Evaluating the Order Dispatching Strategies for a Dynamic On-Demand Meal Delivery System—A Case Study in the City of Riverside

Haishan Liu[1]; Yejia Liao[2]; Peng Hao, Ph.D.[3]; Kanok Boriboonsomsin, Ph.D.[4]; and Matthew Barth, Ph.D.[5]

[1]Center for Environmental Research and Technology, Univ. of California, Riverside, Riverside, CA. Email: hliu240@ucr.edu
[2]Center for Environmental Research and Technology, Univ. of California, Riverside, Riverside, CA. Email: yliao045@ucr.edu
[3]Center for Environmental Research and Technology, Univ. of California, Riverside, Riverside, CA. Email: haop@cert.ucr.edu
[4]Center for Environmental Research and Technology, Univ. of California, Riverside, Riverside, CA. Email: kanok@cert.ucr.edu
[5]Center for Environmental Research and Technology, Univ. of California, Riverside, Riverside, CA. Email: barth@ece.ucr.edu

ABSTRACT

The prevalence of information and communication technologies has catalyzed the emergent on-demand meal delivery (ODMD) service. Order dispatching policies play a crucial role in determining the operational performance and environmental impact of ODMD system. In this paper, we proposed a comprehensive framework to evaluate meal delivery efficiency and quantify the corresponding environmental impact of ODMD system under different order dispatching policies. It consists of three components: (1) the real-world meal delivery operational context that is generated to simulate daily activities and travel patterns in the city of Riverside and construct the on-demand meal delivery scenarios; (2) an efficient optimization approach: rolling horizon-based adaptive large neighborhood search algorithm, to obtain order dispatching and routing decisions with the dynamic order demand and driver resources; and (3) an energy consumption and emission evaluation model developed via EMFAC to quantify the corresponding fuel consumption and pollutant emissions. With the proposed framework, three order dispatching policies are evaluated: One-Order-Per-Trip (One-O), One-Restaurant-Per-Trip (One-R), and Multi-Restaurant-Per-Trip (Multi-R). Simulation results show substantial benefits of having orders bundled in the One-R and Multi-R policies. The total travel distance is reduced by 14% and 30%, respectively, compared to the One-O policy. Meanwhile, the total number of delivery drivers is reduced by 36% and 60%, which shows great potential to relieve the urban traffic burdens, together with 14%–30% fuel consumption, GHG emissions, and criteria pollutant emissions reductions. Meanwhile, the service quality is maintained with only 1% of late orders.

Keywords: Dynamic on-demand meal delivery, Order dispatching, Impact evaluation, Rolling horizon framework, Adaptive large neighborhood search (ALNS).

INTRODUCTION

Catalyzed by the prevalence of information and communication technologies and boosted by the unexpected COVID-19 pandemic, on-demand meal delivery (ODMD) has achieved explosive growth (Seghezzi et al. 2021). As reported by Statista, the U.S. food delivery has comprised 16% of the total restaurant market (Statista 2022a). The revenue from ODMD is projected to reach 63.02 billion dollars by 2022 and shows an annual growth rate of 8.9% (Statista 2022b). The fast growth of meal delivery demand brings challenges to the online platform to efficiently dispatch meal orders. Meanwhile, the surging delivery trips generated inside the city may further exacerbate traffic congestion and bring negative environmental impacts. Thus, it is necessary to evaluate the platform's possible order dispatching policy from both operational and environmental perspectives, which can, in turn, provide insights to help the ODMD platform balance the primary goal of delivering meals in a shorter time and bringing minimal negative effects.

Most recent research of ODMD can be grouped into two categories: static and dynamic. In the static ODMD setting, the authors assume perfect information of meal orders of the whole operational horizon. In some research, the authors divide the horizon into several time intervals to reduce the complexity, but each time interval's solution is isolated which also falls in the static ODMD scenario. Liu et al (2019) proposed to leverage taxi resources to deliver food orders either in an opportunistic manner or in a dedicated manner with the goal to minimize taxi number and distance cost. Tu et al (2020) developed an online dynamic optimization framework which includes order collection, solution generation and sequential delivery. This approach decomposes the large-scale problem into multiple static small-scale problems without considering the interactions between each time interval. Wang et al (2021) 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 the XGBoost algorithm to select the order sequencing rules. In the dynamic meal delivery setting, orders and drivers are revealed dynamically during operating hours and the online platform has to respond to the new delivery demand efficiently. Zhou et al (2020) formulated an online order dispatching system with new orders arrival and extended the traditional greedy insertion and regret insertion heuristic to evaluate more orders in one iteration, but this research only solved the problem in one time interval without considering the platform update. Reyes et al (2018) studied the meal-delivery routing problem (MDRP) and proposed a rolling-horizon repeated-matching algorithm to solve the dynamic vehicle routing problem and capacity management problem, where they only bundled orders from the same restaurant. Huang et al (2021) investigated the dynamic task scheduling problem of a UAV-based ODMD system and proposed an iterated heuristic to obtain the solution with minimized order tardiness, where each UAV is only allowed to carry one order in the delivery.

However, most existing research focused on developing efficient and fast algorithms, thus assuming simplified operational context, i.e., constant delivery speed and Euclidean distance between two locations, which makes it impossible to evaluate the impact of ODMD, especially from the environmental perspective. On the other hand, the difference of dispatching policies has not been investigated before. Most research obtained the optimized solution without considering the platform's operating strategies. To investigate this issue, we propose a comprehensive framework to evaluate the impact of ODMD service with real-world dynamic operational scenarios. First, a mathematical model is formulated to describe the dynamic ODMD order dispatching and routing problem. Next, with the real-world operational context for delivery drivers, a rolling horizon based adaptive large neighborhood search algorithm is proposed to obtain the optimized order dispatching solution that satisfies the platform's order dispatching policies. Finally, indicators and emission model are used to evaluate the dispatching policy's corresponding operational performance and environmental impact.

DYNAMIC ON-DEMAND MEAL DELIVERY PROBLEM

Problem Description

We consider a dynamic on-demand meal delivery (ODMD) system, which consists of four stakeholders: online platform, delivery driver, restaurant, and dinning customer. Dinning customer chooses preferable food and places a meal order using the online platform. A meal order typically includes information about meal items, restaurant location and customer location. The online platform gathers multiple orders and assigns delivery drivers to complete delivery tasks. An estimated drop-off time is provided by the platform which represents a delivery commitment. The platform aims to construct feasible routes for drivers with the objective to minimize delivery delay and total delivery cost.

Different from traditional static meal delivery settings (Paul et al. 2020), we study the dynamic ODMD scenario, where meals are ordered dynamically and delivery drivers can log on and log off the system freely during the operating time depending on drivers' working schedules. However, it is not practical to update the system dispatching decision when receiving every new order. Similar as (Chen et al. 2022; Reyes et al. 2018), we employ a rolling horizon approach which divides the whole operational horizon H into $\left\lceil \frac{H}{\tau} \right\rceil$ time intervals with a length of τ. Suppose the platform begins at time t_0. At every time $t_h (where\, t_h = t_0 + h \times \tau\,, h = (1,2,\dots, \left\lceil \frac{H}{\tau} \right\rceil)$, the system will re-optimize its order dispatching decision regarding new meal order demand and driver information revealed during $[t_{h-1}, t_h)$.

Within the dynamic setting, order status and driver status are updated during the operating time. As shown in Figure 1, we define three types of states of each order. At time t_h, if an order o is placed in the range of $[t_{h-1}, t_h)$, then it falls in the new order set O^n. After time t_h, the platform will dispatch each new order to drivers. Then the order status changes to "scheduled" in O^s. Drivers will deliver meal orders sequentially and an order is turned to completed status in O^c if the order is finally delivered. Accordingly, each delivery driver has two states: working (in set K^W) or idle (in set K^I). Driver k first logs on the platform and is at idle status waiting for platform order dispatching. Suppose at time t_h, the driver receives delivery tasks and starts working. In addition, some drivers may have multiple trips and their status keep switching between "working" and "idle" until they leave the platform. We further assume driver idles around the last-visit location when completing all orders at hand.

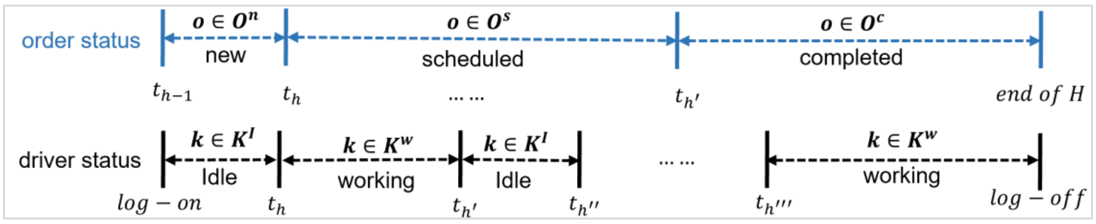


Figure 1. Meal order and driver status definition

Model Formulation

With the above description, at each re-optimization time t_h, we can specifically formulate a pick-up and delivery time window model (PDPTW) (Bent and Van Hentenryck 2006) to represent the order dispatching and routing optimization problem. The model variable and parameter definitions are listed in Table 1. With the clear defined sets of $R^h, P^h, D^h,$ and K^h in Table 1, then we can construct a directed graph $G = (V, E)$, in which each node in V represents the location of a customer, a restaurant, or a driver, $V = P^h \cup D^h \cup K^h$, and each arc in E(where $E = V \times V$) represents the movement from one node to another. Specifically, we use a vector $\langle i, j, q_i, q_j, t_p^i\ t_r^i, \ t_{do}^{ie} \rangle$ to describe the key information of an order, where i, j are the paired pick-up and drop-off locations, $q_i + q_j$=0, and the last are three time variables: order place time, order ready time and estimated drop-off time. Then we can formulate the on-demand meal delivery problem as follows.

Table 1. Variable and parameter definition

Horizon	
τ	Time interval length
t_h	System re-optimized time, $t_h = t_0 + h \times \tau, h \in (1, 2, \dots, \lfloor \frac{H}{\tau} \rfloor)$, t_0 is the start time
Indices	
i/j	pick-up/ drop-off task at a restaurant/customer location
k/k_0	delivery driver, k_0 represents the initial location of driver *k*
Sets	
R^h	Set of all meal order requests. Each request consists of a pair of pick-up and drop-off tasks (i, j), where $R^h = O^n \cup O^s$ during time interval $[t_{h-1}, t_h)$
P^h	Set of all pick-up tasks from R^h during time interval $[t_{h-1}, t_h)$
D^h	Set of all drop-off tasks from R^h during time interval $[t_{h-1}, t_h)$
K^h	Set of all delivery drivers, where $K^h = K^I \cup K^w$ during time interval $[t_{h-1}, t_h)$
Parameters and constants	
q_i	Order number to be served at location *i*. Positive when i is a pick-up location; negative when i is a drop-off location
Q^k	Driver *k* delivery capacity
s_i	Service time of task i (load/unload)
t_p^i	Order place time, $i \in R^h$, decided by customer
t_r^i	Order ready time, $i \in R^h$, decided by the restaurant.
t_{do}^{ie}	Estimated drop-off time, $i \in R^h$, provided by the platform
t_{ij}	Travel time of a OD pair (i, j)
d_{ij}	Travel distance of a OD pair (i, j)
Intermediate variables	
t_{pu}^i	The actual visit time at the pick-up location *i*, $i \in P^h$
t_{do}^i	The actual visit time at the drop-off location *i*, $i \in D^h$
Q_i^k	Number of orders assigned when driver *k* leaves node *i*
Decision Variable	
x_{ij}^k	1 if OD pair (i, j) traveled by driver k; 0 otherwise
T_i^k	Time when vehicle k visit node *i*

The platform aims to minimize the total travel distance and total order delay cost to maintain high service quality. In the objective function, the first term represents the total delivery distance, and the second term is the total delivery delay, which is defined as the difference between the actual drop-off time t_{do}^{i} and the estimated drop-off time t_{do}^{ie}. We enforce a linear delay penalty function p_c to penalize late delivery. α and β are weight factors.

$$Min\ F = \alpha \sum_{k \in K^h} \sum_{(i \in V, j \in V, i \neq j)} d_{ij} x_{ij}^{k} \ + \beta \sum_{i \in D^h} p_c \max\left(0, t_{do}^{i} - t_{do}^{ie}\right)$$

subject to

Route construction constraints:

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

$$\textstyle\sum_{j \in V} x_{k_0,j}^{k} = 1 \qquad \forall\, k \in K^h \tag{2}$$

$$\textstyle\sum_{i \in V} x_{i,k_0}^{k} = 1 \qquad \forall k \in K^h \tag{3}$$

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

$$\textstyle\sum_{j' \in P^h \cup D^h \cup k_0} x_{ij'}^{k} - \sum_{j \in P^h \cup D^h \cup k_0} x_{j',j}^{k} = 0 \qquad \forall (i,j) \in R, \forall k \in K^h \tag{5}$$

$$T_i^k \leq T_j^k \qquad \forall (i,j) \in R, \forall k \in K^h \tag{6}$$

Capacity constraints:

$$Q_i^k \leq Q^k \qquad \forall i \in V, \forall k \in K^h \tag{7}$$

$$x_{ij}^{k} = 1 \Rightarrow Q_j^k \geq Q_i^k + q_j \qquad \forall i,j \in V, \forall k \in K^h \tag{8}$$

Time constraints:

$$t_{pu}^{i} \geq \max(t_r^i, T_i^k) \qquad \forall i \in P^h \tag{9}$$

$$x_{ij}^{k} = 1 \Rightarrow T_j^k \geq t_{pu}^{i} + t_{ij} + s_i \qquad \forall i \in P^h, \forall k \in K^h \tag{10}$$

$$x_{ij}^{k} = 1 \Rightarrow T_j^k \geq t_{do}^{i} + t_{ij} + s_i \qquad \forall i \in D^h, \forall k \in K^h \tag{11}$$

Variable constraints:

$$Q_i^k \geq 0 \qquad \forall i \in V, \forall k \in K^h \tag{12}$$

$$x_{ij}^{k} \in \{0,1\} \qquad \forall (i,j) \in E, \forall k \in K^h \tag{13}$$

$$T_i^k \geq 0 \qquad \forall i \in V, \forall k \in K^h \tag{14}$$

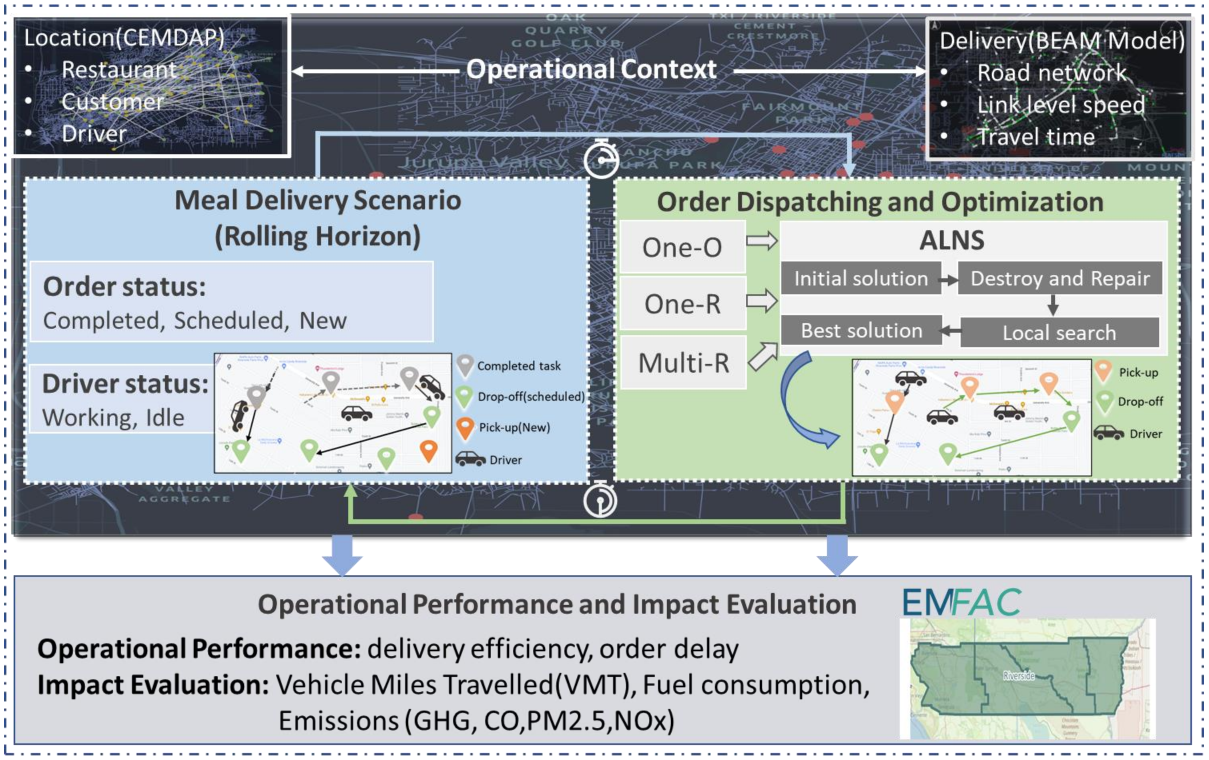
To ensure the feasibility of routes, we set up specific constraints and grouped them into four categories. The first category mainly states the basic requirements of route construction. Constraint (1) ensures all meal orders are served. Constraint (2)-(4) enforce each driver to log on from the initial location, return to it after log-off, and every route should obey the flow conservation constraint. Constraint (5) states that a pair of pick-up and drop-off tasks from one order should be completed by the same driver. We do not allow transferring tasks between drivers. Constraint (6) guarantees that for one order, the driver should visit the pick-up location first and then deliver the meal at the customer location. The second category, including constraints (7) and (8), specifies the capacity constraints of the delivery driver, which describes the maximum number of orders a driver can receive at any given time and the loaded orders number change in the delivery route. Time constraints are summarized in the third group. Constraint (9) states that one driver can arrive earlier at the restaurant but should wait until t_r^i to pick up the meal order. Constraints (10) and (11) define driver arrival time at node j is no less than the pick-up or drop-off time at node i, plus travel time t_{ij} and service time at node i. The last group ensures driver served order number is no less than zero (constraint (12)) and provides the decision variables' definition (constraint (13)-(14)).

FRAMEWORK OVERVIEW AND METHODOLOGY

This section proposed a comprehensive framework to evaluate the operational performance and environmental impact of the dynamic on-demand meal delivery service. Figure 2 illustrates the main components of this framework. First, to enhance the accuracy, we derive a real-world operational context in the City of Riverside, California. Then a rolling horizon approach is utilized to construct the meal delivery scenario at each re-optimization time. Since meal delivery problem is a variant of the vehicle routing problem (VRP) which is a famous NP-hard problem. The adaptive large neighborhood search (ALNS) algorithm is applied to gain the optimized order dispatching and routing results in a computationally efficient manner (Ropke and Pisinger 2006). We specifically design the ALNS algorithm to provide the optimized solution that exactly satisfies the platform operation strategy (i.e., One-O, One-R and Multi-R). With the order dispatching result, we can evaluate the delivery efficiency and quantify the fuel consumption and pollutant emissions of delivery drivers. The remaining parts of this section will describe the details of each component.

Operational Context

The meal delivery operational context is construct from both CEMDAP and BEAM model (BEAM 2020; Bhat et al. 2004). We first utilize CEMDAP, a daily activity generation software, to generate the eat out activity of residents and sample a portion of customers to use the ODMD service instead of dine in. The delivery drivers are assumed to be local people who starts from their home location. The road network and traffic information are extracted from BEAM model, an agent-based simulation platform, where link-level travel distance and travel speed are employed to estimate the drivers' travel time from one location to another. The locations and movements of all drivers in the system are tracked and archived during the entire simulation process.



Location(CEMDAP)
Restaurant
Customer
Driver
Operational Context
Delivery(BEAM Model)
Road network
Link level speed
Travel time
Meal Delivery Scenario (Rolling Horizon)
Order status: Completed, Scheduled, New
Driver status: Working, Idle
Completed task
Drop-off(scheduled)
Pick-up(New)
Driver
Order Dispatching and Optimization
One-O
One-R
Multi-R
ALNS
Initial solution
Destroy and Repair
Best solution
Local search
Pick-up
Drop-off
Driver
Operational Performance and Impact Evaluation
EMFAC
Operational Performance: delivery efficiency, order delay
Impact Evaluation: Vehicle Miles Travelled(VMT), Fuel consumption, Emissions (GHG, CO,PM2.5,NOx)

Figure 2. Overview of the proposed evaluation framework

Meal Delivery Scenario Simulation with Rolling Horizon

Each meal order generated from CEMDAP has an order place time t_p^i, the customer location, and restaurant location. Drivers have their own initial home location and working schedule. At each re-optimization time t_h, to construct the meal delivery scenario, we have to specifically define the sets of orders R^h, restaurants P^h, customers $D^h$and drivers K^h. The detailed scenario simulation is presented in Algorithm 1. The meal scenario update depends on the previous order/driver status update (Step 1) and the accumulated the new orders and drivers from the time interval $[t_{h-1}, t_h)$(Step 2).

```
Algorithm 1: Meal delivery scenario simulation
   Input: Last dispatching result $S^{h-1}$, new order and drivers in $[t^{h-1}, t^h)$
   Output: updated set $R^h$, $P^h$, $D^h$, $K^h$
1  Step 1: update order and driver status in $S^{h-1}$
2      if order is delivered before $t^h$ then
3          Add order to set $O^c$
4      else
5          Add order to set $O^s$
6      end
7      if driver has on-going task at $t^h$ then
8          Add driver to set $K^w$
9      else
10         Add driver to set $K^I$
11     end
12 Step 2: Receive new order and driver revealed in $[t^{h-1}, t^h)$
13     add new orders to $O^n$ and add new drivers to $K^I$
14     update order set $R^h = O^s \cup O^n$ and get $P^h$, $D^h$ accordingly
15     update driver set $K^h = K^I \cup K^w$
```

ALNS-based Order Dispatching and Optimization

With the updated meal delivery scenario, the platform needs to dispatch new orders to delivery drivers and update driver's task sequence. With a feasible initial routing solution, we employed the ALNS algorithm to achieve an optimized solution efficiently. ALNS is a meta-heuristic 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. We mainly followed the main components of ALNS proposed in (Ropke and Pisinger 2006). Five removal operators are implemented to perturb the solution space, including random removal, worst removal, Shaw removal, longest distance path removal, worst delay path removal. Four repair operators are used to re-insert the orders to find a better solution, including random repair, greedy repair, regret-2 and regret-3 repair. The generic idea of each operator can be found in (Liu et al. 2022).

Generally, the platform has specific dispatching policies to achieve its goals. The ALNS algorithm should be properly designed to ensure that the optimized dispatching results exactly follow the dispatching policy. In this paper, we aim to evaluate three commonly used dispatching polices:

- One-Order-Per-Trip (One-O): allowing drivers to finish only one order per delivery trip to avoid meal delivery delay.
- One-Restaurant-Per-Trip (One-R): allowing drivers to visit only one restaurant per delivery trip to pick up one or more orders altogether, and then complete the delivery task sequentially.
- Multi-Restaurant-Per-Trip (Multi-R): allowing drivers to pick up meal orders from multiple restaurants either in the same commercial zone or near the planned delivery route.

In all three policies, transferring the previous assigned orders to another driver is not allowed. Thus, only the new orders can be removed in the ALNS removal process. In the Multi-R policy, all repair operators can be utilized to obtain a better solution. In the One-O policy, we specifically set each driver's capacity to be one. Thus, repair operators under One-O policy only insert new orders to idle drivers. In the One-R policy, the repair operator inserts new order either to an idle driver or drivers served the same restaurant. An overview of the order dispatching and optimization process with ALNS is shown in Algorithm 2.

Algorithm 2: Order dispatching and optimization with ALNS 1:

```
Input: Last dispatching result $S^{h-1}$,updated set $R^h, P^h, D^h, K^h$, dispatching policy
Output: Optimized solution $S^h$ at time $t^h$
1  Step 1: Construct a feasible initial solution $S^h$ w.r.t to dispatching policy
2  Step 2: ALNS improvement
3      while not reach the max iteration or max iteration without improvement do
4          select a removal operator to destroy the solution /* only removed the new orders */
5          select a repair operator to re-insert the orders and obtain $S^{h-new}$ w.r.t dispatching
            policy
6          if $obj(S^{h-new}) \leq obj(S^h)$ then
7              $S^h = S^{h-new}$
8          else
9              Accept $S^{h-new}$ with the probability from simulated annealing
10         end
11         update operator score
12     end
```

Operational Performance and Impact Evaluation

In this section, we introduce indicators and methods to analyze the delivery service quality and the impact of adopting different order dispatching polices. At the end of the operational horizon, we summarize each order's actual pick-up time t_{pu}^{i}, drop-off time $t_{do}^{i}$and the distance cost with the optimized dispatching result. We define following indicators in this model.

- Click to door time (CtD): the time difference between drop-off time and order place time.
- Ready to door time (RtD): the time difference between drop-off time and order ready time.
- Number of dispatched driver: drivers required to finish the delivery tasks.
- Batching rate: portion of orders that are batched with at least one order to deliver together.
- Ratio of late order: orders that are suffered more than 10 minutes delay.

Meanwhile, EMFAC model (EMFAC 2021) is chosen to quantify the delivery driver's fuel consumption and pollutant emission. EMFAC is an Emission Factor Model, in which an emission factor is derived from the average value of repeated measurements of total emissions per driving cycle. We construct a Riverside EMFAC model and specify the delivery vehicle to be the gasoline vehicle. Emission rates with multiple speed ranges are obtained. Then with the driver routing result and traffic network, we can sum up all the delivery vehicle emissions to evaluate the environmental impact. In this research, we will evaluate fuel consumption and the emission of greenhouse gas (GHG), Nitric oxide (NOx), Carbon Monoxide (CO), and PM2.5. The vehicle-miles-travelled (VMT) of delivery drivers can also be obtained from the routing result.

EXPERIMENTS AND RESULT

Parameters Setting and Experiments

Given the inputs of various land-use, sociodemographic, activity, and transportation level-of-service attributes, CEMDAP provides the output of complete daily activity-trip patterns for each individual (Bhat et al. 2004). Focusing on the noon peak from 11:30am-12:30pm, we obtained 1328 eat-out trips in total and sampled 21% trips (total 278) as meal orders for delivery according to the statistics from (Statista 2022a). To simulate the ODMD operation, we set up the key parameters as follows. We focused on a 60-minute operating period and set the system update and re-optimization time interval τ to be 5 minutes. The meal order is added to the system according to its place time t_{p}^{i}. Then we assume the restaurant needs 5-20 minutes for meal preparation, which determines the food ready time t_{r}^{i}. The estimated drop-off time is set to 40 minutes after the order place time (that is $t_{do}^{ie} = t_{p}^{i} + 40$), indicating that the system aims to deliver every food order within 40 minutes. Each driver needs extra one minute to pick-up or drop-off when arrive at the restaurant/ customer location. Drivers are first generated around the residential area and the driver number is according to the order/driver ratios (set as 5) at each re-optimizing time interval. If current drivers are not sufficient to construct a feasible solution to deliver all the new orders, we add new drivers as needed. Finally, the weighted factor α and β are both set to be 1 in the objective function.

For parameter setting in ALNS algorithm, we mainly took the reference in (Ropke and Pisinger 2006) and modified some key parameters. Specifically, we set the destruction degree of removal operator is 0.2, indicating 20% new orders are removed in each iteration in order to sufficiently explore the solution space. The reaction factor in the operator weight adjustment is set to be 0.6. The cooling rate in simulated annealing process is set to be 0.99. In the ALNS iteration, we total ran 100 segments, and each segment contains 50 iterations. If 250 continuous iterations without any improvement occurred, we terminated the algorithm. Both Rolling Horizon and ALNS algorithms are coded in python 3.8 and run with a ThinkPadX1 Carbon 2021 with 16GB of RAM and an Intel Core I7- 1165G7 processor. Figure 3 shows ALNS convergence with the three proposed order dispatching policies in one time interval. Even in the Multi-R case, which has the highest computational complexity since the algorithm needs to find the best insertion position across all possible routes, the average running time for one time interval is 47.85 seconds. The proposed algorithm shows the potential of real-time decision support for the ODMD platform.

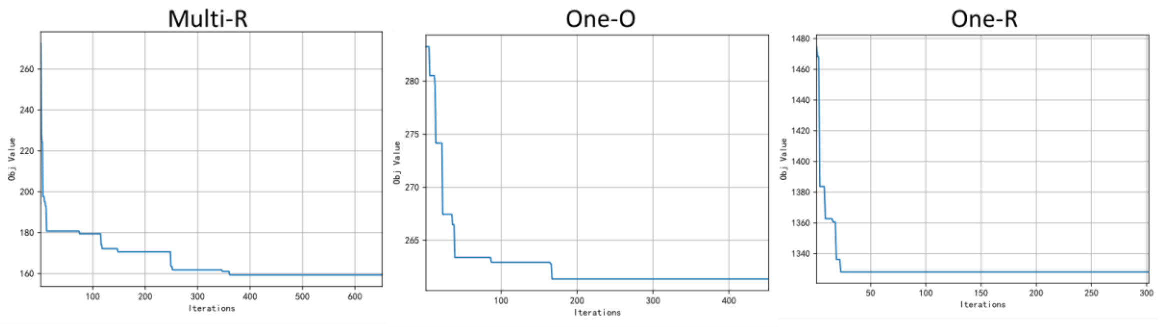


Figure 3. The performance of ALNS with Multi-R, One-O and One-R policy

Evaluation Result

We obtained the order dispatching and routing result with the proposed algorithm under three policies and evaluated the ODMD service operational performance and related impact. Table 2 summarizes the performance of different policies. With the Multi-R policy, drivers are allowed to pick up multiple orders from multiple restaurants nearby before making delivery or even to pick up a new order during the delivery route. The order batching rate reaches 87.41% and only requires 61 drivers to finish the total 278 orders. With more orders batched together and optimized delivery trips, we can reduce the redundant trips between the commercial and residential areas in the city, thus achieving around a 30% reduction in travel distance. Results from the EMFAC model also show nearly 30% savings of fuel consumption, GHG, CO, PM2.5 and NOx emissions compared to the One-O policy. In the One-R policy, drivers only pick up one or more orders that are arrived at the same restaurant. The order batching rate is 61.17% and it can reduce around 14% of travel distance, fuel consumption and emissions. In the One-O policy, as every order is delivered separately, more drivers are expected to be dispatched with higher total delivery cost. On the other hand, One-O shows a significant advantage in delivery time as all orders are directly delivered from the restaurant to the customer. The average CtD in the One-R and Multi-R policies are 3 minutes and 16 minutes higher than that in the One-O policy since the driver needs to visit more locations before drop-off the meal order. One-O policy can also keep the meals' freshness to the largest extent as the average RtD is only 9.41 minutes. But only 1% of orders, that is 2 out of 278 orders, may be delivered later than 10 minutes of the estimated drop-off time with orders bundled in the One-R and Multi-R policy.

Table 2. Performance comparison of three order dispatching polices

	Dispatching policy	One-O	One-R	Multi-R	One-R vs One-O	Multi-R vs One-O
Operational Performance	Orders delivered	278	278	278	0	0
	Number of dispatched driver	155	99	61	-36.13%	-60.65%
	Batching rate	0%	61.17%	87.41%	61.17%	87.41%
	Avg_CtD(min)	16.91	20.37	33.56	↑ 3.46	↑ 16.65
	Avg_RtD(min)	9.41	12.83	26.06	↑ 3.42	↑ 16.65
	Ratio of late order	0%	1.09%	1.07%	1.09%	1.07%
Impact Evaluation	Total travel distance(km)	4341.63	3724.35	2613.17	-14.22%	-29.84%
	Fuel(gallon)	97.12	83.38	58.64	-14.15%	-29.67%
	GHG(kg)	835.58	717.41	504.48	-14.14%	-29.68%
	CO(g)	2674.96	2321.13	1644.72	-13.23%	-29.14%
	PM2.5(G)	3.72	3.22	2.28	-13.44%	-29.19%
	NOx(g)	154.76	133.13	93.61	-13.98%	-29.69%

Evaluation with Customized Click to Door Time

In the above evaluation, we set all orders are estimated to be delivered in 40 minutes. While in the real world, if an order is placed from a distant restaurant or the order is placed exactly during noon, then the customers may be willing to accept a slightly longer waiting time. Thus, in this section, we customized each order's estimated drop-off time. If an order is placed 15 km away from the customer's location or is placed between 12:00-12:20 pm, we extend five more minutes of the estimated drop-off time of each condition. The meal delivery performance and impact evaluation are summarized in Table 3 with this customized click-to-door time setting. With the customized CtD, the order batching rate increases to 67.73% and 92.54% in One-R and Multi-R policy respectively. Accordingly, the travel distance and environmental impact are reduced by 18%-35% compared to the One-O policy. As the customized CtD strategy eliminates the tough orders that require long-distance delivery in a fixed time window, the system has enough time to deliver all orders in time without any late delivery. Regarding the delivery time, the One-O policy keeps outperforming the One-R and Multi-R policies with 4 minutes and 19 minutes savings from the average CtD or RtD respectively.

Table 3. Performance comparison of three order dispatching polices with customized CtD

	Dispatching policy	One-O	One-R	Multi-R	One-R vs One-O	Multi-R vs One-O
Operational Performance	Orders delivered	278	278	278	0	0
	Number of dispatched driver	151	94	54	-37.75%	-64.24%
	Batching rate	0%	67.63%	92.45%	67.63%	92.45%
	Avg_CtD(min)	16.76	21.17	36.11	↑ 4.41	↑ 19.35
	Avg_RtD(min)	9.26	13.68	28.6	↑ 4.42	↑ 19.34
	Ratio of late order	0%	2.15%	0.00%	0.00%	0.00%
Impact Evaluation	Total travel distance(km)	4324.09	3500.8	2244.06	-19.04%	-35.90%
	Fuel(gallon)	96.72	78.4	50.48	-18.94%	-35.61%
	GHG(kg)	832.18	674.51	434.32	-18.95%	-35.61%
	CO(g)	2650.73	2197.03	1421.69	-17.12%	-35.29%
	PM2.5(G)	3.7	3.04	1.98	-17.84%	-34.87%
	NOx(g)	154.03	125.31	80.69	-18.65%	-35.61%

CONCLUSIONS

In on-demand meal delivery (ODMD) service, the platform's order dispatching policy determines the operational performance and environmental impact of ODMD. This paper presented a comprehensive framework to evaluate the delivery efficiency and environmental impact of ODMD under three dispatching policies. Simulation results illustrate the substantial benefits of order batching in the One-R and Multi-R policies. The VMT is reduced by 14% and 30% respectively compared to the One-O policy along with the 36% and 60% reduction in the number of delivery drivers. The reductions in VMT and delivery drivers show significant potential to relieve the urban traffic burdens with the One-R and Multi-R policies. Meanwhile, fuel consumption, GHG emission, and criteria pollutants emissions are reduced significantly (14%-30%). On the other hand, there is a trade-off between the batching benefits and delivery time. Although the One-O policy has the highest negative impact on traffic and environment, it can deliver orders in the shortest time to guarantee a timely delivery while keeping freshness of the meal to the largest extent. This research will encourage the platform to consider a dispatching policy that is more friendly to the environment and traffic when designing the delivery system.

In the future, to make the evaluation framework explain more ODMD delivery factors, we need to consider the initial distribution of driver location and the impact of empty trips, i.e., deadheading, while waiting for new orders. Meanwhile, for the platform operating cost, we should consider real-world operation factors such as profit, incentives, and compensation for drivers. Finally, exact solvers, i.e., CPLEX or Gurobi, can be utilized to solve small-scale ODMD problems, then compare with the ALNS solutions to understand the gaps with global optimal solutions.

ACKNOWLEDGEMENT

This study is funded by the National Center for Sustainable Transportation (NCST) and California Air Resources Board. The contents of this paper reflect only the views of the authors, who are responsible for the facts and the accuracy of the data presented.

REFERENCES

BEAM. (2020). "The Modeling Framework for Behavior, Energy, Autonomy, and Mobility.", <https://transportation.lbl.gov/beam>(Jul. 1, 2022).

Bent, R., and P. Van Hentenryck. (2006). "A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows." *Comput. Oper.Res.*, 33 (4): 875–893.

Bhat, C. R., J. Y. Guo, S. Srinivasan, and A. Sivakumar. (2004). "Comprehensive Econometric Microsimulator for Daily Activity-Travel Patterns." *Transp. Res. Rec.*, 1894 (1): 57–66.

Chen, J.-F., L. Wang, H. Ren, J. Pan, S. Wang, J. Zheng, and X. Wang. (2022). "An Imitation Learning-Enhanced Iterated Matching Algorithm for On-Demand Food Delivery." *IEEE Trans. Intell. Transp. Syst.*, 1–17.

EMFAC. (2021). "Emission inventory", <https://arb.ca.gov/emfac/emissions-inventory/>(Jun. 7, 2022).

Huang, H., C. Hu, J. Zhu, M. Wu, and R. Malekian. (2021). "Stochastic Task Scheduling in UAV-Based Intelligent On-Demand Meal Delivery System." *IEEE Trans. Intell. Transp. Syst.*, 23 (8): 13040–13054.

Liu, H., Y. Liao, P. Hao, K. Booriborsomsin, and M. Barth. (in press). “Model-based VMT and Emission evaluation of on-demand food delivery considering the impact of COVID-19 pandemic.” *102nd Transporation Research Board*, Washington DC, U.S.

Liu, Y., B. Guo, C. Chen, H. Du, Z. Yu, D. Zhang, and H. Ma. (2019). “FooDNet: Toward an Optimized Food Delivery Network Based on Spatial Crowdsourcing.” *IEEE Trans. Mob. Comput*, 18 (6): 1288–1301.

Paul, S., S. Rathee, J. Matthew, and K. M. Adusumilli. (2020). “An optimization framework for on-demand meal delivery system.” *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, Singapore, Singapore.

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

Ropke, S., and D. Pisinger. (2006). “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows.” *Transp. Sci.*, 40 (4): 455–472.

Seghezzi, A., M. Winkenbach, and R. Mangiaracina. (2021). “On-demand food delivery: a systematic literature review.” *The Int. J. Logist. Manag.*, 32 (4): 1334–1355.

Statista. (2022a). “ Change in online restaurant delivery penetration share of the restaurant market in the United States due to the coronavirus pandemic from 2020 to 2025.” Statista, <https://www.statista.com/statistics/1170614/online-food-delivery-share-us-coronavirus/ >(Jul. 30, 2022).

Statista. (2022b). “Online Food Delivery - United States.” Statista, <https://www.statista.com/outlook/dmo/eservices/online-food-delivery/united-states>(Jul. 30, 2022).

Tu, W., T. Zhao, B. Zhou, J. Jiang, J. Xia, and Q. Li. (2020). “OCD: Online Crowdsourced Delivery for On-Demand Food.” *IEEE Internet Things J.*, 7 (8): 6842–6854.

Wang, X., L. Wang, S. Wang, J.-F. Chen, and C. Wu. (2021). “An XGBoost-enhanced fast constructive algorithm for food delivery route planning problem.” *Comput. Ind. Eng.*, 152: 107029.

Zhou, Q., H. Zheng, S. Wang, J. Hao, R. He, Z. Sun, X. Wang, and L. Wang. (2020). “Two Fast Heuristics for Online Order Dispatching.” *IEEE Congress on Evolutionary Computation (CEC)*, 1–8.