A Four-stage Heuristic Algorithm for Solving On-demand Meal Delivery Routing Problem

Lejun Zhou
Zhejiang University/University of Illinois at Urbana–Champaign Institute
Zhejiang University
Hangzhou, China

Anke Ye
College of Civil Engineering and Architecture;
Zhejiang University/University of Illinois at Urbana–Champaign Institute
Zhejiang University
Hangzhou, China

Simon Hu *
Zhejiang University/University of Illinois at Urbana–Champaign Institute; College of Civil Engineering and Architecture
Zhejiang University
Hangzhou, China

***Abstract*—Meal delivery services provided by platforms with integrated delivery systems are becoming increasingly popular. This paper adopts a rolling horizon approach to solve the meal delivery routing problem (MDRP). To improve delivery efficiency in scenarios with high delivery demand, multiple orders are allowed to be combined into one bundle with orders from different restaurants. Following this strategy, an optimization-based four-stage heuristic algorithm is developed to generate an optimal routing plan at each decision point. The algorithm first generates bundles according to orders' spatial and temporal distribution. Secondly, we find feasible bundle pairs. Then, routes for delivering any single bundle or a bundle pair are optimized, respectively. Finally, the routes are assigned to available couriers. In computational experiments using instances from open datasets, the system's performance is evaluated in respect of average click-to-door time and ready-to-pickup time. We demonstrate that this algorithm can effectively process real-time information and assign optimal routes to the couriers. By comparing the proposed method with existing the-state-of-the-art algorithms, the results indicate that our method can generate solutions with higher service quality and shorter distance.**

Keywords—Meal delivery problem, pickup and delivery problem, rolling horizon method, heuristic algorithm.

I. INTRODUCTION

As the digital economy enables a variety of convenient daily services, the mobile app-based on-demand meal delivery market is booming worldwide. According to Statista [1], revenue in the online meal delivery industry is expected to grow at an annual rate of 9.9 percent from 2019 to 2023, bringing the market size to $53.786 billion. Grubhub, Deliveroo and Uber Eats are just a few examples of these online food delivery platforms, a business model that is rapidly gaining popularity around the world.

With the rapid expansion of the market, growing competition arises among meal delivery platforms on providing services with higher quality and lower costs. Meal delivery platforms often encounter the following challenges when dealing with a large number of orders in reality. On the one hand, orders cannot be picked up on time or even cancelled by limited service capacity. On the other hand, due to limited information perceived by couriers, a delivery route without optimization results in a longer detour time and thus lowers the system's service efficiency. Therefore, many strategies have been implemented to efficiently group the orders into bundles and then optimize the delivery routes accordingly. So that it can help improve the system's efficiency and reduce the cost.

The service requests of meal delivery routing problem (MDRP) are often related to two important characteristics — dynamism and urgency [2]. Dynamism is defined by the fact that requests and system decisions are placed over time. Urgency indicates the high time sensitivity of the orders, which should be assigned and delivered within a short time window. The precise definition of these important concepts keeps evolving. In this paper, we use the definition from van Lon et al. [3]. They found that there is a nonlinear relationship between dynamism, urgency, and "cost". This matches the work of Lund et al. [4] suggesting that as long as the optimization system receives requests before actual service time, despite a large number of dynamic requests, it is still possible to get a good solution.

A common strategy for solving such dynamic problems is incorporating rolling horizon technique in time-dependent model, i.e., optimizing tasks periodically [2, 5, 6]. The purpose of this study is to develop a heuristic algorithm for solving MDRP with a rolling horizon technique. To improve the delivery efficiency in scenarios with high service demand, the algorithm allows multiple bundles from different restaurants to be combined into a bundle pair and delivered on one route. In this paper, we limit the restaurant number of bundle pairs to two for the simplicity of the model.

In the existing literature, this Meal Delivery Routing Problem (MDRP) generally belongs to the category of Dynamic Vehicle Routing Problem (DVRP) and is closely related to the Dynamic Delivery Problem (DDP) [7]. It is one of the important classes of Dynamic Pickup and Delivery Problems (DPDP) that have emerged recently. In DPDP, the input data revealed over time is usually user requests. The solution strategy uses the dynamic information of requests to specify what actions should be performed over time [8]. As requests for on-demand services grow, many researchers are devoted to investigating on-demand dynamic delivery systems, not only

*Corresponding author. Email: simonhu@zju.edu.cn (Simon Hu)
This study was supported in part by the Zhejiang University Global Partnership Fund with the University of Sydney and University of Cambridge, the ZJU-UIUC Joint Research Centre Project of Zhejiang University (DREMES202001) and led by Principal Investigator Simon Hu.

from a theoretical [9,10,11], but also from a practical perspective [12,13,14].

Our algorithm is developed based on the meal delivery model proposed by [7]. The paper aims to find an exact solution to the delivery problem to minimize total service time of all orders with all order information known before optimization, which is far from reality. Our method is inspired by the pivotal work of [2]. One of the differences between our works lies in the generation of bundles. The bundle generation in [2]'s model is the result of a single-step optimization. In our model, we optimize bundles generation by considering orders' temporal and spatial distribution. In addition, our model focuses on minimizing the freshness loss and maximizing the delivery volume per unit of time, while their model concerns more about the cost of couriers.

The main contributions of this paper are threefold: i) We propose a heuristic algorithm to deal with bundle generation, bundle pairing, route optimization, and task assignment in MDRP. ii)A comparison with an existing algorithm and exact solution is conducted for assessing system efficiency. iii) Computational results of our algorithm are presented for evaluating algorithm performances.

The remainder of this paper is organized as follows: Section II describes the problem formulation. A heuristic algorithm is developed in Section III to find optimal solutions. Section IV conducts result analysis on algorithm performance and system performance. Section V draws conclusions and provides an outlook for future works.

II. Problem Formulation

Consider an on-demand meal delivery system served by a fleet of couriers, who are crowd-sourced freelancers and tend to work for a shift voluntarily. As orders arrive throughout a service day and order information is revealed dynamically, instructions are sent to couriers based only on known requests. In order to solve this dynamic problem, a rolling-horizon approach is adopted in this paper. We separate the whole service time evenly into multiple small time intervals of f minutes. The system is then optimized at the decision point t_{opt}, which is the end of each time interval.

For time interval $T = \{t: t_{opt} - f < t \leq t_{opt}\}$, orders placed in T or unserved in previous interval are waiting to be assigned at t_{opt}. R is the set of restaurants in the system. Each restaurant $r \in R$ has an associated location. U_r is the set of orders to be assigned at restaurant r. Taking order $o \in U_r$ as an example, its information includes placement timestamp P_o, associated restaurant R_o (which is r here), drop-off position D_o, and ready timestamp r_o. For courier c, information includes the start-timestamp S_c and the off-timestamp O_c of the day, as well as the current location of the courier L_c.

Two strategies are considered for improving delivery efficiency. On the one hand, orders from a same restaurant are allowed to be grouped into a bundle, denoted as b, and delivered on one route. Here we regard ready timestamp r_o of the latest ready order o in the bundle b as the bundle's ready timestamp r_b. Note that any bundle b can only be picked up after its ready timestamp r_b. On the other hand, up to two bundles from two different restaurants can be seen as a bundle pair and be delivered on one route.

A available courier that we can give instructions to at t_{opt} should be the courier c on duty ($S_c \leq t_{opt} \leq O_c$). In addition, he is neither in route to pick up orders nor delivering orders at this decision point. Note that a courier c cannot receive new instructions after his off-timestamp O_c but can keep delivering orders which are assigned before. After receiving instructions, couriers must take the bundles according to the instruction and deliver orders according to the optimized route. In this paper, we consider a fixed pick-up service time when a courier is picking up a bundle in a restaurant. This service time is independent of the number of orders in the bundle. Similarly, a drop-off service time is counted on a courier's arrival at a drop-off location. A flow chart to represent the problem sequence at a decision point is shown in Figure 1. Figure 2 demonstrates the timeline of events to deliver an order.

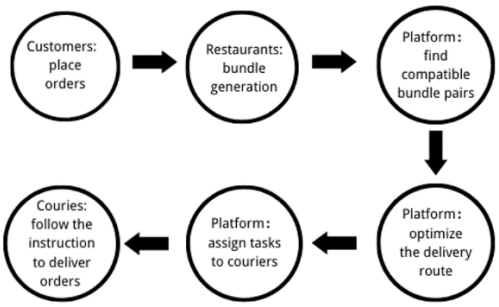

Fig. 1. Flow chart of problem sequence within one optimization

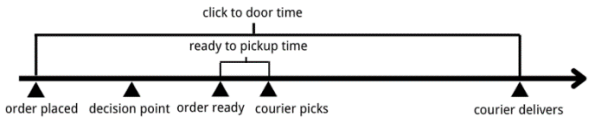

Fig. 2. Timeline of events to deliver an order

For the optimization at each decision point, we have two goals. The first is to improve delivery efficiency, which means that couriers can deliver more orders per unit time. The second is to reduce the freshness loss of orders.

For clarity, we list some important notations and related descriptions used in section III below:

TABLE I. Notation and Description

Notation	Description
U_r	Set of all orders which are placed in restaurant r
C	Set of all couriers available at this decision point
S	Set of bundles of all restaurants in the system
Q	Set of all single bundles and feasible bundle pairs
N_{total}	Total number of all orders placed in the system
$N_{o,r}$	Total number of all orders placed in the restaurant r

N_d	Number of available couriers who are available at this decision point
$N_{b,r}$	Number of bundles in a restaurant
$N_{o,q}$	Number of orders in a route q
r_b	Last ready timestamp of order in bundle b
r_o	Ready timestamp of order o
R_o	Associated restaurant of order o.
R_b	Associated restaurant of bundle b
D_o	Drop-off position of order o.
$\Pi_{q,c}$	Pick up timestamp if route q is assigned to courier c
T_q	Time to deliver all orders of route q according to the optimized route
β	Penalty parameter for delivery delay
α	Pick up delay tolerance time
θ	Penalty parameter for freshness loss
o	Order
r	restaurant
c	Courier in set C
q	Single bundle or bundle pair in set Q with an optimized route (we also call it a "route" in this paper)
t	Deliver sequence number in optimized route

III. A Heuristic Algorithm for the Meal Delivery Routing Problem

The algorithm for optimizing the delivery strategies can be divided into four steps. The first step is to combine unassigned orders into bundles for each restaurant. Secondly, the generated bundles are matched in pairs to find the feasible bundle pairs that can be picked up by one courier. Then, delivery routes of all generated bundles and bundle pairs are optimized to minimize delivery time. Finally, optimized routes are assigned to appropriate couriers.

A. Bundle Generation

To determine the number of bundles in restaurant r, we first give a rough estimation on the target bundle size K at t_{opt} as follows:

$$K = \frac{N_{total}}{N_d}$$

The value is then compared with the number of available couriers N_d at this decision point, and the smaller value is taken as the number of bundles $N_{b,r}$. This can help us avoid the situation that there are too many bundles to be taken by couriers. $N_{b,r}$ is defined as below:

$$N_{b,r} = \min\{\frac{N_{o,r}}{K}, N_d\}$$

With a known target bundle number $N_{b,r}$ of restaurant r, we follow the steps below to get a bundle list for the restaurant.

1) Generating a Draft List of Bundles Considering Spatial Dimension

Aggregation degree, adopted as a measure of cost to deliver a bundle, is defined as the sum of travel distance between the drop-off positions of any two orders in the bundle. First, we set up a binary variable $B_{b,o}$. If order $o \in U_r$ is in bundle b, $B_{b,o}$ = 1. Otherwise, $B_{b,o}$ = 0. Then we define the variable *Aggregation Cost*:

$$Aggregation\ Cost(b) = \sum_{o_1 \in U_r} \sum_{o_2 \in U_r} Distance(D_{o_1}, D_{o_2}) * B_{b,o1} * B_{b,o2}$$

A draft list of bundles considering spatial dimension is generated when the sum of aggregation cost of all bundles is minimized:

$$\text{Min } \sum_b Aggregation\ Cost\ (b) \quad (1a)$$

$$\text{s.t. } \sum_b B_{b,o} = 1, \forall o \in U_r \quad (1b)$$

Here (1b) indicates that each order can only appear in one bundle. According to value of the variable $B_{b,o}$, we can obtain a draft bundle list $B_{r,d}$ of the restaurant r. This list can tell us the order information in each bundle.

2) Further Optimizing Bundle List Considering Temporal Dimension

We have two objectives to consider in temporal dimension. One is to reduce the sum of delivery time of all bundles. The other is to reduce the sum of all orders' delay time. Combine these two objectives into one total objective function. Then we can optimize the draft bundle list $B_{r,d}$ in this Procedure 1.

Procedure 1: *Further Optimization*

Input: U_r, set of all orders which are placed in restaurant r.
$B_{r,d}$, the draft bundle list which is generated above.

Output: $B_{r,new}$, improved list of bundles.

/* Initial construction
$B_{r,new} \leftarrow B_{r,d}$
$Cost\ (b) = Sending\_time(b) + \beta * \sum_{o \in b} Predict\_delay(o)$
Temporal Cost = $\sum_{b \in B_r} Cost(b)$
For o ∈ U_r **do:**
    Remove o from its current bundle b_o
    List all bundles (which contains b_o), find a bundle b_{new} to re-insert o to achieve minimum *Temporal Cost*
    Update $B_{r,new}$ with inserting o into b_{new}
End
return $B_{r,new}$

3) Assign Priority Levels to Bundles

Based on the updated bundle list $B_{r,new}$ in Step 2, bundles are assigned priority levels according to the following rules:

(1) The bundles with more orders have higher priority levels.

(2) If two bundles with same number of orders, the one have earlier ready time has higher priority level..

When couriers are limited in the system, bundles with higher priority levels will be assigned to couriers first.

B. Combination of Feasible Bundle Pairs

Considering restaurants with close locations, bundles of two restaurants are allowed to be grouped together for a higher delivery efficiency of a route. A bundle pair is regarded as feasible if its second bundle doesn't suffer an excessive freshness loss in the route, i.e., the courier can get the second bundle within a tolerable delay time.

Procedure 2: Generate Feasible Bundle Pair

Input: *S*, set of all bundles in all restaurants,
α, pick up delay tolerance time ($\alpha \geq 0$)
Output: Set Q of singles bundles and bundle pairs to be assigned.

```
/* Initial construction
P ⟵ ∅
For b1 ∈ S do:
  For b2 ∈ S \ {b1} do:
    If r_b1 + TravelTime(R_b1, R_b2) ≤ r_b2 + α then:
      P ⟵ P ∪ {b1, b2}
    End
  End
End
Q ⟵ P ∪ S
return Q
```

C. Calculation of Shortest Delivery Time of Bundles and Bundle Pairs

For the previously generated bundles and bundle pairs, we need to optimize their delivery routes to achieve the shortest delivery time. For $\forall q \in Q$, we need to follow the below steps to optimize the route:

First, we sort the orders in each bundle and each bundle pair according to their ready timestamp, and label them accordingly. Then, we set up a two-dimensional binary matrix *R* to represent the route for delivery. The first dimension means the sequence of delivery. The second dimension means the tag of order, e.g., if order 1 is delivered as the second order in the route, *R* [2,1] = 1. The optimization problem is formulated as follows:

$$Min(\sum_{o\in q} TravelTime(r, D_o) * R[1, o] + \sum_{t=1}^{Nq} \sum_{o2\in q} \sum_{o1\in q} TravelTime(D_{o1}, D_{o2}) * R[t, o1] * R[t+1, o2]) \quad (2a)$$

$$\text{s.t. } \sum_{o\in q} R[t, o] = 1, \forall\, t \in [1, Nq] \quad (2b)$$

$$\sum_{t=1}^{Nq} R[t, o] = 1, \quad \forall o \in q \quad (2c)$$

Here constraints (2b) indicate that in each time sequence of delivery, the courier can deliver only one order. Constraints (2c) guarantee that each order can only be delivered once.

D. Assignment Model

Based on the set *Q* of available single bundles and bundle pairs and their optimized delivery route, tasks are assigned to available couriers. The goal of the assignment is to improve delivery efficiency and reduce the residence time of orders. Here the residence time of an order means the time difference between the pickup timestamp and its ready timestamp. We define the variable:

$X_{q,c} \in \{0,1\}$, indicates whether route q is assigned to courier c.

The delivery efficiency *E* of courier *c* delivering route $q \in Q$ is calculated as below:

$$E(q,c) = \frac{N_q}{(\Pi_{q,c+T_q} - T_{optimize})}$$

It's important to note here that no matter whether the courier picks up a single bundle or a bundle pair, here N_q denotes the total number of orders in the route *q*. We take the total time the courier needs to complete the task as the denominator. The optimization time is ignored here.

Freshness loss *FL* is measured by the difference between the picks up timestamp of the second bundle and the last ready timestamp in the second bundle. It is given by:

$$FL(q,c) = \Pi_{q,c} - \max_{o\in q}\{e(o)\}$$

Using these definitions, the optimization problem is formulated as:

$$\max \sum_c \sum_{q\in Q} [(E(q,c) - \theta * FL(q,c)] * X_{q,c} - \sum_o p * (1 - y_o) \quad (3a)$$

$$\text{s.t. } \sum_{q\in Q} X_{q,c} \leq 1, \forall\, c \in C \quad (3b)$$

$$\sum_{c\in C} X_{q,c} \leq 1, \forall\, q \in Q \quad (3c)$$

$$\sum_{c\in C} \sum_{q\in Q(o)} X_{q,c} \leq 1, \forall\, o \in U \quad (3d)$$

$$\sum_{c\in C} \sum_{q\in Q(o)} X_{q,c} = y_o, \forall\, o \in U \quad (3e)$$

Constraints (3b) indicate that a courier can only be assigned one route. Constraints (3c) guarantee that a route can only be assigned to one courier. Constraints (3d) guarantee that the assigned routes cannot contain the same order more than once. Constraints (3e) guarantee that if an order is not included in any of the routes assigned to real couriers, a penalty p is added to the objective value.

IV. Computational Study

In this computational study section, two indexed are adopted to assess the algorithms and the dynamic system's performance:

1) The average click-to-door time (aCtD) of all orders assigned to couriers in the system. Click-to-door time (CtD) of an order indicates the difference between its drop-off timestamp and its placement timestamp.

2) The average ready-to-pickup time (aRtP) of all orders assigned to couriers in the system. Ready-to-pickup time (RtP) of an order indicates the difference between its pickup timestamp and its ready timestamp of an order.

A. *Algorithm Performance in Different Data Size Insatnces*

We generate instances with order, restaurant and courier information based on open datasets provide by [15] to test the algorithm's performance. Table II shows the characteristics of the instances. Table III demonstrates the performance of the algorithm in different instances.

TABLE II. COMPARISON OF CHARACTERISTICS OF DIFFERENT INSTANCES

Instance	Number of Orders	Number of Couriers	Number of Restaurants
S1	20	8	4
S2	40	16	8
S3	60	24	12
S4	100	40	20
S5	200	80	40

TABLE III. COMPARISON OF ALGORITHM PERFORMANCE OF DIFFERENT INSTANCES

Instance	Optimization Time (s)	CtD (min)	RtP (min)
S1	1.67s	34.00	2.40
S2	5.26s	33.83	1.43
S3	12.34s	32.97	2.02
S4	40.64s	32.88	2.64
S5	186.71s	32.81	2.66

As can be seen from Table III, the optimization time do exponential growth as the number of orders increase. In the five instances, the CtD and RtP lie around 33 minutes and within 3 minutes, respectively. The problem scale has no obvious impacts on the two indexes. In addition, we found that CtD has a slight advantage in instances with more orders.

B. *System Performance*

Instances with service period of 900 minutes are selected for testing the algorithm in dynamic processes. The full introduction of the datasets can be found in [15]. For clarity, we list the key characteristics of the instances in Table IV.

TABLE IV. COMPARISON OF CHARACTERISTICS OF DIFFERENT INSTANCES

Instance	Number of orders	Preparation time multifaction factor	Number of restaurants	Courier velocity (m/min)
D1	252	1	93	320
D2	252	1.25	93	320
D3	252	1	93	427
D4	242	1	54	320
D5	242	1.25	54	320
D6	505	1	116	320

In order to study the impact of different optimization time frequencies on the system, four instances are optimized with time intervals of 5 minutes, 10 minutes, and 15 minutes, respectively.

TABLE V. COMPARISON OF SYSTEM PERFORMANCE WITH VARIED OPTIMIZATION TIME INTERVALS

Time interval	5 minutes		10 minutes		15 minutes	
Instance	aCtD (min)	aRtP (min)	aCtD (min)	aRtP (min)	aCtD (min)	aRtP (min)
D1	30.83	1.94	31.43	2.65	32.65	3.87
D2	33.94	1.93	34.57	2.64	35.72	4.25
D3	27.94	1.50	29.81	2.29	29.91	2.83
D6	31.83	2.43	32.35	2.89	32.74	3.58

From Table V, we can find that a decrease in optimization time interval leads to shorter aCtD and aRtP in all the instances. Therefore, we select 5 minutes as the default time interval hereinafter.

Fig. 3 and Fig. 4 present the distribution of CtD and RtP in four instances, respectively. Fig. 1 illustrates that in most time intervals, orders can be delivered to customers between 20 to 40 minutes. As depicted in Fig. 2, the RtP is within three minutes for most of the time intervals. However, orders in a few time intervals suffer from a severe freshness loss due to high demand and courier shortage in peak hours.

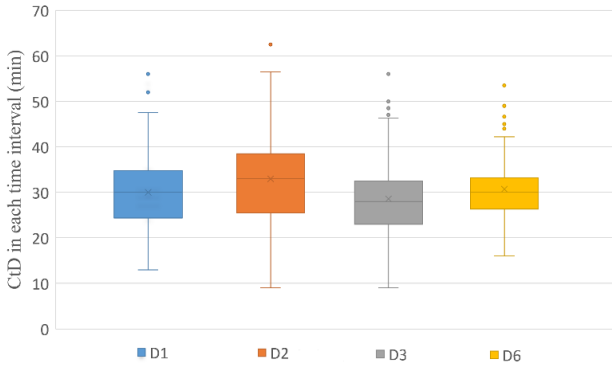

Fig. 3. Distribution of average click-to-door time in different time intervals in four instances

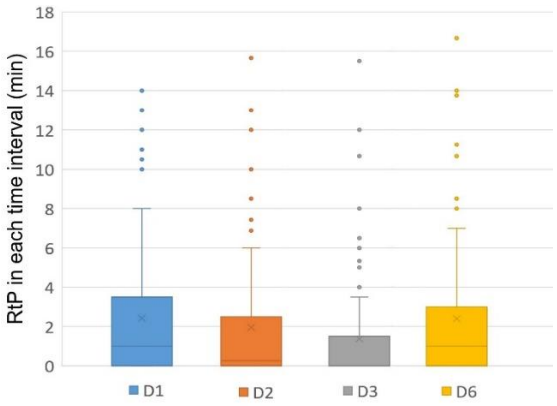

Fig. 4. Distribution of average ready-to-pickup time in different time intervals in all instances

C. Algorithm Comparison

A comparison of the solution obtained by our algorithm and an existing algorithm proposed by [2] is conducted. The results in Table VI indicate that our algorithm improves aCtD and aRtP of the system slightly. We also compare the solutions obtained by our algorithm and the exact solution proposed by [4]. The details are demonstrated in Table VII.

TABLE VI. COMPARISON OF SOLUTIONS OBTAINED BY OUR ALGORITHM AND EXISTING ALGORITHM

	Four-stage heuristic algorithm		**Algorithm in [3]**	
Instance	**aCtD (min)**	**aRtP (min)**	**aCtD (min)**	**aRtP (min)**
0o50t100s1p100 (D1)	30.83	1.94	31.19	2.52
0o50t100s1p125 (D2)	33.94	1.93	34.67	2.27
0r50t100s1p100 (D4)	31.41	2.11	32.46	2.14
0r50t100s1p125 (D5)	35.88	1.41	36.75	2.16

TABLE VII. COMPARISON OF SOLUTIONS OBTAINED BY OUR ALGORITHM AND EXACT SOLUTION

	Four-stage heuristic algorithm		**Exact solution in [4]**	
Instance	**aCtD (min)**	**aRtP (min)**	**aCtD (min)**	**aRtP (min)**
0o50t100s1p100 (D1)	30.83	1.94	29.81	1.46
0o50t100s1p125 (D2)	33.94	1.93	33.40	1.23
0r50t100s1p100 (D3)	31.41	2.11	30.76	1.13
0r50t100s1p125 (D4)	35.88	1.41	34.66	0.97

V. CONCLUSION

This paper proposed a four-stage rolling horizon optimization algorithm to solve MDRP. The method can be employed based on real-time order information. The algorithm first generates bundles according to orders' spatial and temporal distribution. For improving delivery efficiency, up to two bundles are allowed to be delivered on one route. Thus, secondly, we find feasible bundle pairs. Then, routes for delivering any single bundle or a bundle pair are optimized, respectively. Finally, the routes are assigned to available couriers.

Through result analysis, we can conclude that our algorithm is able to solve MDRP with up to 200 orders within about 3 minutes. Additionally, slightly improved service quality is observed in scenarios with higher demand. Through a comparison of the dynamic system solved with different optimization frequencies, smaller aCtD and aRtP are obtained when reducing the time interval. According to the distribution of aCtD and aRtP of orders, we find that orders in most time intervals can be picked up and delivered in time. A comparison of our algorithm with the existing algorithm and exact solutions is conducted as well. Our algorithm is able to produce more efficient and accurate solutions than the existing algorithm.

As limitations of this work, the heuristic algorithm considers no load capacity of couriers, which plays a great role when serving a huge number of orders in reality. Thus, we will include this constraint to improve the algorithm in the future.

REFERENCE

[1] J. Wang, X. Shen, X. Huang, and Y. Liu, "Influencing Factors of the Continuous Usage Intention of Consumers of Online Food Delivery Platform Based on an Information System Success Model," *Frontiers in Psychology*, vol. 12, Aug. 2021.

[2] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil. The meal delivery routing problem. Optimization Online, 2018.

[3] R. R. S. van Lon, E. Ferrante, A. E. Turgut, T. Wenseleers, G. Vanden Berghe, and T. Holvoet, "Measures of dynamism and urgency in logistics," *European Journal of Operational Research*, vol. 253, no. 3, pp. 614–624, Sep. 2016.

[4] Karsten Lund, Oli B.G. Madsen, and Jens M. Rygaard. "Vehicle routing problems with varying degrees of dynamism." Technical Report IMM-REP-1996-1, Department of Mathematical Modeling, The Technical University of Denmark, May 1996.

[5] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, "Making dynamic ride-sharing work: The impact of driver and rider flexibility, "Transportation Research Part E: Logistics and Transportation Review, vol.91, pp. 190 -- 207, Jul. 2016.

[6] X. Wang, N. Agatz, and A. Erera, "Stable Matching for Dynamic Ride-Sharing Systems," SSRN Electronic Journal, 2014.

[7] 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.

[8] G. Berbeglia, J. Cordeau and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, (1), pp. 8-15, 2010.

[9] C. Archetti, D. Feillet, and M. G. Speranza, "Complexity of routing problems with release dates," *European Journal of Operational Research*, vol. 247, no. 3, pp. 797–803, Dec. 2015.

[10] M. W. Ulmer, B. W. Thomas, and D. C. Mattfeld, "Preemptive depot returns for dynamic same-day delivery," *EURO Journal on Transportation and Logistics*, Apr. 2018.

[11] M. A. Klapp, A. L. Erera, and A. Toriello, "The One-Dimensional Dynamic Dispatch Waves Problem," *Transportation Science*, vol. 52, no. 2, pp. 402–415, Mar. 2018.

[12] I. Dayarian and M. Savelsbergh, “Crowdshipping and Same-day Delivery: Employing In-store Customers to Deliver Online Orders,” *Production and Operations Management*, vol. 29, no. 9, Jun. 2020.

[13] S. A. Voccia, A. M. Campbell, and B. W. Thomas, “The Same-Day Delivery Problem for Online Purchases,” *Transportation Science*, vol. 53, no. 1, pp. 167–184, Feb. 2019.

[14] N. Azi, M. Gendreau, and J.-Y. Potvin, “A dynamic Vehicle routing problem with multiple delivery routes,” *Annals of Operations Research*, vol. 199, no. 1, pp. 103–112, Oct. 2011.

[15] “Meal Delivery Routing Problem Test Instances.” GitHub. https://github.com/grubhub/mdrplib (accessed Nov. 11, 2022).