each driver will be assigned with one order sequentially (line 2). Then with the partial route of each driver, the remaining orders will be greedily inserted (greedy insertion described in the next section) into the best driver route and the best position (line 3 – line 7). The best insertion position for each order n is calculated with $\min_k c_n^k$, where c_n^k represents the change in objective value after inserting order n into route k's position that incurs the least objective change. Meanwhile, an insertion will be rejected if this insertion will cause the driver to exceed capacity or work schedule.

Algorithm 1. Construction Algorithm

Input: N: the order set, K: the driver set
Output: the initial route for each driver k
1: Priority Queue L← Sort_Order_by_ETA(N)
2: Pop out order sequentially and assign to an empty driver
3: **for** order *n* in *N* **do**
4: **for** driver *k* in *K* **do**
5: greedy insertion (*n*, *k*)
6: **end for**
7: **end for**

4.2.2 Solution improvement with ALNS

Although the initial solution is feasible, it might involve unacceptable delivery delay and inappropriate task sequences. Hence, we need to further improve the initial solution. We utilized the ALNS framework, in which multiple removal and repair operators are selected based on an adaptive selecting mechanism, to diversify and intensify the initial solution then get the optimized one. ALNS can explore large neighborhoods in a structured way, and it thus has the potential to escape the local minimum, which is common in other search frameworks, i.e., simulated annealing and Tabu Search. The details of the method are described as follows.

1. Removal Process: First, the current solution is destroyed with one of the following operators. In this problem, each order consists of one pick-up and one delivery task, both needing to be served by the same driver. Thus, in the removal process, we will remove a pair of tasks at each iteration. The removed tasks will be placed in a task pool.

a. *Random removal:* This operator randomly selects *N* tasks to remove to diversify the solution space.
b. *Worst Removal:* We rank the insertion cost of every order in an increasing order, introducing a random number $y \in (0,1)$and a parameter p, then removing the order located at $y^p|N|$. This randomization is implemented to avoid removing the same task repeatedly.
c. *Shaw Removal:* This operator was first proposed to solve VRP problems based on evaluating the similarity of two locations [25]. In on-demand food delivery problem, slight modifications are needed since the smallest unit in our problem is an order which consists of two locations.
d. *Distance-Based Path Removal:* This operator is designed to remove the total route for one driver based on travel distance. All tasks on this route will be placed into the task pool directly and the driver's task sequence becomes empty.
e. *Delay-Based path removal:* Similar to Distance-Based Path Removal, this Delay-Based operator will pick the route with longest total delay, then remove all tasks on that route.

2. Repair Process: Repair operator is employed to insert back the task into the destroyed solution. In this project, parallel insertion heuristics is chosen so that multiple routes are built simultaneously. Also, to reduce the computational complexity, we sort orders using ETA then pop out each one to be inserted back.
    a. *Random Repair Operator:* Randomly select *N* feasible position, then insert the tasks. Similar to the Random removal operator, this operator also perturbs the solution space.
    b. *Greedy Insertion Operator:* Greedily insert every task into the best position such that the change of objective function is minimized.
    c. *Regret-q Insertion Operator:* The main drawback of greedy insertion is that it might leave the most "expensive" task to the last iteration where we lack flexibility. The Regret-q insertion could avoid this situation by incorporating look-ahead information. Let Δc_i^j indicate the objective change when inserting task *i* into *j*th cheapest position. Then we need to find the order *i* that maximizes the regret value (14). In this project, Regret-2 and Regret-3 insertion operators are constructed.

$$\max_i \left\{\sum_{j=1}^{q} (\Delta c_i^j - \Delta c_i^1)\right\} \qquad (14)$$

3. Adaptive Weighting and Selection Mechanism.

Instead of only selecting one removal and one insertion operator in the entire searching process, ALNS uses all operators proposed above. In each iteration, one removal and one repair operator are selected independently based on the Roulette Wheel Selection Principle (see (Equation 15)) to generate a new solution. At the beginning, each operator is equally weighted. Weights will be updated after a segment of iterations. Assume that we have *n* operators and

operator *i* with weight w_i^k at segment k. Then the probability of choosing operator *i* at segment *k* is defined:

$$P_i^k = \frac{w_i^k}{\sum_{j=1}^{n} w_j^k} \qquad (15)$$

An adaptive weight adjustment method is introduced (see (16)) to update the weight according to the operator performance.

$$w_i^k = (1-\rho)w_i^{k-1} + \rho\frac{\mu_i}{\pi_i} \qquad (16)$$

The weight of operator *i* at segment k (w_i^k) is derived based on the weight at segment *k-1*(w_i^{k-1}). ρ is a reaction factor that controls the speed of this algorithm reacting to the effectiveness of operators. π_i indicates the number of times that operator *i* is chosen in this segment. μ_i is the accumulated score of operator *i* (see (Equation 17)). At the beginning of each segment, every accumulated score will be set to zero. In each iteration, a score γ^k will be assigned to the operator, which represents the performance of the operator (see (18)).

$$\mu_i = \sum_{j=1}^{\pi_i} \gamma_j^k \qquad (17)$$

$$\gamma_j^k = \begin{cases} \gamma^1 & \text{if a new best solution is obtained} \\ \gamma^2 & \text{if the solution is better than the current solution} \\ \gamma^3 & \text{if the solution is worse than the current one but still accepted.} \end{cases} \qquad (18)$$

4. Acceptance and termination criteria.

To avoid getting trapped in a local minimum, we use simulated annealing strategy to accept a worse solution $s'$with probability of $e^{-\frac{f(s')-f(s)}{T}}$, where *f* is the objective function and $T > 0$ is the temperature. T will decrease with a cooling rate δ: $T = \delta T(\ 0 < \delta < 1)$. Considering the practicality, we prefer good results in short time rather than getting the optimal solution in long computational time. Thus, we proposed the following termination criteria to stop the ALNS algorithm. The algorithm will terminate if one of the rules is met: the maximum number of iterations φ_{max} is reached; or φ iterations have been executed without any improvements.

4.2.3 Environmental impact evaluation

To study the environmental impact of on-demand delivery, we use the link-based traffic data to calculate fuel consumption and emissions (i.e., CO_2, CO, HC, NO_x) for each link with the following formulation proposed in [26].

$$\ln(f_{ij}) = \beta_0 + \beta_1 v_{ij} + \beta_2 v_{ij}^2 + \beta_3 v_{ij}^3 + \ +\beta_4 v_{ij}^4 + \beta_5 g_{ij}$$

f_{ij} is the energy consumption rate/emission rate of link *ij*, which indicates energy consumption/ emission per unit distance (gram/mile). v_{ij} is the link speed and g_{ij} is the road grade of link *ij* (%). In this project, we assume grade of each link to be zero. β_0 to β_5 are the parameters of

different factors calibrated in [26]. Energy consumption/emission of link *ij* is $fc_{ij} = f_{ij} \times L_{ij}$, where L_{ij} is the length of link *ij* (mile). We sum up the energy consumption/emission of the delivery vehicles travelling on all the links in the network to evaluate its impact.

4.3 Impact of on-demand food delivery during Pandemic

4.3.1 Data description

A well-calibrated CEMDAP model is applied to generate eat-out demand in Riverside during the lunch time (described in Section 3), from 11:00am-13:00pm, and then the eat-out demand is partially converted to on-demand delivery orders based on the delivery ratio. As an example, Figure 10 shows the on-demand delivery customer location and restaurant distribution. There are in total 649 delivery orders, which represent a large-scale scenario for a PDPTW problem with high computational load for ALNS algorithm. We use K-means algorithm to first cluster the food orders according to the customer location; in this sense drivers have the possibility to serve the adjacent customers together. Because some clusters still have a too-large number of orders, i.e., larger than 100 orders in one cluster, we further divide the orders into several small clusters with order amounts no larger than 60. After the clustering process, each small order group can be solved independently. As most restaurants are located in the commercial zone, allowing drivers to pick up multiple orders in a short trip then deliver to a specific community, this clustering method could significantly improve the computation efficiency while not sacrificing the optimality too much.

Figure 10. Distribution of customers and restaurants in Riverside

4.3.2 Experiment Setup

In the on-demand food delivery service, whether the food is delivered on time is the main factor influencing customer experience. Hence, time window setting and travel time estimation is crucial in the optimization problem. After the customer places one order, we assume the restaurant needs 5-10 mins to prepare for it, which defines the order's earliest pick-up time (EPT). Based on EPT, we then set the ETA with consideration of the order distance between

customer and restaurant. For the service time, drivers need one extra minute for pick-up or delivery when arriving at the location. Further, for total driver number, we set the order/driver ratio to five. We further assume that each driver can be assigned up to 10 orders in one working schedule to avoid delays. Based on multiple experiments, we set the weight α and β both equal to one in the objective function. For parameter setting in ALNS algorithm, we take [24] as the reference.

Moreover, in this research, real network traffic information is utilized to predict the delivery time of each order accurately. First, we use real-time Riverside link speed data to represent the delivery speed rather than simply assume it to be constant. Then we do the routing in the Riverside network and get the shortest path between two locations using the Networkx module [27]. Link travel time can be obtained by dividing link length by average link speed, and all link travel times in a path are summed up to calculate the path travel time.

The COVID-19 pandemic has changed people's working and living styles and it also impacted people's eat-out patterns. In this project, we use "COVID" to refer to the COVID-19 pandemic. In [28], the authors studied the behavior change of four eat-out modes including dine-in, take-out, pick-up and on-demand delivery before and during COVID. We then calibrate the on-demand delivery ratio based on the research results in [28], which is 16% and 21% in before-COVID and during-COVID periods, respectively. The total eat-out demand in during-COVID also reduces by 34% due to regulations and concerns during COVID.

We then define six scenarios with respect to COVID and ODFD penetration rate as follows. The penetration rate is set to be 0%, 16% (before COVID) & 21% (during COVID), and 50%. The on-demand food delivery ratio and total eat-out demand change as presented in Table 2. First, the eat-out demand reduced by 34% compared with the before-COVID situation due to people's hygiene concern. The scenario name indicates the COVID status and ODFD penetration rate, i.e., B_COVID_0% means before COVID with 0% ODFD penetration rate, thus no ODFD service is offering. D_COVID indicates the during-COVID status.

Table 2. Different on-demand food delivery ratio in six scenarios setting

Class	Scenario	% of original demand using ODFD	% of original demand not using ODFD	% of demand loss due to COVID
Non-Delivery	B_COVID_0%	0	100	0
	D_COVID_0%	0	100	34
Mid-Delivery	B_COVID_16%	16	84	0
	D_COVID_21%	21	79	34
High-Delivery	B_COVID_50%	50	50	0
	D_COVID_50%	50	50	34

4.3.3 ODFD impact before and during COVID

First, with the penetration rate from real-world study in [28], we study the impact of ODFD during COVID period. In addition, we create a simple baseline to study the performance of on-demand food delivery. The baseline assumes that all delivery orders turn out to be self-pick-up orders.

All results are summarized in Table 3. In the before-COVID case, it contains 649 orders accounting for 16% of total eat-out demand. Compared with the baseline in which all customers pick up independently, on-demand food delivery can reduce the total travel distance by 78%, and meanwhile save above 54% of fuel consumption and tailpipe emissions (including CO_2, CO, HC, and NO_x). These decreases are as expected since on-demand food delivery can bundle more orders together to pick up most food orders in the commercial zone first, then deliver to the customers sequentially, avoiding redundant trips between restaurants and communities. However, we may overestimate the VMT, fuel, and emission benefit in this research as 1) we ignore the trip to pick up the first order by assuming drivers locate around restaurants; and 2) we assume all orders are ideally optimized by distance and delay, ignoring other factors in a realistic business model. In the during-COVID case, the order amount slightly decreases to 558 due to COVID, while accounting for more percentage (21%) because total eat-out demand during COVID shrinks. Comparing D_COVID_21% with its baseline, similar results can also be gained. This result demonstrates the excellent performance of on-demand food delivery in terms of reducing VMT, fuels and emissions.

Table 3. On-demand food delivery VS Baseline

		B_COVID 16% Baseline	B_COVID 16%	D_COVID 16% Baseline	D_COVID 16%
Number of delivery order		0	649	0	558
Number of self pick-up order		649	0	558	0
% of total eat out demand		16%	16%	21%	21%
Travel Distance(km)		15439.06	3427.78	13937.36	3009.51
Ratio of late delivery		0.00	0.17	0.00	0.62
Fuel(kg)		939.11	426.26	845.13	370.64
CO2(kg)		2995.70	1359.73	2695.91	1182.31
CO(g)		1154.45	519.39	1040.37	453.36
HC(g)		133.28	60.28	120.02	52.50
NOx(g)		912.07	407.11	822.90	356.59
Shared Delivery VS Baseline	Travel Distance(km)	-77.80%		-78.41%	
	Ratio of late delivery		0.17%		0.62%
	Fuel(kg)	-54.61%		-56.14%	
	CO2(kg)	-54.61%		-56.14%	
	CO(g)	-55.01%		-56.42%	
	HC(g)	-54.77%		-56.26%	
	NOx(g)	-55.36%		-56.67%	

Regarding the quality of delivery service, the average additional waiting time due to high demand (i.e., the difference between the actual delivery time and regular ETA) is 38s and 40s in B_COVID_16% and D_COVID_21% scenarios respectively, and the distribution of the time difference is shown in Figure 11. Note that 76% orders are delivered earlier than the regular

ETA. For orders delivered later 10 mins than ETA, we tag them as late delivered order. Table 3 shows the ratio of late delivery is less than 1%.

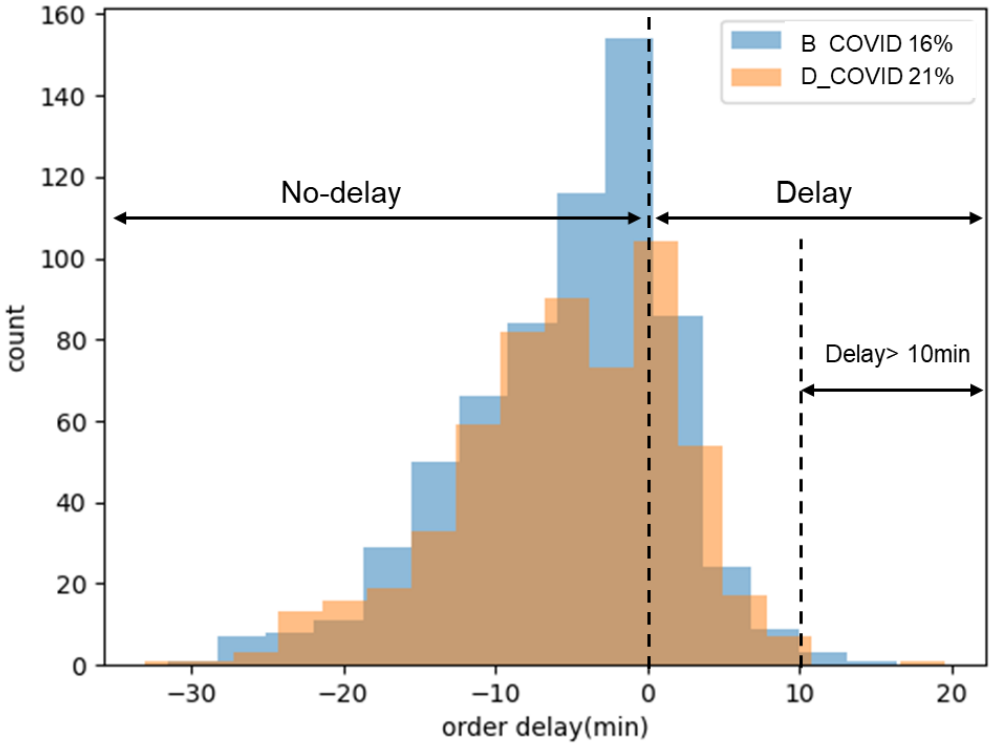

Figure 11. Distribution of time difference between actual delivery time and regular ETA

Table 3 also reveals that during COVID, the total ODFD travel distance reduced to 3009km from 34.27 km in the before-COVID case, around 12% of reduction. The energy and emission also reduced by 12%.

4.3.4 Impact of ODFD penetration rate before and during COVID

In this part, we vary the ODFD service penetration rate and combine all the ODFD trips and in-person trips (without ODFD, individuals should visit the restaurant in-person then return back) together to evaluate all eat-out demand related travel distance in the urban city in multiple scenarios. The results are shown in Figure 12 and Table 4.

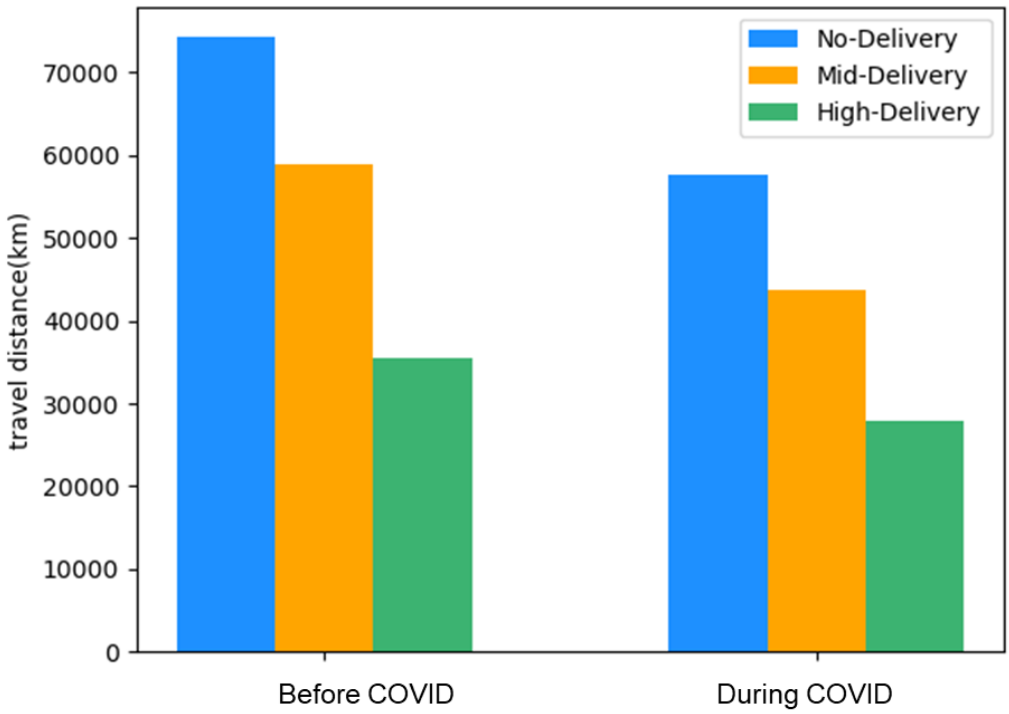

Figure 12. Total travel distance incurred by eat-out demand

Figure 12Figure 12 reveals that in both the before-COVID and during-COVID periods, No-Delivery has the longest travel distance in the traffic network, since all eat-out customers have to make an independent trip to take their food. The total VMT reduces with the increasing penetration rate.

When on-demand food delivery penetration rate increases, the total distance and corresponding fuel consumption and pollutant emission all reduced significantly (in Table 4). Specifically, in the before-COVID case, when the ratio increases from 16% to 50%, total travel distance reduces by 31% bringing fuel consumption and tailpipe emissions reduction all above 22%. In the during-COVID case, the total travel distance can be reduced by 39% if the on-demand food delivery penetration rate increases from 21% to 50%. This can result in saving above 25% of fuel consumption and CO, HC, NO_x emissions. Meanwhile, CO_2 emissions can reduce by 22%. Thus, high penetration rate of on-demand food delivery can help reduce VMT in the traffic network to build an eco-friendly transportation system.

Table 4 also shows the impact of COVID-19 by comparing the total VMT and emissions related to eat-out activities in B_COVID_16% and D_COVID_21% cases. According to the table, all six factors reduced by 25% during COVID comparing with the before-COVID case. This result is as expected with two reasons. First, the total eat-out demand shrinks during COVID which can undoubtedly reduce the total travel distance. Besides, on-demand food delivery ratio increases during COVID as it is a desirable choice for convenient and contactless food delivery under the wide concern of personal health.

Table 4. Result of total eat-out demand in multiple cases

case_name		Total distance(km)	Energy(kg)	CO2(kg)	CO(g)	HC(g)	Nox(g)
B_COVID 0%		74262.48	4594.70	14656.78	5609.02	650.30	4407.17
B_COVID 16%		62251.20	4081.84	13020.81	4973.96	577.29	3899.21
B_COVID 50%		42980.22	3156.22	10068.31	3839.52	446.08	3005.22
D_COVID 0%		57665.80	3545.95	11311.35	4339.96	502.39	3415.43
D_COVID 21%		46739.31	3071.46	9797.75	3752.95	434.87	2949.12
D_COVID 50%		33638.53	2442.19	8060.44	2980.71	345.61	2339.85
B_COVID VS D_COVID	B_COVID 16% VS D_COVID 21%	-24.92%	-24.75%	-24.75%	-24.55%	-24.67%	-24.37%
mid-delivery VS high-delivery	B_COVID 16% VS B_COVID 16%	-30.96%	-22.68%	-22.68%	-22.81%	-22.73%	-22.93%
	D_COVID 21% VS D_COVID 50%	-38.95%	-25.77%	-21.55%	-25.91%	-25.82%	-26.04%

5. On-demand food delivery with dynamic information

In Section 4, we assume that all order and driver information is known beforehand to study the ODFD problem. This setting is too ideal to be practical, since the ODFD system is dynamic: 1) orders might arrive dynamically during the operating period; 2) drivers' shifts start at different time and different initial locations, and 3) the traffic status changes over time. Therefore, the ODFD system should consider the dynamism in the order dispatching and routing process to make better decisions.

In this section, we present a dynamic ODFD system that is efficient to implement in real time, while considering both practical feasibility for the service provider and system benefit for transportation and environment agencies. We then develop a rolling horizon-based approach with adaptive large neighborhood search (ALNS) to solve this ODFD problem, which leverages both the power of ALNS to optimize the large-scale ODFD problem for a given time step and a rolling horizon framework to deal with the system dynamism. The method is capable of handling both multiple-restaurant policy (i.e., orders from different restaurants have a chance to be bundled) and one-restaurant policy (i.e., only orders from the same restaurant can be bundled, so in a single trip the driver can only visit one restaurant).

In a dynamic ODFD system, one order will experience four stages (illustrated in Figure 13. Order life cycle and key time steps

). (1) Finished orders O^d: the order has been delivered before time t_i. This type of order will be removed from the system. (2) Loaded orders O^l: the order has been picked up by the driver but not delivered yet. This type of order must be tied to the assigned driver. (3) Scheduled orders O^s: the order has been assigned to a driver in the previous time interval but not picked up yet. We only allow scheduled orders to be rearranged within the assigned drivers to be optimized to

routing plan, but it cannot be reassigned to a new driver. (4) New orders O^n: new orders arrive in this re-optimized time interval; their placement time falls into the time-interval $[t_i - \tau, t_i]$.

Accordingly, three types of drivers are defined in each time stamp t_i: (1) idle driver K^i: drivers in the system that have not been assigned any orders, including idle drivers from the previous time interval and new drivers arriving in the time-interval $[t_i - \tau, t_i]$. (2) Active driver K^a: the driver that has at least one order and is still able to accept new orders. (3) Full-load drivers K^f: the driver that cannot accept any new orders due to capacity or time limit, i.e., any additional order will cause time window violation (see Section 5.2.1). This type of driver will receive any new orders before at least one of the current orders is completed. When an active or full-load driver completes all the assigned orders, they will become idle again and will wait at the latest drop-off location until a new order arrives.

The life cycle of an order and some key time stamps is illustrated in Figure 13**.** We use lower case t to represent time step and upper case T to denote a certain time period. When a specific order $\boldsymbol{o} \in \boldsymbol{O}$ is placed, the placement time is denoted as $\boldsymbol{t_{po}}$, and before the system nearest update time stamp $\boldsymbol{t_i}$, the order is a new order in $\boldsymbol{O^n}$. After the system re-optimizes, the order will turn to a scheduled order in $\boldsymbol{O^s}$. Then the restaurant will confirm the order and provide a ready pick-up time $\boldsymbol{t_{ro}}$ when the meal is prepared and packed. The driver can arrive at the restaurant earlier but must wait until $\boldsymbol{t_{ro}}$ to pick up the food. Let $\boldsymbol{t_{uo}}$ be the pick-up time of order $\boldsymbol{o}$, after which the order is loaded belonging to the set $\boldsymbol{O^l}$. For each order, we use $\boldsymbol{t_{ko}}$ to denote the drop-off time with the driver *k.* The time difference between the drop-off time and placement time is the click-to-door time ($\boldsymbol{T_{ctd}}$), and the ready to door time is denoted as $\boldsymbol{T_{rtd}}$. In the system, we set a target CtD as $\boldsymbol{\delta}$ to guarantee customers' satisfaction. If the $\boldsymbol{T_{ctd}}$ is greater than $\boldsymbol{\delta}$ then a CtD overage occurs. We set a penalty $\boldsymbol{p_c}$ factor with the corresponding overage amount $\boldsymbol{c = T_{ctd} - \delta}$. If the actual CtD is less than $\boldsymbol{\delta}$, then the penalty $\boldsymbol{p_c}$ is set to be zero. Note that the target CtD here is a measure of delivery performance from the operation perspective. An overage here does not necessarily lead to a late delivery, but a high overage may cause late delivery.

The travel distance and travel time information in this research are derived based on a traffic network in the city of Riverside, CA, which is well-calibrated using GIS database and traffic simulation software. For each link in the network, we can derive the link-length, average link-speed, and average link travel time from the database. Then for each origin-destination (OD) pair, we can create a shortest path, and archive its total travel distance d_{ij} and total travel time t_{ij} into a table, which can be considered as a database when we need to efficiently acquire the trip distance and time to optimize the sequence of each stop in an ODFD problem.

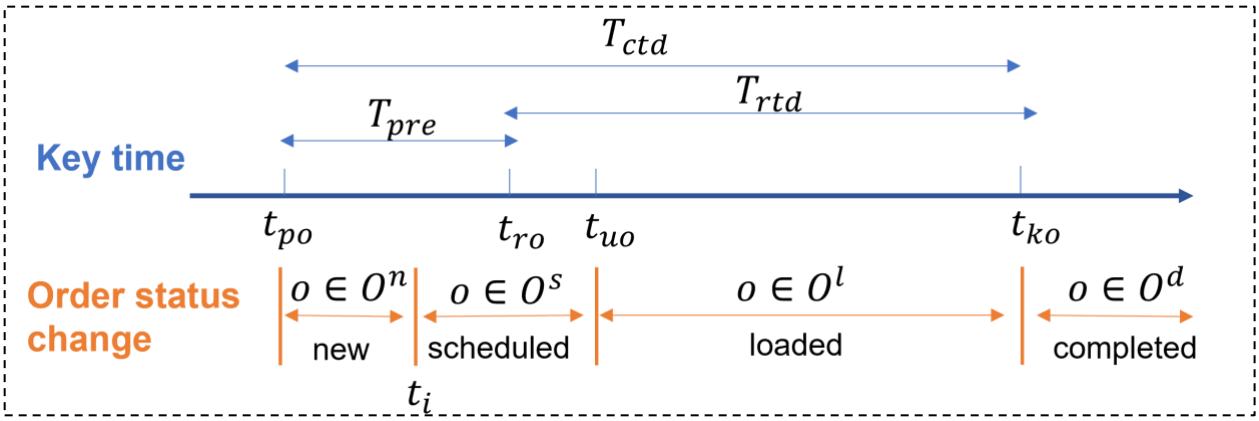

Figure 13. Order life cycle and key time steps

With the above defined types of order and driver, for a given time interval time window *i*, the system has $n = |O^l \cup O^s \cup O^n|$ number of orders and *m* = $|K^i \cup K^a \cup K^f|$ number of drivers. The optimization problem the system needs to solve is to assign all new orders $o \in O^n$ to drivers $k \in (K^i \cup K^a)$, meanwhile finding an optimized solution where the objective increase of the system is minimal. We will formally construct the mathematical model in the next section.

5.1 Dynamic PDPTW model

In this section, we describe a dynamic ODFD system with multiple drivers, multiple restaurants, and dynamic order arrival through a target period, allowing orders and drivers to continuously arrive and leave the system. The goal is to determine feasible routes for drivers to execute pick-up and delivery tasks of orders, with the objective to optimize the system performance in terms of VMT and order delivery time. For a specific time interval, the system needs to solve an ODFD problem with new orders and drivers' information, which can be mathematically formulated as a dynamic PDPTW model. Since we need to consider several types of orders and drivers in the system, new variables have to be defined (summarized in Table 5).

Table 5. Parameters and Variables definition in dynamic setting

Symbol	Description
n	Number of orders. $n = \left\| O^l \cup O^s \cup O^n \right\|$
m	Number of drivers. m = $\left\|K^i \cup K^a \cup K^f\right\|$
R	Set of restaurants.
C	Set of customers.
K	Set of drivers. $K = (K^i \cup K^a \cup K^f)$
K^i	Set of idle drivers which has no order assignment.
K^a	Set of active drivers which at least has one order.
K^f	Set of full-load drivers which cannot further receive new orders.
O	Set of orders in each time interval. $O = (O^l \cup O^s \cup O^n)$
O^d	Set of orders have been finished before the start time of the time interval
O^l	Set of orders have been pick-up by one driver, but not yet delivered.
O^s	Set of orders have been assigned to one drive, but not yet picked-up.
O^n	Set of new orders arrive in the time interval.
Q^k	Capacity of driver k.
q_i	Loads needed to be serviced at node *i.* Positive when i is a pick-up location; negative when i is a delivery location.
V	Set of nodes in the system. ($V_k{}^+ \cup V_k{}^- \cup V_o{}^r \cup V_o{}^c$), for $\forall k \in K, \forall o \in O$
E	Set of arcs in the system, indicating the driver can travel between the initial location, pick-up location, drop-off location and off-location. $E = V \times V$
t_{ij}^k	Travel time of OD pair (i,j) with driver k.
s_i	Service time at node i (load/unload).
t_{po}	The order *o* placement time.
t_{ro}	The order *o* ready-to-pick-up time, which is provide by the restaurant.
t_{uo}	The order *o* pick-up time.
t_{ko}	The order *o* drop-off time with driver *k*
δ	The target click-to-door time the system aims to achieve.
d_{ij}	Travel distance of OD pair (i,j).
Decision variable	x_{ij}^k: Equals to one if driver k visits a OD pair (i,j); zero otherwise Q_i^k: Load of vehicle k when leave node i. T_i^k: Time when vehicel k visit node i.

Each order inherently consists of two locations: restaurant and customer. Each driver has an initial location and final location. Thus, a network representation can be utilized to formally define the meal delivery problem. We first denote K, R, C and O as the set of drivers, restaurants, customers, and orders respectively. Each driver $k \in K$ is characterized by a 5-tuple $(e_k, l_k, V_k{}^+, V_k{}^-, Q_k)$, where e_k is the driver's scheduled on-time when the driver is available to work, l_k is the driver's off-time, $V_k{}^+$ is the on-location, $V_k{}^-$ is the off-location, and Q_k is the delivery capacity denoting the maximum order number the driver can receive. For an order $o \in O$ placed by a customer $c \in C$ from restaurant $r \in R$, a 4-tuple $(t_{po}, t_{ro}, V_o{}^r, V_o{}^c)$ is formulated where t_{po} is the order placement time, t_{ro} is the order ready-to-pick-up time, $V_o{}^r$ is the restaurant location, and $V_o{}^c$ is the customer location. Then an undirected graph *G* = (*V*, *E*) can

be constructed with the set of nodes V = ($V_k^+ \cup V_k^- \cup V_o^r \cup V_o^c$), for $\forall k \in K$, $\forall o \in O$, and the set of arcs $E = V \times V$.Then a dynamic PDPTW model can be formulated:

$$\min F = \alpha \sum_{k \in K} \sum_{(i,j) \in E} d_{ij} x_{ij}^k + \beta \sum_{o \in O} p_c (T_{ctd} - \delta) \qquad (0)$$

$$\sum_{j \in V} x_{V_k^+, j}^k = 1 \qquad \forall k \in K \qquad (1)$$

$$\sum_{i \in V} x_{i, V_k^-}^k = 1 \qquad \forall k \in K \qquad (2)$$

$$T_{V_k^+}^k \geq e_k \qquad \forall k \in K \qquad (3)$$

$$\sum_{i \in V} x_{ij}^k - \sum_{i \in V} x_{ji}^k = 0 \qquad \forall j \in (V_o^r \cup V_o^c, for \ \forall o \in O), \forall k \in K \qquad (4)$$

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k = 1 \qquad \forall i \in (V_o^r \cup V_o^c, for \ \forall o \in O) \qquad (5)$$

$$\sum_{j \in V} x_{V_o^r, j}^k - \sum_{j \in V} x_{j, V_o^c}^k = 0 \qquad \forall o \in O, \forall k \in K \qquad (6)$$

$$x_{ij}^k = 1 \Rightarrow Q_j^k \geq Q_i^k + q_j \qquad \forall (i,j) \in E, \forall k \in K \qquad (7)$$

$$Q_i^k \leq Q^k \qquad \forall i \in V, \forall k \in K \qquad (8)$$

$$x_{ij}^k = 1 \Rightarrow T_j^k \geq T_i^k + t_{ij} + s_i \quad \forall (i,j) \in E, \forall k \in K \qquad (9)$$

$$t_{uo} \geq \max (t_{ro}, T_i^k) \qquad \forall o \in O, \forall k \in K \qquad (10)$$

$$T_{V_o^r}^k \leq T_{V_o^c}^k \qquad \forall o \in O, \forall k \in K \qquad (11)$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall (i,j) \in E, \forall k \in K \qquad (12)$$

$$Q_i^k \geq 0 \qquad \forall i \in V, \forall k \in K \qquad (13)$$

$$T_i^k \geq 0 \qquad \forall i \in V, \forall k \in K \qquad (14)$$

The objective function aims to minimize the weighted sum of total travel distance of all on-demand delivery drivers and the penalty of potential CtD overage occurring to orders. α and β are weight factors designed to balance the distance cost and late delivery penalty. The second part of the penalty function is a linear piecewise function, e.g., low penalty factor for slight overage, and high penalty factor for late delivery. The setting of p_c depends on the willingness of the system to accept late deliveries. The target CtD δ can also be customized with a function to consider the order characteristics, such as peak hour factor, distance between the pick-up and drop-off location, customer preference, meal type (cold food or cooked food), etc.

Constraint (1)-(3) states the driver working schedule requirement. Constraints (1) and (2) together guarantee that each driver will start to work from their initial location and end at the

final location. The last terminated location can be the driver's living place or the last drop-off location. Constraint (3) indicates that the driver cannot start to work before time e_k (the beginning of their working schedule). Constraint (4) governs the flow conservation to ensure the route feasibility. Constraint (5) and Constraint (6) satisfy the delivery requirement. Constraint (5) ensures that for every order, its pick-up and drop off location should be visited exactly once. Constraint (6) further requires the same driver to serve at the pick-up and drop-off location of one order. If the order has already loaded on one driver, then this driver must visit the corresponding customer location to drop-off the meal. Driver capacity is addressed in Constraint (7) and (8). Specifically, Constraint (7) states that when the driver travels from location i to location j, the meal loaded when driver leaving location j is equal to the meal loaded when driver leaving location i plus the additional load/unload in location j. If location j is a pick-up location, $q_i > 0$. Otherwise, $q_i < 0$ indicating the driver drop-off meal at the customer location. Constraint (9)- (11) establish the time relationships. Constraint (9) states that if a driver k travels from i to j, the earliest time the diver can arrive at location j is equal to the arrival time at location i plus the service time s_i at location i plus the travel time t_{ij} from i to j. This constraint defines every node arrival time in the system, except the driver initial location visit time which we define with Constraint (3). Constraint (10) allows the driver to pick up an item from the restaurant no earlier than order ready-to-pick-up time t_{ro}. However, the driver can arrive the restaurant earlier and wait until t_{ro}. Thus, the order pick-up time t_{uo} is the maximum of order ready-to-pick-up time and the driver arrival time to the restaurant. Constraint (11) requires that the item be collected from a restaurant before drop-off at the customer location. Finally, Constraints (12)-(14) are the definitions of decision variables as shown in Table 5.

5.2 A Rolling Horizon Optimization Approach with ALNS

The ODFD system is highly dynamic; in essence, planning all orders before receiving the order information is unrealistic. For this case, we introduce a rolling horizon algorithm with ALNS optimization framework where the system triggers a re-optimization process every τ minutes. In our route planning, any orders have the chance to be bundled into one driver's route if the bundle can decrease the objective value compared with other assignment options. Drivers are heterogenous with unique shift plan, initial location, and off-location. At the optimization time t, the algorithm determines the best assignment of new orders. For every re-optimized process, we have the following assumptions.

5.2.1 Re-optimization assumption and structure

In the dynamic ODFD system, the set of overall system state and components such as order information are changing dynamically, and we need to serve the new arriving demand. For this update and re-optimization process, we have the following assumptions.

Assumption 1. Idle driver will wait at the on-location or latest drop-off location. The drivers who haven't been assigned any orders should wait at their on-location. The drivers who return to idle state after completing the latest order should wait at the last drop-off location.

Assumption 2. Driver cannot be diverted when they are on the way to a restaurant or a customer. Drivers should complete the on-going task then adjust delivery plan with respect to new routing information in the new time window.

Assumption 3. The pre-assigned loaded orders and scheduled orders in the previous time interval should be tied with the same assigned driver.

Assumption 4. Full-load drivers that meet the time window limit cannot accept new orders until they finish all current assigned orders.

With these necessary assumptions, the system will perform the following updates. First, for the system efficiency purpose, any completed orders should be removed from set O (set of orders to be re-optimized) and put into the set O^d to record the distance cost and CtD overage cost of the order. The corresponding ${V_o}^r$ and ${V_o}^c$ of $o \in O^d$ will also be removed from the undirected graph G. Similarly, for orders already picked-up, the system will tag the order as $O^l$and remove only the pick-up location ${V_o}^r$. For orders that have not been picked up, the system will tag the order as O^s and keep both ${V_o}^r$ and ${V_o}^c$ of each order. After this update procedure, the set of order and nodes will have been updated to unserved tasks and unvisited nodes. The driver assignments of remaining tasks and status will also be updated with respect to the order status. Next, the system will incorporate new information to construct a complete order set $O^l \cup O^s \cup O^n$ and a complete driver set for re-optimization.

5.2.2 Algorithm framework

The framework of our proposed algorithm is depicted in Figure 14. This algorithm is a time-based periodic approach which divides the total operational horizon H into a number of time intervals of length τ. For every predefined time step $t_i = i \times \tau$ for $i = 0,1,2, \dots \left\lceil \frac{H}{\tau} \right\rceil$, the simulation part of the framework will update the driver and order status up to t_i of the solution in the last time interval and reconstruct the input of the system. The order input size is $n = |O^l \cup O^s \cup O^n|$ and driver input size is $m = \left|K^i \cup K^a \cup K^f\right|$. The size of the input depends on the order arrival pattern and the length of time interval τ. Then the system will start the re-optimization process to solve any new orders accumulated during time interval $[t_i - \tau, t_i]$ and construct a routing plan for drivers for this time interval.

With the new input, the system will re-construct an initial solution with a tailored driver matching strategy and use the ALNS algorithm to search the further optimized neighborhoods to improve the initial solution. In our setting, loaded orders are not allowed to be transferred to another driver because pick-up and drop-off task of one order should be finished by a single driver is a hard constraint in our model. Besides, scheduled orders are not allowed to be rescheduled. The pre-assigned loaded orders and scheduled orders should be tied with the assigned driver, but we can perform an intra-route operator to search for a better visit sequence. Thus, in ALNS the optimization is mainly performed between the new orders to search for the best assignments to the drivers in $K^i \cup K^a$. With the optimized solution from ALNS, the driver will receive updated order assignments and routing plan based on which to do

the pick-up and delivery task in time $[t_i, t_i + \tau]$. This workflow will keep updating until reaching the last time interval of the operational horizon.

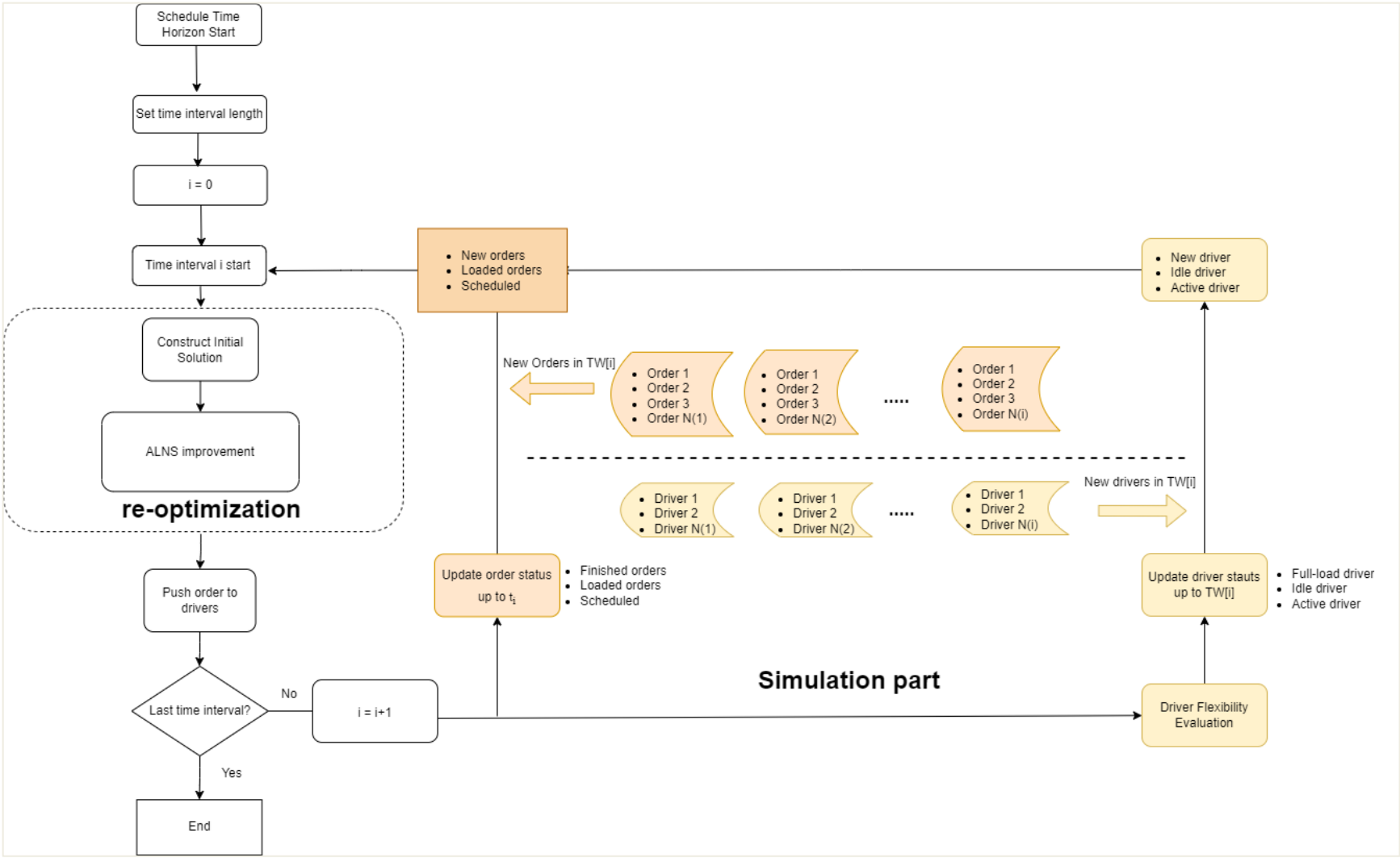

Figure 14. Rolling Horizon algorithm with ALNS framework

5.2.3 Driver Flexibility evaluation

Instead of only encouraging drivers to drop off the meal as early as possible, we also tend to consider the system level travel distance and vehicle utilization. If the driver has some flexible time to pick up a new order along the way to deliver other orders and this additional pick-up task won't cause serious late delivery of the previous assignment, then our algorithm will bundle this new order to this type of driver.

One tool to check the flexibility of the driver route is the Forward Time Slack (FTS), which was originally proposed to solve the TSPTW in [29]. The main idea is that given a feasible route, the feasibility of every visiting location is still maintained if the service time of some locations is delayed to some extent. This optional and voluntary delayed amount is the FTS of the location. Later Cordeau and Laporte extended the FTS to evaluate neighborhoods in solving the multi-vehicle dial-a-ride problem [30]. Gschwind and Drexl integrated FTS to propose a feasibility test for a route to evaluate the request insertion in the repair step of ALNS with constant time complexity [31].

In our on-demand meal delivery problem, we set soft time window of the exact pick-up and drop-off time, so the route is still considered feasible if some amount of time window violation occurs. The reason of this setting is because the congested traffic and high demand under peak hour may make some orders not possible to be delivered on-time. Similar to the Cordeau and Laporte approach in solving the dial-a-ride problem, we also slightly modified the FTS definition to be the largest increase in the beginning of the service at vertex V_i that will not cause any increase in the time window violation of other locations in the route. Mathematically, we can define the forward time slack F_i of each node in one route as follows. Consider a particular route $(V_1, V_2, V_3, \dots, V_i, \dots, V_q)$, where V_1 indicates the starting location, V_q indicates the ending location, and any V_i for $i \in (2, q)$ can represent either pick-up or drop-off location of the meal order.

$$F_i = \min_{i \le j \le q} \left(\sum_{i \le p \le j} W_p + \left(l_j - B_j \right) \right) \qquad (15)$$

Where W_p denotes the waiting time at node p, l_j is the soft time window upper bound location j, and B_j is the start service time at location j. In our problem setting, W_p and $l_j - B_j$ can be specifically defined for the pick-up and drop-off task as follows.

$$W_p = \begin{cases} \max\left(0, t_{ro} - T^k_{V_o{}^r}\right) & p = V_o{}^r \ of \ order \ o \\ 0 & p = V_o{}^c \ of \ order \ o \end{cases} \qquad (16)$$

$$l_j - B_j = \begin{cases} 0 & j = V_o{}^r of \ order \ o \\ \max\left(0, \ \delta - (t_{ko} - t_{po})\right) & j = V_o{}^c \ of \ order \ o \end{cases} \qquad (17)$$

The equation (15) states that F_i is equal to minimal FTS of any location j after location i; each FTS of j is calculated as the accumulated waiting time and the time difference between the latest service time l_j and starting service B_j. In our case, we set the ready-to-pick-up time for pick-up tasks t_{ro}, if the driver arrives earlier than t_{ro}, then the driver should wait at the

restaurant. We assume that drivers will drop off the order as soon as they arrive at the customer location. This is shown in equation (16). Similarly, in equation (17), the latest service time of a pick-up task is exactly the start service time. For a drop-off node, if the target CtD(δ) is larger than the actual CtD ($t_{ko} - t_{po}$), then this amount of time difference contributes to the FTS. Based on the FTS of each node in one route, we take the largest possible FTS on the route to determine the route flexibility (18). If R_f is larger than zero, then some locations still have slack to insert a new task before it. If R_f is less than zero, then the driver executing route will be tagged as full-load driver.

$$R_f = \max_{0 \le i \le q} F_i \qquad (18)$$

More details are shown in the pseudocode of the driver flexibility evaluation algorithm. Specifically, the inner for-loop between line 4 to line 18 computes the FTS of each node of the routes. The outer for-loop summarizes the F_i of each node and return the largest possible FTS of the route based on which to decide the status of driver. With the driver flexibility evaluation, the system can avoid inserting new orders into the route that may cause serious CtD overage of the pre-scheduled orders. Besides, we can put the full-load drivers aside and reduce the computation complexity in the ALNS process.

Algorithm 2. Driver Flexibility Evaluation	
Input:	Driver k, assigned route AR = $(V_1, V_2, V_3, \dots, V_i, \dots, V_q)$
Output:	R_f
1:	Initialize a list F to record F_i for each node
2:	**for** node V_i in *AR* **do**
3:	Initial a list S
4:	**for** j in (0, *AR*[i :]) **do**
5:	p ← i
6:	wait_time ←0
7:	**while** p ≤ j **do**
8:	**if** $V_p = {V_o}^r \ of \ order \ o$:# if V_p is a pick-up location, get waiting time
9:	wait_time += max (0, $t_{ro} - T^k_{V_o r}$)
10:	**end if**
11:	**end while**
12:	ahead_time ←0
13:	**if** $j = {V_o}^c \ of \ order \ o$: # if V_j is a drop-off location, get ahead time w.r.t δ
14:	ahead_time = $\max(0, \ \delta - (t_{ko} - t_{po}))$
15:	**end if**
16:	S. append (wait-time + ahead_time)
17:	**end for**
18:	F. append(min(S))
19:	**end for**
20:	R_f = max(F)
21:	return R_f

5.2.3 Construct initial solution

For the new orders arriving in the system, we will first sort them in ascending order according to the target drop-off time. Then each order will be assigned to one driver to construct an initial solution. The driver matching strategy is the nearest driver with priority. The priority here is that we first look for the nearest idle driver. If there is no idle driver, we will start searching for the nearest active driver for the order (as shown in Algorithm 3). With this strategy, when the system has more drivers than orders, we intend to encourage single bundle. This will help the system to warm-up and reduce the CtD overage penalty as much as possible.

For the idle driver, the system calculates the distance between the driver initial location and the restaurant location and then get the nearest driver (line 5). For the active driver, the system will instead calculate the distance between the driver last drop-off location and the restaurant (line 6). After finding the nearest driver *k*, order *o* will be inserted into the best position that causes minimum objective value increase (line 9). If an idle driver has been assigned an order, then we should update it into the active driver set K^a (line 10).

Algorithm 3. Nearest Driver with Priority

Input: New orders O^n, driver set $K^i \cup K^a$
Output: The initial assignment of each order to a specific driver
Priority Queue L← Sort Order by target drop-off time (O^n)
Pop out order sequentially
for all order *o*∈ O^n **do**
  if $|K^i| > 0$:
    find nearest driver *k* in K^i
  else:
    find nearest driver *k* in K^a
  end if
  find best position in driver *k* to insert order *o*
  update driver *k* to K^a if *k* is chosen from K^i
end for

5.2.4 ALNS for dynamic ODFD

To further improve the initial solution, we propose to use an Adaptive Large Neighborhood Search (ALNS) algorithm as in the static ODFD problem. The difference is that in the dynamic version of meal delivery problem, more specific requirements need to be carefully set up in the ALNS removal and repair process to ensure the solution satisfies our assumptions in Section 5.2.1.

First, as in assumption 3, transferring the previous assigned orders to another driver is not allowed, so in the ALNS removal process, only the new orders can be removed. Second, in assumption 2 we only allow drivers to change routes until reaching a task location, either pick-up or drop-off. Thus, in the ALNS repair process, no task can be inserted earlier than the on-going task. Meanwhile, after inserting the new orders, an intra-route search after the on-going task will be executed between all orders assigned to the driver.

A concrete example is shown in Figure 15. The green circle is the drop-off task and the blue rectangle is the pick-up task. In the pre-assigned route, at this time step, the driver is en route to deliver order 1 (on-going task). Order 2 is loaded. Orders 3 and 4 are scheduled. Then a new order, order 5, needs to be inserted. The algorithm will search for the best position to insert pick-up and drop-off task of order 5 and perform the intra-route operator [32]. One possible new route is depicted, in which the order 5 is successfully inserted and the task sequences between order 3 and order 4 are also modified to gain a better system performance.

With the above specific set up in the removal and repair process of ALNS, we can apply the same destroy and repair operators as described in Section 4.2.2.

5.2.5 EMFAC model for environmental impact evaluation

EMFAC model is utilized to evaluate the environmental impact of eat-out incurred traffic in the system. EMFAC 2021 is the latest version of the California mobile source emissions model,

which is an officially approved model providing both on-road and off-road emission rate of multiple types of vehicles [33]. Here we use the EMFAC model to gain the energy consumption and emission rate of the passenger car (with gasoline). Because we assume the on-demand delivery drivers will use their private cars to do the delivery. The following factors are considered: Fuel consumption (gasoline), Green House Gas (including CO_2, CH_4, N_2O), CO, PM 2.5, and NO_X.

Assume f_{ij} is the fuel consumption/emission rate of link *ij*, which indicates fuel consumption/ emission per unit distance. Then fuel consumption/emission of link *ij* is $fc_{ij} = f_{ij} \times L_{ij}$, where L_{ij} is the length of link *ij* (mile). We sum up the energy consumption/emission of the delivery vehicles travelling on all the links in the network to evaluate its impact.

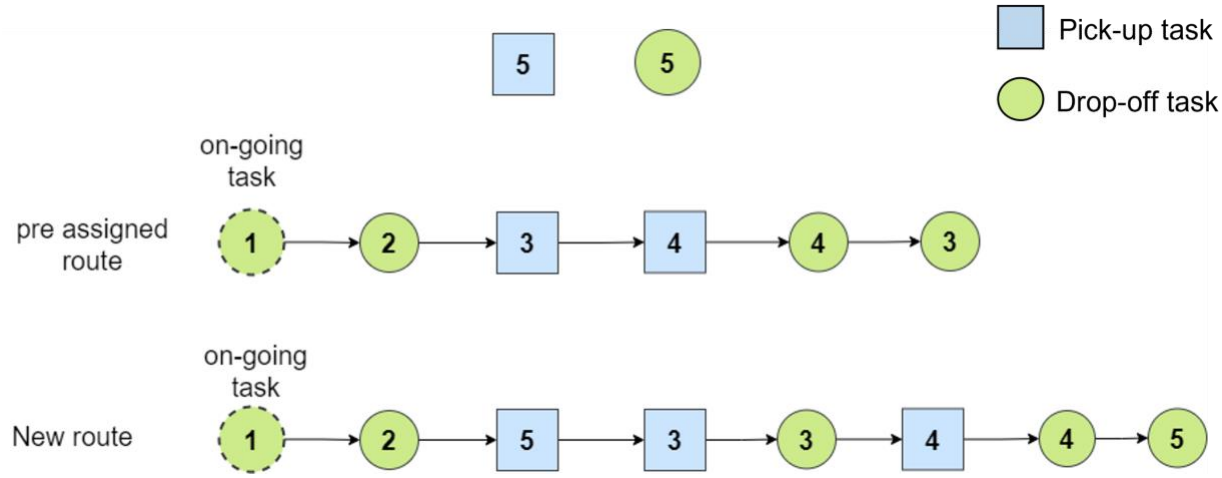

Figure 15. Repairing process of ALNS

5.3 Numerical experiment

The big picture of the simulation study is shown in Figure 16. Simulation study workflow

. For a CEMDAP model, given the inputs as various land-use, sociodemographic, activity, and transportation level-of-service attributes, it provides the outputs of complete daily activity-trip patterns for each individual in each household of a population [22]. Among all activities, we mainly focus on the eat-out trips related to our meal delivery problem. Based on the delivery ratio, the ODFD orders are created by converting part of eat-out activities into ODFD requests. On the other hand, a BEAM model is utilized to obtain the traffic information [34]. The delivery trip only accounts for a small number of traffic, which has little impact on the background traffic. We obtained the traffic information from BEAM at 12:00 pm and assume this is the noon traffic status. All the ODFD drivers will also be generated in the network. With all orders, drivers, and traffic information, we use the above proposed method: A rolling horizon framework with ALNS, to efficiently dispatch meal orders to best drivers in the system and gain the best routing plan simultaneously. With the order assignments result, we can evaluate the algorithm performance in 1) operational view: customer experience related factors; 2) in traffic impact view: VMT cost; and 3) in environmental view: fuel consumption and emissions in

EMFAC model [35]. In our case, the on-demand meal delivery simulation environment was developed in Python 3.8. All experiments are run with ThinkPadX1 Carbon 2021 with 16GB of RAM and an Intel Core I7- 1165G7 processor.

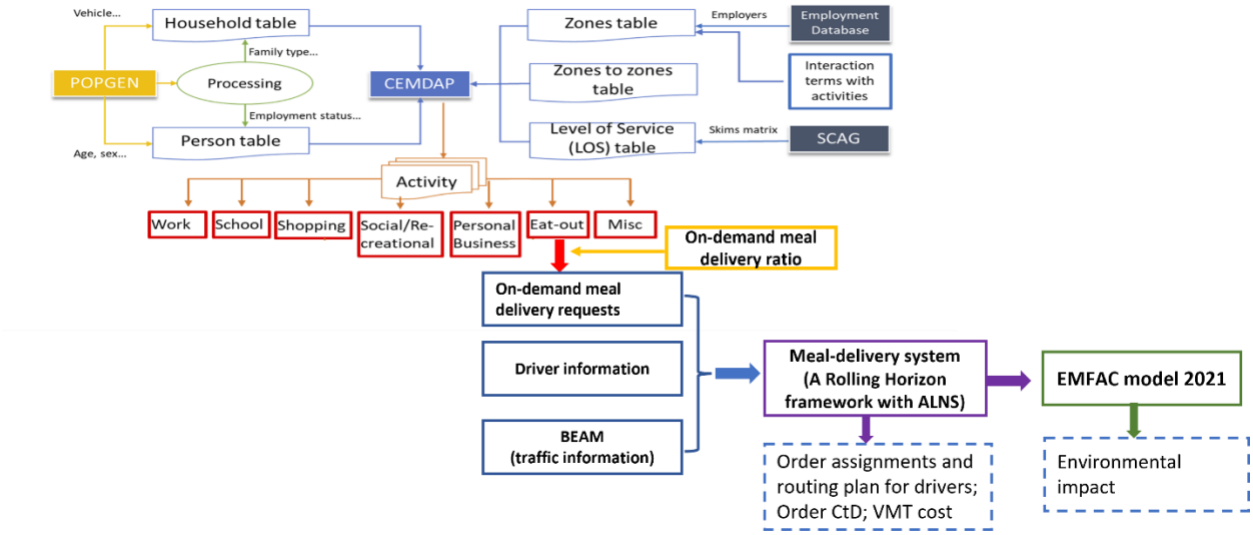

Figure 16. Simulation study workflow

5.3.1 ODFD orders description

For the dynamic ODFD, we generated a new dataset with 1328 eat-out trips spanning from 11:30am-12:30pm in total. The survey data from [6] shows the change in on-demand food delivery penetration share in the U.S reach 14% in 2021, and is expected to reach 21% in 2025 considering the short-term and long-term impact of COVID-19. Based on the near future estimate in 2025, we randomly sample 278 trips to be the delivery orders, each order information contains order placement time, restaurant location, and customer location. For the number of drivers, we assign new drivers with 5 as the order/driver ratio. Meanwhile we set certain amounts of idle drivers available in the first time interval to provide the system more flexibilities of searching for nearest drivers. The driver's shift start time is based on the order intensity during the horizon.

Table 6 summarizes the key sampled order statistics. We have generated 66 drivers distributing in the city. In total this sampled dataset contains 141 unique customers and 77 unique restaurants, which means we have covered both cases: 1) one restaurant can receive multiple orders and 2) one customer (or multiple customers from the same address) can place multiple orders at the same or different time. For case two, we treat the orders separately to let the algorithm decide whether to bundle or not, instead of posing a hard non-split policy as in [17]. The average distance between the customer and restaurant per order is 8.59 km.

Table 6. Sampled orders information

Case	# of orders/trips	# of drivers	# of unique customer locations	# of unique restaurant locations	Avg_dis (km)
On-demand orders	278	66	141	77	8.59

5.3.2 Experiment setup

After the customer places one order, similar in Section 4, we assume the restaurant needs 5-10 mins to prepare for it, which defines the order ready pick-up time. The driver is not able to pick up earlier than ready time. We set the default customer target click to door time (CtD) value to be 40 minutes, which indicates that the platform aims to finish each order within 40 minutes. Drivers need 1 extra minute to pick up or deliver when arriving the location. Further, we assume each driver can be assigned no more than 10 orders in one working schedule. Driver travel time estimation is the same as in Section 4.3.2. In the objective function, based on multiple experiment trials, we set the weight α and β both equal to one. And the lateness penalty is set to be 1, 5, and 10 (if CtD overage< 5min, then $p_c = 1$;if CtD overage > 10min, then $p_c = 10$; otherwise $p_c = 5$). Parameter setting in ALNS algorithm is the same as in the static problem (Section 4).

Two ODFD policies are proposed in this project. In a strategy inspired by [16], only orders from the same restaurant can be bundled to one driver. Thus, one driver can only visit one restaurant to pick up one or more orders then carry out the delivery task. This policy is named as One-Restaurant (One-R) policy. Another strategy is Multi-Restaurant (Multi-R) policy which allows one driver to visit multiple restaurants to pick up food orders nearby then sequentially deliver them to the customers. We also consider a simple baseline strategy, in which all customers select to dine-in or pick up the meals by driving themselves.

Table 7 shows the experimental setup. The default case is Scenario 0, in which the system update frequency is 5 minutes, the target CtD is fixed with 40 minutes, and the bundle policy is Multi-R. In Scenario 1 and 2, we test One-R and baseline policy respectively while keeping all other parameters the same to ensure a fair comparation of different delivery policies (See 5.4.1). The second experiment group varies CtD for each order considering peak-hour traffic or long distance trip (See 5.4.2). Then we vary the system update frequency to study the impact of system dynamics (See 5.4.3). Finally, we change the penetration rate of on-demand meal delivery and combine all dine-in and delivery together to study the role of on-demand meal delivery at the city-level (See 5.4.4).

Table 7. Experimental scenario setup

Scenario id	Descriptions
0	Default (τ = 5, fixed CtD = 40 minutes, policy= Multi-R
1	τ = 5, fixed CtD = 40 minutes, policy = One-R
2	τ = 5, fixed CtD = 40 minutes, policy = self-pick-up
3	τ = 5, customized CtD, policy = Multi-R
4	τ = 5, customized CtD, policy = One-R
5	τ = 1, fixed CtD = 40 minutes, policy = Multi-R
6	τ = 5, fixed CtD = 40 minutes, policy = Multi-R
7	τ = 10, fixed CtD = 40 minutes, policy = Multi-R
8	τ = 15, fixed CtD = 40 minutes, policy = Multi-R
All eat-out demand impact (include dine-in and on-demand delivery)	
9	Penetration rate = 0%, 14%, 21%, 40% τ = 5, fixed CtD = 40 minutes, policy= Multi-R
10	Penetration rate = 0%, 14%, 21%, 40% τ = 5, fixed CtD = 40 minutes, policy= One-R

5.4 Results analysis

5.4.1 Effects of delivery policy (Scenario 0-2)

In this section, we investigate three different policies of food delivery: baseline, One-R, and Multi-R. The total VMT, fuels and emissions are summarized in Table 8. Compared with the baseline case, the One-R policy can reduce the VMT by 23% and meanwhile save over 21% of fuel consumption and tailpipe emissions (including GHG, CO, PM2. 5, NO_x). The Multi-R policy also outperforms the baseline case, showing 43% reduction of VMT and emissions. The improvements brought by ODFD are as expected, since orders in the commercial zone are coordinated and bundled to one driver to reduce the redundant trips between restaurants and communities. Regarding ODFD policies, Multi-R policy achieves around 29% of VMT, fuel and emission savings comparing with One-R policy, as Multi-R has higher flexibility to group orders, especially from restaurants located densely in one shopping plaza or commercial zone. It is also noticed that very similar reduction rate can be found in VMT, fuel, and emissions. The main reason is that in EMFAC model, VMT and speed pattern are two key inputs for fuel and emission estimation. As the objective function only aims to minimize the VMT, the speed pattern is neither optimized nor influenced. Thus, the percentage change in fuel and emissions are similar to the percentage change in VMT when switching from one policy to another.

The CtD and RtD distributions of all food orders are shown in Figure 17. The average RtD and CtD of One-R policy are 16.88 min and 24.39 min, while these two values for Multi-R are 26.06 min and 33.56 min. One-R policy tends to deliver food orders earlier than the Multi-R policy as drivers in One-R policy are likely to have a delivery trip with less orders bundled. Both policies can meet the customers' satisfaction that only less than 3% orders be delivered later than 10 minutes of the target drop-off time.

Table 8. Comparison of different scenarios

	self-pick-up	One-R	Multi-R
Number of delivery order	0	278	278
Number of self pick-up order	278	0	0
%of total eat out demand	21%	21%	21%
Travel Distance(km)	4777.32	3681.46	2613.17
Ratio of late delivery	0.00	2.51%	1.00%
Fuel(gallon)	106.80	82.47	58.64
GHG(kg)	918.86	709.49	504.48
CO(g)	2918.95	2292.25	1644.72
PM2.5(g)	4.06	3.19	2.28
NOx(g)	170.00	131.62	93.61
Comparision	**One-R VS self-pick-up**	**Multi-R vs self-pick-up**	**Multi-R vs One-R**
Travel Distance(km)	-22.94%	-45.30%	-29.02%
Ratio of late delivery	2.51%	1.00%	1.00%
Fuel(gallon)	-22.78%	-45.09%	-28.90%
GHG(kg)	-22.79%	-45.10%	-28.90%
CO(g)	-21.47%	-43.65%	-28.25%
PM2.5(g)	-21.43%	-43.84%	-28.53%
NOx(g)	-22.58%	-44.94%	-28.88%

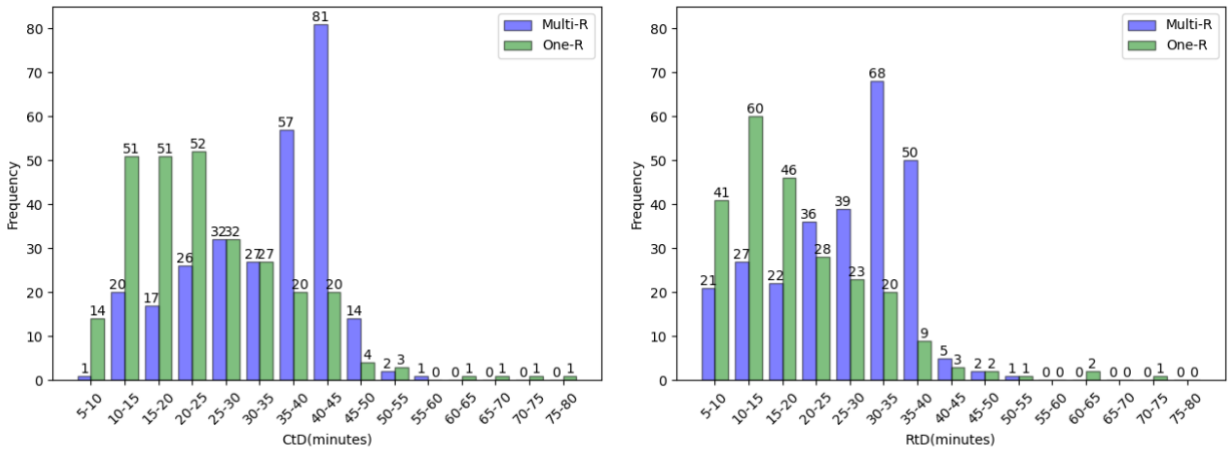

Figure 17. CtD (Click to door time) and RtD (ready to door time) distribution of 278 orders

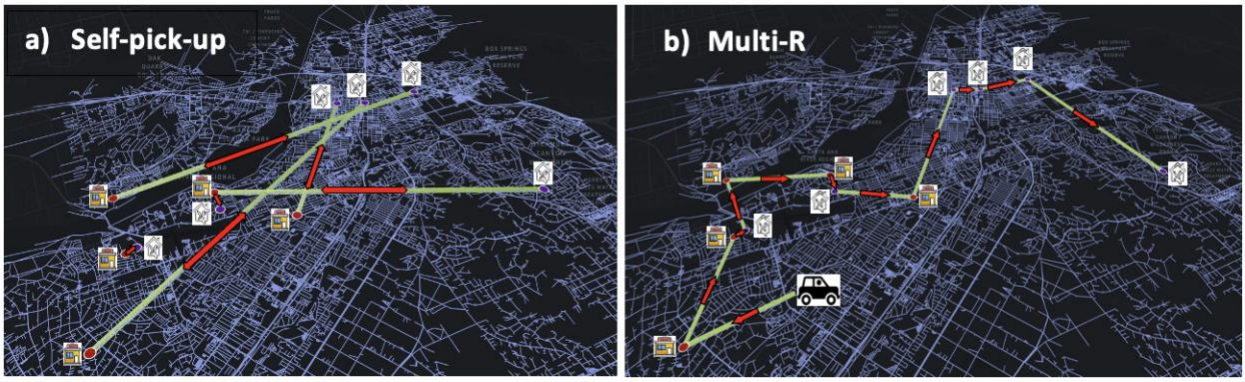

Figure 18. Example of one driver route in Multi-R policy

Specifically, we plot one example of routing result (Figure 18) to show the capability of Multi-R policy in saving VMT. In this example, we have six total orders. In the baseline case, all customers should have a round-trip to the restaurant, with the total VMT of 99 km. Instead, with the Multi-R policy, the platform can assign one driver nearby to pick up all the food orders and then deliver to all customers sequentially. In this case, the Multi-R policy can save up to 50% VMT and corresponding emissions, while easing the burden of traffic congestion and parking in the city. For One-R policy, it is complicated to exactly plot these six orders routing result because it also involves other orders that originated from the same restaurant. But in One-R policy, we need to assign five drivers to finish these six orders and it can only save around 22% VMT compared to the baseline.

5.4.2 Effects of Click-to-Door time setting (Scenario 3 and 4)

In practice, the platform can customize the target CtD for each order to accommodate unavoidable lateness due to peak-hour traffic or long restaurant-customer distance. Therefore, we add additional buffer time (5 min) to the order placed between 12:00 pm – 12:20 pm and extra buffer time if the customer place is more than 10 km away from the restaurant location. Table 9 presents the results regarding the performance of tailored CtD. With customized CtD, the meal-delivery cost (VMT, energy consumption, and emission) can be further reduced by around 16% and 6% in Multi-R and One-R policy respectively. This is because for some "tough" orders, the system can have more time to handle and can bundle more orders together. Figure 19 reveals that the average CtD and RtD increase by around 7% (Multi-R) and 2% (One-R) as more orders are bundled together.

Due to extended target CtD, the CtD increasing does not worsen the CtD overage. From the boxplot in Figure 20, we can figure out that with customized CtD, the system suffers from less order lateness in both Multi-R and One-R policy. Compared with One-R policy, one can note that Multi-R policy can provide the routing plan with less variance in delivery lateness and without extreme CtD overage, which shows the robustness of Multi-R policy.

Table 9. Comparison of different CtD setting

Metric	Multi-R		One-R	
	Fixed	Customized	Fixed	Customized
Distance(km)	2613.17	2244.07	3681.46	3467.06
Avg_CtD(min)	33.56	36.11	24.39	24.77
Avg_RtD(min)	26.06	28.60	16.89	17.26
Energy(Gallon)	58.63	50.48	82.47	77.75
GHG (kg)	504.48	434.32	709.49	668.93
CO(g)	1644.72	1421.69	2292.25	2166.35
PM2.5(g)	2.28	1.98	3.19	3.02
NO_x(g)	93.61	80.69	131.62	124.13

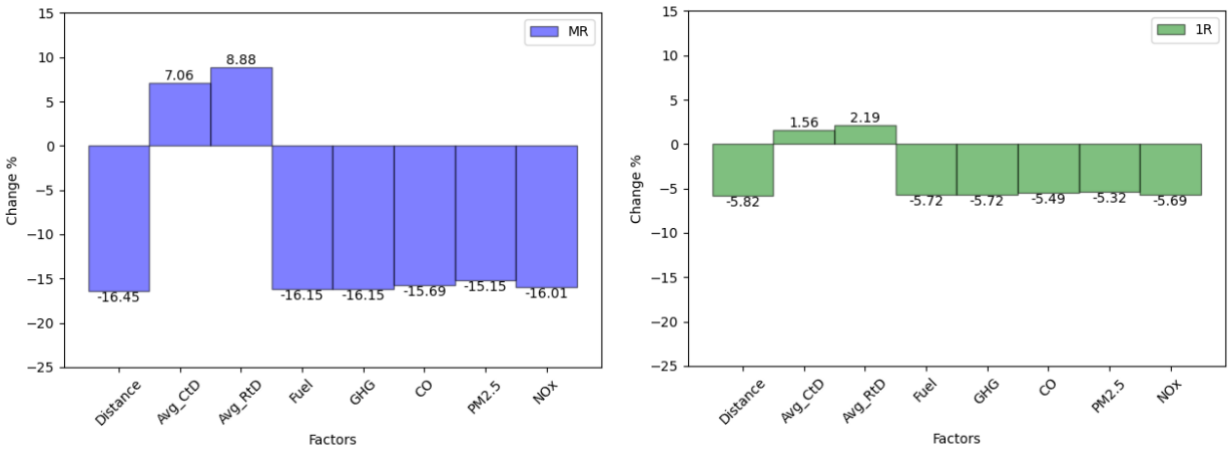

Figure 19. Metric change compared to fixed CtD. Left: Multi-R; right: One-R

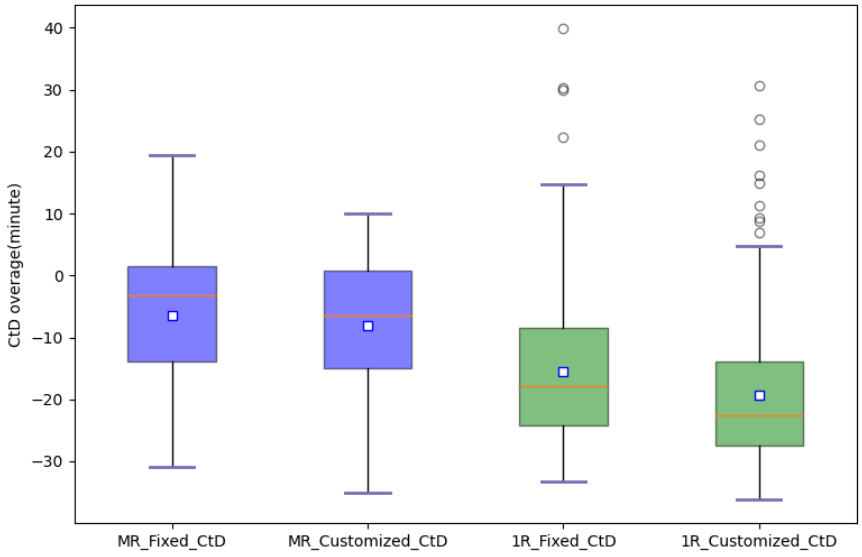

Figure 20. CtD overage distributions

5.4.3 Effects of system update frequency (Scenario 5-8)

The time window length controls the system update frequency. The smaller the time window, the higher frequency the system updates. In this part, we vary the time window length from 1 min to 5 min, 10 min, and 15 min. The obtained result is shown in Figure 21 and Table 10. The results illustrate that with lower system update frequency, the system can reduce more VMT and emissions as the delivery plan for each order can be better coordinated when the system process more orders at one time. The VMT cost can be reduced from 2946 km to 2509 km when the time window length increases from 1 min to 15 min. The average CtD and RtD also decrease, indicating customers can receive food orders earlier. More detailed energy savings and emissions reduction are listed in Table 10. However, the average computational time for

each time interval increases from 1.89 sec to 149.77 sec, as the solution space grows exponentially when more orders are collected, requiring more computational time to find the optimal solution.

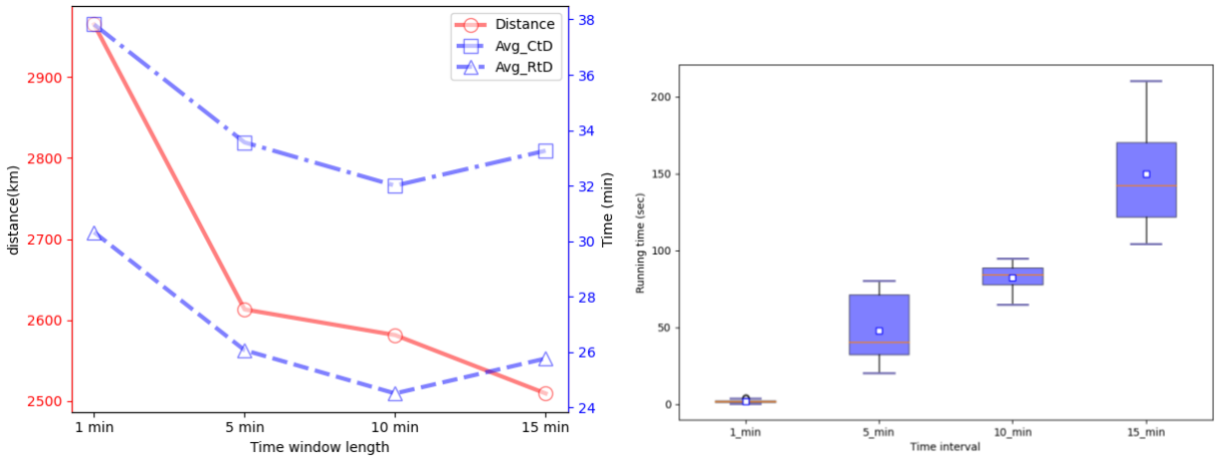

Figure 21. Effect of time window length

Table 10. Results from 1min, 5min, 10min, 15 min time window length

	Distance (km)	Avg_CtD (min)	Avg_RtD (min)	Energy (gallon)	GHG (kg)	CO (g)	PM2.5 (g)	NO_x (g)	Avg running time(sec)
TW = 1	2964.96	37.82	30.32	66.51	572.25	1850.01	2.58	106.11	1.89
TW = 5	2613.17	33.56	26.06	58.63	504.48	1644.72	2.28	93.61	47.85
TW= 10	2581.49	32.00	24.50	58.09	499.76	1620.04	2.26	92.69	82.38
TW=15	2509.43	33.26	25.76	56.37	485.00	1580.91	2.2	90.03	149.77

5.4.4 City level impact of on-demand meal delivery (Scenario 9 and 10)

From the above two parts, we have demonstrated the performance of on-demand food delivery in saving VMT, fuel consumption, and emissions. In this part, we further study the system-level impact of all eat-out incurred trips with different on-demand delivery penetration rate under two delivery policies. Four penetration rate scenarios are set up:

- None (0%, no on-demand food delivery)
- Low (14%, 185 delivery orders, 1143 dine-in orders)
- Mid (21%, 278 delivery orders, 1050 dine-in orders)
- High (40%, 531 delivery orders, 797 dine-in orders).

As described in Section 5.1, low penetration case corresponds to scenario in 2021, and mid penetration case corresponds to scenario in 2025 as predicted by [6].

All four cases results are summarized in Table 11. Under Multi-R policy, with higher on-demand food delivery penetration rate, the total eat-out incurred VMT reduces by 21%. Under all

penetration rates, the average CtD is around 33 minutes which is acceptable to most customers. Meanwhile, the percentage change of VMT and environmental factors compared to the No-Delivery case is shown in Figure 22. We can note that with 14%, 21% and 40% the eat-out incurred VMT, fuel consumption, and emission factors can reduce by around 5%, 10%, 25% respectively. This reveals the potential of on-demand food delivery to reduce traffic in the urban city and bring a greener transportation system. Similar results can also be obtained in the One-R policy: VMT and environmental factors are reduced by 14% under 40% of penetration rate. Table 11 also reveals that the Multi-R outperforms One-R in reducing the traffic and environmental impact. With the same penetration rate of 40%, the Multi-R policy produced 10% less eat-out incurred trips than the One-R policy. Meanwhile, the Multi-R policy requires a smaller number of drivers to finish the same amount of meal orders, which can maximize the vehicle utilization.

Table 11. Results from multiple penetration rates

Metric	Multi-R				One-R			
	None(0%)	Low(14%)	Mid(21%)	High(40%)	None(0%)	Low(14%)	Mid(21%)	High(40%)
Delivery Distance(km)	0	1966.00	2613.17	4167.29	0	2394.65	3681.46	6175.83
dine in distance(km)	22954.76	19870.27	18177.44	13946.67	22954.76	19870.27	18177.44	13946.67
total distance(km)	22954.76	21836.27	20790.62	18113.95	22954.76	22264.91	21858.91	20122.49
Avg_CtD (min)	n/a	33.58	33.56	31.93	n/a	21.78	24.39	23.33
Avg_RtD (min)	n/a	26.11	26.06	24.48	n/a	14.31	16.88	12.42
Energy (gallon)	512.22	487.31	464.05	404.52	512.22	496.94	487.88	449.67
GHG (kg)	4406.77	4192.51	3992.39	3480.20	4406.77	4275.33	4197.40	3868.67
CO (g)	13936.48	13279.16	12662.24	11055.26	13936.48	13554.29	13309.78	12270.36
PM2.5(g)	19.367	18.449	17.585	15.367	19.367	18.830	18.488	17.083
Nox (g)	815.242	775.811	738.859	644.250	815.242	791.244	776.864	715.845
total driver	n/a	42	61	106	n/a	56	81	147

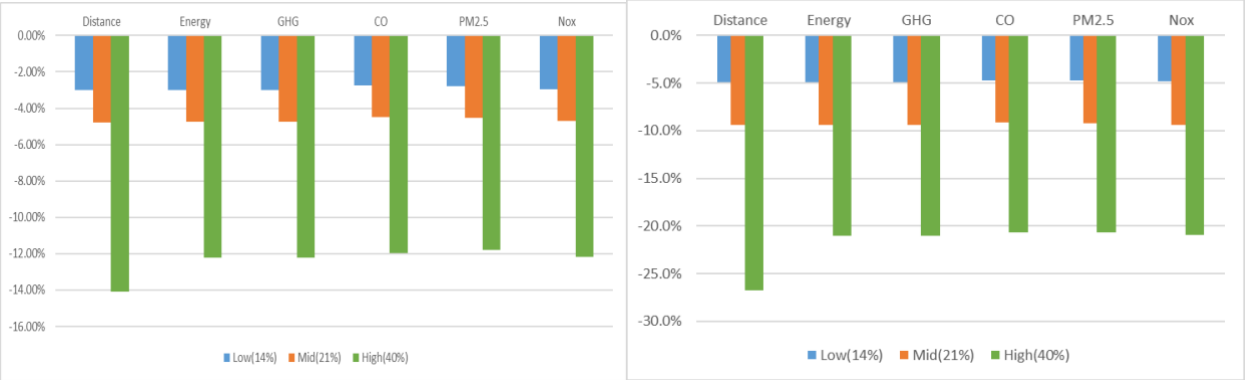

Figure 22. % change in VMT and environmental factors compared to None (0%) case. Left: One-R, right: Multi-R

6. Conclusion

This research proposes to improve freight vehicle utilization and energy efficiency (productive ton-miles per unit of energy) by modeling and evaluating the innovative shared mobility services for freight, with a specific focus on on-demand food delivery. We use a CEMDAP model to generate realistic eat-out demand, utilize Riverside BEAM model to quantify the driver travel distance and travel time, and employ the emissions model (from [26] and EMFAC) to evaluate the environmental impact of the eat-out incurred trips (including on-demand delivery and dine-in trips).

Based on the generated trips, we first assume all information is known beforehand to study a static on-demand food delivery problem. Multiple scenarios were simulated to evaluate the urban traffic and environmental impact of on-demand food delivery services before and during COVID. The evaluation results show that the total travel distance incurred by eat out demand during COVID time decreased by 25% compared with the before-COVID period. Fuel consumption and tailpipe emissions were also reduced by 25% approximately. If the penetration rate of on-demand food delivery increases from 16% to 50%, then the total travel distance can be reduced by 31%. Currently, the ODFD has already accounted for 14-16% of market share according to multiple surveys. The future higher penetration level of ODFD depends on three aspects. First, the ODFD platform needs to foster the online food ordering habits of customers by providing diverse and reliable service at a lower cost. Second, more restaurants are encouraged to offer the delivery option. Last, the ODFD platform needs improvement in driver capacity management, eco-friendly delivery strategy, and an efficient order allocation system responding to large volume of meal orders, real-time traffic condition, etc. We further extend the static assumption to dynamic setting, where food orders and drivers arrive and leave the ODFD system continuously. Two delivery policies are explored: Multi-R and One-R. The evaluation result shows that the Multi-R policy can save 45% of VMT, fuel and emissions compared with the baseline case. The average CtD of Multi-R is slightly higher than that of the One-R policy but it can bring 28% of VMT and and emissions reductions while not introducing additional delay. The effect of system update frequency and the on-demand penetration rate are evaluated as well. With lower update frequency and higher penetration

rate, the system can save more VMT and energy. Overall, the developed rolling horizon approach is able to produce high-quality order assignment and routing plans within in one minute which is capable of real time implementation. The proposed framework can also be readily implemented to other types of delivery problem (i.e., same-day grocery delivery, parcel delivery) by replacing the time constraints and vehicle capacity and shift plan.

Directions for future research can be summarized as follows:

1. Setting fuel consumption and emissions as the main objective of the PDPTW model directly to investigate the potential of eco-friendly on-demand food delivery.
2. Considering the initial distribution of driver location and impact of empty trips while waiting for the new orders
3. Considering the commercial factors in real-world operation such as profit, incentives and compensation for drivers.
4. The preference of drivers and restaurants can also be considered in the problem formulation.

References

[1] M. Jaller, C. Otero, E. Pourrahmani, and L. Fulton, “Automation, Electrification, and Shared Mobility in Freight,” Jun. 2020, doi: 10.7922/G2RV0KZB.

[2] O. Us Epa, “Fast Facts on Transportation Greenhouse Gas Emissions,” Aug. 2015, Accessed: Jun. 24, 2022. [Online]. Available: https://www.epa.gov/greenvehicles/fast-facts-transportation-greenhouse-gas-emissions

[3] W. E. gov/aeo, “Transportation energy consumption peaks in 2020 in the AEO2020 Reference case because rising fuel efficiency more than offsets the effects of increases in total travel and freight movements, but this trend reverses toward the end of the projection period.” https://www.eia.gov/outlooks/aeo/pdf/AEO2020%20Transportation.pdf (accessed Jun. 24, 2022).

[4] Beliën, Boute, Creemers, and De Bruecker, “Collaborative shipping: logistics in the sharing economy,” *ORMS*, 2017, [Online]. Available: https://lirias.kuleuven.be/1661024?limo=0

[5] Y. Zhao and F. Bacao, “What factors determining customer continuingly using food delivery apps during 2019 novel coronavirus pandemic period?,” *Int. J. Hosp. Manage.*, vol. 91, p. 102683, Oct. 2020.

[6] Statista, “Impact of COVID-19 on online restaurant delivery market share in the U.S. 2020-2025,” 2022. https://www.statista.com/statistics/1170614/online-food-delivery-share-us-coronavirus/ (accessed Jun. 17, 2022).

[7] C. Li, M. Mirosa, and P. Bremer, “Review of Online Food Delivery Platforms and their Impacts on Sustainability,” *Sustain. Sci. Pract. Policy*, vol. 12, no. 14, p. 5528, Jul. 2020.

[8] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, R. O’neil, and H. M. Stewart, “The Meal Delivery Routing Problem,” *Optimization Online*, 2018.

[9] B. Fan, L. Lv, and G. Han, “Online platform’s corporate social responsibility for mitigating traffic risk: Dynamic games and governmental regulations in O2O food delivery industry,” *Comput. Ind. Eng.*, vol. 169, p. 108188, Jul. 2022.

[10] J. Tao, H. Dai, W. Chen, and H. Jiang, “The value of personalized dispatch in O2O on-demand delivery services,” *European Journal of Operational Research*, May 2022, doi: 10.1016/j.ejor.2022.05.019.

[11] S. Ropke, J.-F. Cordeau, and G. Laporte, “Models and branch-and-cut algorithms for pickup and delivery problems with time windows,” *Networks*, vol. 49, no. 4, pp. 258–272, Jul. 2007.

[12] Y. Liu *et al.*, “FooDNet: Toward an Optimized Food Delivery Network Based on Spatial Crowdsourcing,” *IEEE Trans. Mob. Comput.*, vol. 18, no. 6, pp. 1288–1301, Jun. 2019.

[13] W. Tu, T. Zhao, B. Zhou, J. Jiang, J. Xia, and Q. Li, “OCD: Online Crowdsourced Delivery for On-Demand Food,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6842–6854, Aug. 2020.

[14] X. Wang, L. Wang, S. Wang, J.-F. Chen, and C. Wu, “An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem,” *Comput. Ind. Eng.*, vol. 152, p. 107029, Feb. 2021.

[15] Q. Zhou *et al.*, “Two Fast Heuristics for Online Order Dispatching,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2020, pp. 1–8.

[16] B. Yildiz and M. Savelsbergh, “Provably High-Quality Solutions for the Meal Delivery Routing Problem,” *Transportation Science*, vol. 53, no. 5, pp. 1372–1388, Sep. 2019.

[17] Z. Steever, M. Karwan, and C. Murray, “Dynamic courier routing for a food delivery service,” *Comput. Oper. Res.*, vol. 107, pp. 173–188, Jul. 2019.

[18] Ye, Konduri, Pendyala, and Sana, “A methodology to match distributions of both household and person attributes in the generation of synthetic populations,” *88th Annual Meeting*, 2009.

[19] “PopGen,” *MARG - Mobility Analytics Research Group*. https://www.mobilityanalytics.org/popgen.html (accessed Jun. 24, 2022).

[20] US Census Bureau and Center for Economic Studies, “US Census Bureau Center for Economic Studies Publications and reports page,” Jan. 01, 2010. https://lehd.ces.census.gov/data/ (accessed Jun. 24, 2022).

[21] Employment Development Department, https://edd.ca.gov/ (accessed Jun. 24, 2022).

[22] C. R. Bhat, J. Y. Guo, S. Srinivasan, and A. Sivakumar, “Comprehensive Econometric Microsimulator for Daily Activity-Travel Patterns,” *Transp. Res. Rec.*, vol. 1894, no. 1, pp. 57–66, Jan. 2004.

[23] *cemdap_input*. Github. Accessed: Jun. 24, 2022. [Online]. Available: https://github.com/yliao43/cemdap_input

[24] S. Ropke and D. Pisinger, “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows,” *Transportation Science*, vol. 40, no. 4, pp. 455–472, Nov. 2006.

[25] R. Bent and P. Van Hentenryck, “A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows,” *Comput. Oper. Res.*, vol. 33, no. 4, pp. 875–893, Apr. 2006.

[26] K. Boriboonsomsin, M. J. Barth, W. Zhu, and A. Vu, “Eco-Routing Navigation System Based on Multisource Historical and Real-Time Traffic Information,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1694–1704, Dec. 2012.

[27] A. Hagberg, P. Swart, and D. S Chult, “Exploring network structure, dynamics, and function using networkx,” Jan. 2008, [Online]. Available: https://www.osti.gov/biblio/960616

[28] T. Forscher, E. Deakin, J. Walker, and S. Shaheen, “How is the COVID-19 Pandemic Shifting Retail Purchases and Related Travel in the Sacramento Region?,” *Transportation Research Board*, Oct. 2021.

[29] M. W. P. Savelsbergh, “The Vehicle Routing Problem with Time Windows: Minimizing Route Duration,” *ORSA Journal on Computing*, vol. 4, no. 2, pp. 146–154, May 1992.

[30] J.-F. Cordeau and G. Laporte, “A tabu search heuristic for the static multi-vehicle dial-a-ride problem,” *Trans. Res. Part B: Methodol.*, vol. 37, no. 6, pp. 579–594, Jul. 2003.

[31] T. Gschwind and M. Drexl, “Adaptive Large Neighborhood Search with a Constant-Time Feasibility Test for the Dial-a-Ride Problem,” *Transportation Science*, vol. 53, no. 2, pp. 480–491, Mar. 2019.

[32] T. Ahamed, B. Zou, N. P. Farazi, and T. Tulabandhula, “Deep Reinforcement Learning for Crowdsourced Urban Delivery,” *Trans. Res. Part B: Methodol.*, vol. 152, pp. 227–257, Oct. 2021.

[33] “EMFAC.” https://arb.ca.gov/emfac/emissions-inventory/ (accessed Jun. 07, 2022).

[34] “BEAM.” https://transportation.lbl.gov/beam (accessed Jul. 01, 2022).

[35] EMFAC, 2021. https://arb.ca.gov/emfac/ (accessed Jun. 24, 2022).

Data Summary

Products of Research

In this project, we generated the eat-out trips using a CEMDAP model. The eat-out trip includes customer location, restaurant location, departure time. Then we set up the food delivery time of each order. Driver information is sampled from the customer location and assigned with specific working shift. Those data are used to validate the proposed algorithms and estimate the performance on VMT saving and emission reduction.

Data Format and Content

The data were saved in CSV files, each row indicates a food order. During the optimization process, the driver and order status are updated with the real-time traffic information extracted from BEAM. Every order will be assigned to one driver and the delivery time, VMT, fuel consumption, and emission cost to finish one order are recorded.

Data Access and Sharing

The data are publicly available via the UC Riverside instance of Dash: https://dash.ucr.edu/stash/, which is in compliance with the USDOT Public Access Plan. This dataset can be cited as:

> Hao, P., Liu, H., Liao, Y., Boriboonsomsin, K., Barth, M. (2022), Simulation Data for On-demand Food Delivery in Riverside, CA, UC Riverside, Dataset, https://doi.org/10.6086/D19X1J

Reuse and Redistribution

The data are restricted to research use only. If the data are used, our work should be properly cited: Hao, Hao, P., Liu, H., Liao, Y., Boriboonsomsin, K., Barth, M. (2022), Simulation Data for On-demand Food Delivery in Riverside, CA. UC Riverside Dash, Dataset.