Meal Delivery Routing with Crowd-sourced Vehicles

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
COMPUTER SCIENCE: ARTIFICIAL INTELLIGENCE

Author :	Zhe Deng
Student ID :	2696266
Supervisor :	Dr. Y. Fan
2nd corrector :	Dr. B. van Stein

Leiden, The Netherlands, August 31, 2022

Meal Delivery Routing with Crowd-sourced Vehicles

Zhe Deng

Leiden Institute of Advanced Computer Science (LIACS)
Snellius Building, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

August 31, 2022

Abstract

Meal box delivery can be regarded as the ultimate challenge of last-mile logistics: a typical order is anticipated to be delivered in a short time soon after the food is ready. The order arrival flow is extremely unpredictable and volatile. The crowd-sourced delivery fleets are free to decline requests as well as log in and out at will. Current research described the problem and provided feasible solutions. However, their formulations touch upon some resources about uncertainty and the methods are not that comprehensive. In this research, we implement a model which focuses on assigning the orders to crowd-sourced couriers using mixed-integer programming with time windows. Besides, we integrate the model into a computational framework, where it is simulated in Open Source Routing Machine. A more advanced row generation assignment model and a two-stage set-partitioning model are studied as well. The model describes the problem more accurately and myopically matches the couriers' routes with the orders with the use of information at the current time point. The results of the model simulation with real-life instances meet all the time criteria of performance metrics. The rates of order fulfilment are over 50%. Compared with the Meal Delivery Routing network flow model, it reduces the average computation time by 14% on all instances, while achieving a 7% higher average fill rate.
Additionally, it offers valuable insights into demand management as well as the potential benefits of crowd-sourced delivery.
keywords: on-demand meal delivery, crowd-sourced delivery, dynamic vehicle routing ,integer programming, matching

Contents

1 **Introduction** **1**

2 **Related Work and Literature Review** **3**
2.1 Vehicle Routing Problem Variants 3
2.2 Meal Delivery Routing Problem 5
2.3 Crowd-sourced Delivery 6

3 **Problem Description and Mathematical Formulations** **9**
3.1 MDRP-based formulations 9
3.2 Set partitioning formulations 11
3.3 Assumptions 12

4 **Method** **13**
4.1 CV Routes Generation 13
4.2 Assignment Logic 14
4.2.1 Row generation 15
4.3 Matching Mechanism 17
4.3.1 Scope limit 17
4.3.2 Commitment strategy 17
4.3.3 Matching policy 18
4.3.4 DV Routes as Supplements 18

5 **Computational Study** **19**
5.1 Instances 19
5.2 Implementation 21
5.3 Experiment 22
5.4 Result 23
5.4.1 Performance Metrics 23

5.4.2 Computational Efficiency 27

6 Discussion 31

A Notations 37

B Simulation Settings 39

C Computer and Software Information 41

Chapter 1

Introduction

Meal delivery has been gaining popularity with the development of online retailing. In particular, work-from-home requirements and preferences during these pandemic years intensified the need for online ordering and on-demand delivery. The meal delivery routing problem (MDRP) which falls under a significant new category of dynamic delivery operations is introduced and formulated by Reyes et al.[1] It is a dynamic deterministic model of how meal delivery systems are set up and work, which contains structural assumptions and some basic performance metrics. Apart from the MDRP itself, an algorithm to solve it in near real-time is mentioned.

The growth of E-commerce increased demand for last-mile delivery and the level of congestion in transportation in urban areas. Crowd-sourced vehicles provide the additional capacity needed to meet the growing demand in a cost-effective way. Crowd-source vehicles play an extremely important role in meal-box delivery, which does not go against the will of people to focus on getting reliable delivery at the right time at a reasonable cost.

The earliest solution algorithms to MDRP used the linear relaxation model that mitigates uncertainty into the postponement of decisions that are not time-critical. Later, provably high-quality solutions involving a simultaneous column- and row-generation method are proposed. Since the meal delivery routing problem is highly dynamic where several variables interact with each other, there is great potential to extend the formulation and introduce a more complex model as well. More complex models that could be considered are supposed to involve more factors about the sources of uncertainty. Considering the development status and prospects, it would be of great value and interest to focus on models that rely on crowd-sourced delivery and explore the potential of using extended formulations in a rolling horizon framework.

Existing methods and models can describe the problem and solve it, but as the times change, we need new and more realistic modelling and more advanced solutions that can balance the quality and the cost.
On the basis of completing order allocation, our research explores more advanced and complex algorithms, to reduce the cost as much as possible. We involve the crowd-sourced vehicles in delivery and try to formulate the problem with procedure decomposition. About the assignment model, a mixed-integer model and a row generation model are explored. The MIP model is implemented and merged into the computational framework for simulation, and it works together with myopic matching policy and scope limit to achieve the goal of a higher order fulfilment rate in a shorter time.

This chapter 1 contains the introduction; In Chapter 2 we discuss related work; In Chapter 3 we describe the problem and formulate it with mathematical equations; In Chapter 4 we express the methods we use and some background knowledge we apply; In Chapter 5 we will explain the experiments including the implementation, simulation and results as well; Results and conclusions will be discussed in Chapter 6 as well as avenues for future research.

Chapter 2

Related Work and Literature Review

The Meal Delivery Routing Problem (MDRP) belongs to the large class of Vehicle Routing Problems (VRP). Moreover, MDRP is highly dynamic with pickups and deliveries, which means that it belongs to Dynamic Vehicle Routing Problems (DVRP), specifically the class of Dynamic Pickup and Delivery Problems (DPDP). Considering the urgent need of customers and the rising service offered by online retailers, the problem has been extended and associated with various aspects of the same-day delivery problem (SDDP).

2.1 Vehicle Routing Problem Variants

The vehicle routing problem (VRP) was originally introduced by Dantzig and Ramser (1959)[2], evolving consistently into heterogeneous variants currently analyzed in the literature (Toth and Vigo, 2014[3], 2002[4];Golden et al., 2008[5]; Eksioglu et al., 2009[6]). In the VRP literature, vehicles are generally divided into two main categories: homogenous (Lin et al., 2014[7]; Marinaki and Marinakis, 2016[8]) and heterogeneous. Heterogeneous transportation systems were introduced by Golden et al. (1984)[9] and generally account for capacity differences across vehicles (Siddiqui and Verma, 2015[10]; Talebian and Salari, 2015[11]; Lai et al., 2016[12]). The VRP generalizes the well-known travelling salesman problem (TSP). and bin packing problem (BPP). It is a combinatorial optimization and integer programming problem to find the optimal path for delivering a product to a given customer location. Determining the optimal solution for VRP is NP hard.

There are a large number of researchers have studied these problems from different angles, and we focus on DVRP and DPDP.

Pillac et al. (2013)[13] propose a framework that is driven by events to optimize the DVRP parallelly and provide an overview of the DVRP, in which the problems are divided according to the evolution and quality of information. Later, Psaraftis et al. (2016)[14] compile a comprehensive review that classifies the DVRP literature by different features (for example type of problem, mode of transportation, objective function, constraints on time and capacity, stochasticity, solution method and etc.).

Berbeglia et al. (2010)[15] survey the DPDP literature where the solution strategies and the assessment of algorithmic performance are discussed. Mitrovic-Minic et al. (2004)[16] propose a double-horizon algorithm that evaluates the actions by different cost functions if the actions occur within a given horizon (short term) or beyond that (long term). It tries to balance the fulfilment of current tasks and the preservation of flexibility to complete future and unknown tasks. In the meantime, on when to delay and commit to making decisions to reduce uncertainty, they explore four waiting strategies for a DPDP with time windows: drive-first, wait-first, dynamic waiting and advanced dynamic waiting. The basis of studying dynamic delivery systems is to perceive dynamism and urgency, as most of the requests during the period of operations need to be completed within a relatively short time window. With regards to this, Lund et al. (1996)[17] suggested measuring the degree of dynamics based on the proportion of dynamic queries. In addition, Larsen et al. (2002)[18] refined the definition and and the real measure of dynamics was developed. This measure sought to encompass both the urgency and the evolution of information in a single unit. However, van Lon et al. (2016)[19] have corrected this by using two independent criteria to measure dynamism and urgency separately to avoid previous drawbacks. These definitions are adopted by Reyes et al. (2018)[1] while proposing the MDRP. More recently, a dynamic waiting model is introduced by Yan et al. (2020)[20] where dynamic pricing helps solve a ride-hailing problem out of Uber data.

Compare with general DPDP, SDDP usually has a shorter time limitation and the dedicated vehicles should return to the depot after delivery. The SDDP was presented by Voccia et al. (2015)[21] as well as a solution with a Markov decision process (MDP), and the SDDP problem itself has been receiving much attention soon after. The problem is highly dynamic as future information is incorporated into route planning. Klapp et al.

(2016)[22] use a simple setup to study the core trade-off encountered in same-day delivery, that is to dispatch a car to deliver a known order immediately, or wait for other orders to arrive. Waiting for additional orders may result in a lower-cost route, however, reduce the delivery flexibility of known orders at the same time because of less delivery time available, which would may raise the cost in turn. Three types of solution methods are discussed: a priori plan, a roll-out strategy and a more involved strategy engaging approximate linear programming, among which the approximate linear programming is almost of the same efficiency as the roll-out strategy. Then, a sample scenario planning method is proposed by Voccia et al. (2017).[23] It takes advantage of the sampled information to design a consensus function specifically, which could identify when it is beneficial to wait at the site in anticipation of future requests thus helping generate the vehicle routing. Next, Klapp et al. (2018)[24] formulate the dynamic dispatch waves problem (DDWP). In DDWP, orders arrive at the depot dynamically and are supposed to be delivered on the same day. Information related to the solution is deterministic. Waves, that is moments in time where vehicles can be dispatched or stopped, play an important role in decision making.

2.2 Meal Delivery Routing Problem

Similar to SDDP, MDRP formally introduced by Reyes et al. (2018)[1] is also faced with a series of challenges of delivery by online services. In the MDRP paper, a myopic rolling horizon matching-based algorithm is used to solve the assignment logic by only prescribing the next pickup and delivery for each courier. Bundles are allowed which means single orders could be picked up together and then delivered by one courier on a specified route. Linear and more complex integer programming assignment models are considered, and a two-stage additive commitment to mitigate uncertainty by postponing decisions according to the priority scheme as well. Almost at the same time, Ulmer et al. (2017)[25] introduce the Restaurant Meal Delivery Problem (RMDP), which involves stochastic cook time and thus the solution of it includes a cost function approximation on an MDP model.

Yildiz and Savelsbergh (2019)[26] develop a simultaneous column and row generation (CR) algorithm for the solution of a novel MDRP formulation that assumes perfect information regarding order arrivals. Besides, an enumeration algorithm (PS) to identify such columns is proposed as well as a branch-and-price (BP) algorithm. The results of experiments on solving the linear relaxation using the selective column inclusion (SCI) scheme for the instances remark the viability of the algorithmic thoughts to develop a dynamic dispatching heuristic. Their findings highlight that it is important to size and schedule delivery capacity, and show there is a great potential for retargeting-focused requirements management strategy. In an order of one single customer, there may involve multiple restaurants. To tackle such a situation, Steever et al. (2019)[27] define the Virtual Food Court Delivery Problem (VFCDP) and show that the approach incorporated with look-ahead policies outperforms the purely myopic one on the instances artificially created by probability distributions. They provide a Mixed Integer Programming (MIP) formulation to this and combine it with a heuristic based on an auction for sake of decision-making with future demand. It has been also proved that proper predictions on arrival time improve the system performance by Hildebrandt and Ulmer (2020).[28]

2.3 Crowd-sourced Delivery

The development of crowd-sourced delivery has been getting more intentions these years. People turn to be more concerned about ensuring dependable delivery at the appropriate time and price, instead of mainly finding as many optimal routes with available and effective delivery capacity as possible. To achieve this, crowd-sourced delivery is considered more. Static models treat crowd-sourced shipping as VRP. Archetti et al. (2016)[29] define the form of crowd-sourced delivery as VRP with occasional drivers (OD). They propose a static deterministic model where professional vehicles (PV) are assigned with closed routes and OD may visit only one customer before heading to their destinations without return. Other extensions to this model emerge. Macrina et al. (2017)[30] allow OD to deliver multiple packages to the transshipment nodes. Dahle et al. (2019)[31] formulate the problem with time windows and focus on compensation schemes to balance reducing the total cost and incenting the willingness of drivers. Liao et al. (2020)[32] object to finding a green way with minimal carbon footprint by using a genetic algorithm (GA) and clustering with principal component analysis (PCA) to initialize the routes and optimize by adaptive large neighbourhood search (ALNS) afterwards.

In natural, crowd-sourced delivery is under great stochasticity since uncertainty originates from plenty of aspects. That is the reason why the dynamic version of the crowd-sourced delivery is being taken into account. Gdowska et al. (2018)[33] view the problem as a bi-level stochastic problem where OD still visit a single customer but can reject a task allocation with a given probability. Arslan et al. (2019)[34] introduce a rolling horizon method that could solve the problem of matching tasks with the ad-hoc drivers on the fleet iteratively. The literature mentioned above focuses on the dynamism of one side, while dynamic aspects of both customers and drivers are considered in the paper written by Dayarian and Savelsbergh (2020).[35] Approaches including a myopic one and SSP are used to guarantee the quality of service. Tabu search is adopted for the routing of PV and OD, and PV has no limitation on capacity while OD could visit two customers at most. As for delivery systems with crowd-sourced drivers, Sampaio et al. (2020)[36] frame the problem as PDP with transfers. An ALNS algorithm is introduced so that favourable transfer opportunities are efficiently identified and driver operations are synchronized. Apart from modelling the crowd-sourced delivery problem with a set partitioning formulation, Arslan et al. (2019)[34] propose the decomposition heuristic (D-H) algorithm. With the help of it, the instances could be dealt in a large scale. Lately, Torres et al. (2022)[37] consider a two-stage stochastic programming model with recourse. Set partitioning performs at the first stage, and then the subset of the routes with compensations is assigned to crowd vehicles when they are available. The remaining routes as well as the routes that fail by crowd vehicles (at a penalty) are dispatched to professional vehicles in the second stage recourse. They modify the set covering model as the master problem using column generation and branch and price so that the formulation is strengthened by deriving upper bounds.

From the above literature review, we could conclude that for stochastic VRPs, uncertainty comes from regular sources (demands of customers, service, travel time including the weather and road conditions) and other sources (characteristics of fleet including vehicle types, if crowd-sourced delivery is involved). A model that is close to reality should be robust and flexible to the changes, which means it could make good use of recent and future information from time to time, make relatively fast and accurate decisions, and leave some probability for optimizations as well as responses to accidental events.

Chapter 3

Problem Description and Mathematical Formulations

There are two ways to formulate the problem: one is the traditional VRP based MDRP formulations, and the other one uses the set partitioning. The glossary for all sets, variables and parameters are in the notation table A.1 in Appendix A.

3.1 MDRP-based formulations

$$\min \Theta_1 = \begin{array}{l} \sum_{v\in C}(z_v(\sum_{w\in W(v)} p_1 d_w x_w^v) + p_2(\sum_{w\in W(v)} x_w^v - b)) \\ \qquad + \sum_{v\in D}\sum_{w\in W(v)} p_3 d_w x_w^v + \sum_{v\in D} F_d y_v \end{array} \tag{3.1}$$

$$\text{s.t.} \sum_{v\in V}\sum_{w\in W(v)\cap W(o)} x_w^v = 1 \qquad \forall o \in O, \tag{3.2}$$

$$\sum_{\substack{w\in W(v)\\ s_w=i}} x_w^v - \sum_{\substack{w\in W(v)\\ f_w=i}} x_w^v = \begin{cases} 1, & \text{if } i = \ell_v, \\ -1, & \text{if } i = \overline{\ell}_v, \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in V, i \in N^v \backslash \{h\}, \tag{3.3}$$

$$\sum_{v\in C}\sum_{\substack{w\in W(v)\\ s_w=\ell_o\\ \sigma_w\leq t}} x_w^c + \sum_{v\in C}\sum_{\substack{w\in W(v)\\ f_w=\ell_o\\ \phi_w>t-u^o/2}} x_w^c \leq 1 \qquad \forall o\in O, t\in T_o, \tag{3.4}$$

$$\sum_{\substack{w\in W(v)\\ s_w=h}} x_w^v - \sum_{\substack{w\in W(v)\\ f_w=h}} x_w^v = 0 \qquad \forall v\in D, \tag{3.5}$$

$$y_v \geq \sum_{\substack{w\in W(v)\\ s_w=h}} x_w^v \qquad \forall v\in D, \tag{3.6}$$

$$\sum_{w\in W(v)} x_w^v \leq z_v \times q_c \qquad \forall v\in C, \tag{3.7}$$

$$\sum_{w\in W(v)} x_w^v \leq y_v \times q_d \qquad \forall v\in D, \tag{3.8}$$

$$\tau_o \geq \sum_{v\in V}\sum_{w\in W(v)\cap W(o)} \delta_w^o x_w^v \qquad \forall o\in O, \tag{3.9}$$

$$\tau_o \leq a_o + \varrho^{max} \qquad \forall v\in V, \forall o\in O, \tag{3.10}$$

$$x_w^v \in \{0,1\} \qquad \forall v\in V, \forall w\in W(v), \tag{3.11}$$

$$z^v \in \{0,1\} \qquad \forall v\in C, \tag{3.12}$$

$$y^v \in \{0,1\} \qquad \forall v\in D, \tag{3.13}$$

$$\tau_o \geq 0 \qquad \forall v\in C, \forall o\in O. \tag{3.14}$$

In the MDRP-based formulation, the object function aims to minimize the total monetized cost of delivery meal boxes using crowd-sourced vehicles (CV) and dedicated vehicles (DV). The first half multiplied by the indicator variable includes the price based on distance and the compensation over the threshold on the number of work packages for CVs. The other one is composed of the total distance cost of DVs as well as the fixed cost of using DVs.

Constraints (2) to (8) are for the routing problem. Constraint (2) ensures that each order is served. The constraints in Equation (3) are to balance the flow at each node thus ensuring the spatial consistency of work packages performed by couriers in succession. Analogously, constraint (4) ensures temporal consistency of work packages executed by couriers continuously by enforcing that a work package can start at the location ℓ_o at some time t, provided that there is another package whose job ends at ℓ_o before t minus half of the service time required to place the order. The service time is the time required for arrival to the vehicle after delivery. The start times of work packages are limited to continuous time intervals rather than discrete points in time, which implies that variables and constraints are infinite, but it is sufficient to consider only a limited subset of work package variables and time-consistency constraints. Constraint (5) guarantees a DV is supposed to return to the depot after leaving it for delivery. The DV usage constraint (6) indicates that only DVs existing in the depot can be activated to serve the work packages. Constraints (7) and (8) are for capacity, and they regulate the number of work packages that can be delivered by a single CV or DV.

Constraints (9) and (10) are time window constraints with which the click-to-door time is guaranteed not to exceed the given maximum. The constraints in Equations (11) to (13) are binary for the decision variables, while the constraint in Equation (14) is non-negative.

3.2 Set partitioning formulations

$$\min \Theta_2 = \begin{aligned} &\sum_v \sum_r \sum_w p_1 d_w y^c_{r,v} + p_2 (\sum_v \sum_r \sum_w \alpha^c_{r,w,v} y^c_{r,v} - b) \\ &+ \sum_v \sum_r \sum_w p_3 d_w y^d_{r,v} + F_d \sum_v \sum_r y^d_{r,v} \end{aligned} \tag{3.15}$$

$$\text{s.t.} \sum_v \sum_w \alpha^c_{r,w,v} \times y^c_{r,v} + \sum_v \sum_w \alpha^d_{r,w,v} \times y^d_{r,v} = 1 \qquad \forall r \in \{N_w\}, \tag{3.16}$$

$$\sum_r y^c_{r,v} = 1 \qquad \forall v \in C, \tag{3.17}$$

$$\sum_r y^d_{r,v} \leq 1 \qquad \forall v \in D, \tag{3.18}$$

$$y^c_{r,v} \in \{0,1\} \qquad \forall (r,v) \in R, \tag{3.19}$$

$$y^d_{r,v} \in \{0,1\} \qquad \forall (r,v) \in R, \tag{3.20}$$

$$\alpha^c_{r,w,v} \in \{0,1\} \qquad \forall (r,v) \in R, \forall w \in W(v), \tag{3.21}$$

$$\alpha^d_{r,w,v} \in \{0,1\} \qquad \forall (r,v) \in R, \forall w \in W(v). \tag{3.22}$$

The object function (15) of set partitioning is similar to the previous formulation and also aimed to minimize the total cost of all types of vehicles.

Each work package must appear once and only once on all routes of the two types of vehicles, which is shown in the constraint (16). Constraint (17) ensures that only one feasible route for each CV is selected as part of the optimal one, while Constraint (18) indicates that for each DV at most one feasible route should be utilized. The constraints in Equations (19) to (22) are for the binary decision variables.

3.3 Assumptions

For the sake of thoroughness, structural assumptions are made.

Any route only has a certain number of orders.
After off time, couriers are unable to receive new instructions but are still able to complete any open tasks.
Every user has an order associated with it, but a user can only place one order per day.
One notification of order allocation can only be sent to a courier at once.
Only one courier may be given an order. The courier's vehicle is unchanged over time.
The travel time between any pair of locations varies with time.
A courier moves and complies with traffic signals while adhering to the actual street layout of the city.
Different vehicle types travel at various speeds.

Chapter 4

Method

In general, our solution method is inspired by the procedure of Decomposition Heuristic [34] and the overview can be described as the algorithm below:

1. CV routes initialization and generation (1^{st} stage):
   Assign as many as possible work packages to all feasible CVs;

2. DV routes as supplement(2^{nd} stage):
   Generate routes for DVs to deliver the work packages not served in the previous stage;
   Switch some work packages from CV to DV for some specific reasons;

3. Re-routing and optimization;

4.1 CV Routes Generation

In order to match the routes to crowd-sourced vehicles, orders need to be bundled as work packages into routes. We define a target route size S_t shown in Equation(4.1), which is adapted from Reyes et al. (2018)[1] for the ease to calculate routes.

$$S_t = max[\lceil \frac{|\{o \in O_t : e_o \leq t + \Delta\}|}{|\{v \in C_t : s_v = \emptyset\}|} \rceil, S_{max}], \Delta \geq f \tag{4.1}$$

where e_o is the ready time of order o in O_t which refer to unassigned orders and C_t are the idle crowd-sourced vehicles for picking up at time $t \in T$. A specific value for Δ is set through a tuning procedure and not less than the size of the rolling horizon f.

Then, the set of routes R can be generated by using parallel insertion algorithm 1.

Algorithm 1 Routes generation with parallel-insertion

```
Require: $O_t, C_t, S_t$
  for $u \in U$ do                                              ▷ for each restaurant
      $O_u \leftarrow \{o, \forall o \in O_t : u_o = u\}$        ▷ get orders for the restaurant
      $O^*_u \leftarrow \{O_{u(i)}, \forall i \in |O_t| - 1 : e_{O_{u(i)}} \leq e_{O_{u(i+1)}}\}$
                                         ▷ sort orders in increasing order of ready time
      $C_u \leftarrow \{v, \forall v \in C_t : h_{\ell_v, \ell_u} \leq d\}$
                         ▷ get CVs linked to the restaurant according to the distance
      $n_r \leftarrow max(\lceil \frac{|O^*_u|}{S_t} \rceil, |C_u|)$      ▷ calculate the number of routes
      $R \leftarrow \{\emptyset, \forall i = 0, ..., n_r - 1\}$    ▷ initialize routes with empty-sets
      if assignment updates allowed then
          $C^*_u \leftarrow \{v, \forall v \in V_u : r_v \neq \emptyset \wedge u_o = u, \forall o \in r_v\}$ ▷ get vehicles with
  non-empty routes of which pickup location is the restaurant
          $R \leftarrow R \cup \{r_v, \forall v \in C_u\}$      ▷ update the vehicle's assignments by
  appending vehicle routes
      end if
      for $o \in O^*_u$ do
          Find the route $r \in R$ and the insertion position $i_r$ for the order
  which minimizes increment of the route cost
          if route efficiency decreases by insertion then
              Ignore $r$ and find the next best route and insertion position
          end if]
          $r^{i_r} \leftarrow o$              ▷ insert the order o at position $i_r$ of the route $r$
      end for
  end for
  return $R$
```

4.2 Assignment Logic

After the routes are generated, the work packages containing bundles of orders need to be assigned to CV routes. Instead of matching work packages to individual CVs simply, we assign them to the routes of CVs generated in the previous procedure.

The **CV route assignment formulation** which shows the process of assigning work packages to CV routes is as follows:

$$\max_{x_{w,r,v}} \Theta_3 = \sum_v \sum_r \sum_w \mu \alpha_{w,r,v} x_{w,r,v} - \sum_v \sum_r c_{r,v} y_{r,v} \tag{4.2}$$

$$\text{s.t.} \sum_r y_{r,v} \leq 1 \qquad \forall v \in C, \tag{4.3}$$

$$\sum_v \sum_r x_{w,r,v} \leq 1 \qquad \forall w \in W, \tag{4.4}$$

$$\sum_w x_{w,r,v} \leq y_{r,v} q_c \qquad \forall (r,v) \in R, \tag{4.5}$$

$$x_{w,r,v} \in \{0,1\} \qquad \forall w \in W, \forall (r,v) \in R, \tag{4.6}$$

$$y_{r,v} \in \{0,1\} \qquad \forall (r,v) \in R. \tag{4.7}$$

4.2.1 Row generation

As mentioned earlier, time information for dispatching is embedded in the variables and constraints ensure time consistency in the MDRP-based formulation. To solve it perfectly, almost infinite variables and constraints are required. It is not possible to directly solve these formulations, however, it can be solved by using a Bender decomposition algorithm efficiently.

Benders decomposition is also known as row generation, since it adds new rows (constraints) as it progresses toward the solution. In contrast, column generation corresponds to Dantzig Wolfe decomposition as it generates columns (variables) for entry into the basis so that their inclusion improves the objective function.
Benders decomposition is a technique in mathematical programming that allows solving very large linear programming problems with block structures. This block structure is often found in applications such as stochastic programming since uncertainty is often represented by scenarios.
The strategy behind it can be summarized as divide and conquer. That is, in Benders decomposition, the variables of the original problem are divided into two (or more) sub-problems that are individually much easier to solve. If the subproblem determines that the fixed first-stage decision is infeasible, Benders cuts are generated and added to the master problem, and the problem is resolved until no cuts can be generated.

After applying row generation, the original CV route assignment formulation can be transformed to the **formulation with master problem and subproblem** below:

Master Problem (MP):

$$\max_{y_{r,v}} \Theta_{MP} = Y \tag{4.8}$$

$$\text{s.t.} \sum_{r} y_{r,v} \leq 1 \qquad \forall v \in C, \tag{4.9}$$

$$Y \leq Cuts \tag{4.10}$$

$$y_{r,v} \in \{0,1\} \qquad \forall (r,v) \in R. \tag{4.11}$$

Subproblem (SP):

$$\max_{y_{r,v}} \Theta_{SP}(\bar{z}_{r,v}) = \sum_{(r,v)} \sum_{w} \mu \alpha_{w,r,v} x_{w,r,v} - \sum_{(r,v)} c_{r,v} \bar{z}_{r,v} \tag{4.12}$$

$$\text{s.t.} \sum_{(r,v)} x_{w,r,v} \leq 1 \qquad \forall w \in W, \tag{4.13}$$

$$\sum_{w} x_{w,r,v} \leq \bar{z}_{r,v} q_c \qquad \forall (r,v) \in R, \tag{4.14}$$

$$x_{w,r,v} \geq 0 \qquad \forall (r,v) \in R. \tag{4.15}$$

The dual of the subproblem (DSP) is:

$$\max_{\gamma} \Theta_{DSP}(\bar{z}_{r,v}) = \sum_{w} \lambda_w + \sum_{(r,v)} \lambda_{r,v} \bar{z}_{r,v} q_c - \sum_{(r,v)} c_{r,v} \bar{z}_{r,v} \tag{4.16}$$

$$\text{s.t. } \lambda_w + \lambda_{r,v} \geq \alpha_{w,r,v} \qquad \forall w \in W, \forall v \in C, \tag{4.17}$$

$$\lambda \geq 0 \tag{4.18}$$

Algorithm 2 Row generation for CV route assignment

```
Initialize a feasible solution $\tilde{z}^0_{r,v}$
Initialize an upper bound and a lower bound: $U = +\infty, L = -\infty$
$t = 1$
while $U - L > \varepsilon$ do
    Obtain extreme points: $\tilde{\lambda}^t$                                  ▷ solve the DSP
    Add cut to the MP: $Z \leq \sum_w \tilde{\lambda}^t_w + \sum_{(r,v)} \tilde{z}^{t-1}_{r,v} q_c \tilde{\lambda}^t_{r,v} + \sum_{(r,v)} z_{r,v}(\tilde{\lambda}^t_{r,v} - c_{r,v})$
    Update the lower bound: $L = max[\sum_w \tilde{\lambda}^t_w + \sum_{(r,v)} \tilde{z}^{t-1}_{r,v}(q_c \tilde{\lambda}^t_{r,v} - c_{r,v}), L]$
    Obtain the solution $\tilde{z}_{r,v}, \tilde{\alpha}_{w,r,v}$                ▷ solve the MP
    Update the upper bound: $U = \Theta^*_{MP}$
end while
return $\tilde{z}_{r,v}, \tilde{\alpha}_{w,r,v}, L$
```

4.3 Matching Mechanism

4.3.1 Scope limit

We need scope limit to reduce complexity by discarding some prospective matches that will not be considered in real life. For instance, if a crowd-sourced vehicle is out of the distance range of the starting points of all orders, it will not get any assigned work packages.

The scope limit lie in multiple aspects: First of all, a crowd-sourced vehicle is not supposed to be too far from the restaurant. Then, a maximum total offset allowed from the expected time of stops is bounded, where the sum of the ready time of pick-up stops and the user's expected drop-off time should not exceed the upper offset bound. Next, the newly formed route after the insertion of an order must be related to the vehicle.

If a matching could meet all the conditions of the scope limit shown above, it could be processed by the matching policies.

4.3.2 Commitment strategy

Apart from the scope limit, a commitment strategy is required to guarantee the quality of the delivery service. The aim of the commitment strategy is to have the vehicle arrive just in time at the pickup locations. Without it, some situations we do not expect would happen like that a crowd-sourced vehicle would be matched again or the pickup location would change during the journey. To avoid such situations, the strategy can let the busy

vehicles wait for the next optimization after they become idle, which will not delay the pick-up or drop-off of any order.
If a vehicle could arrive at the location and all orders in the route are ready before the end of the next time window, the route can be kept; If a vehicle could not arrive at the location by the end of the next time window and is idle, a backup is done in case no other best vehicle is found; If any order in the route has been ready for a long time, it would execute immediately.

4.3.3 Matching policy

A myopic matching policy in Algorithm 3 that makes decisions solely based on the current state at every decision points and ignores the future events is utilized by us. This policy attempts to assign as many work packages as possible to crowd-sourced vehicles at this stage while combining the mentioned routes generation, assignment logic, scope limit and commitment strategy.

Algorithm 3 Myopic matching policy

```
Generate routes $R_t$ with parallel-insertion and then update $R_t$
Generate scope limit $\{p_{r,v}, \forall v \in V \, \forall r \in R_t\}$ for matching
Generate matches $M_t$ with the assignment model
for $m \in M_t$ do
    Check with the commitment strategy
end for
return $M_t$
```

4.3.4 DV Routes as Supplements

After the first stage, we assign as many as possible routes to all crowd-sourced vehicles, There may exist some work packages not being delivered. They are assigned to dedicated vehicles directly to complete all tasks. DV routes are generated for these work packages with the insertion algorithm. The insertion algorithm is used for both single vehicle routing and multi-vehicle routing. It first obtains an estimated cost of all work packages that will be delivered by DV. Then routes are generated to serve them. Next, a check process is performed periodically with a given time duration to see if there are some orders canceled. Cases that if the order is canceled by users need to be removed from the lists. If the time of the order exceeds the limit, a neighborhood search is performed to seek the possibility to switch it from CV to DV.

Chapter 5

Computational Study

A series of experiments are conducted in the computational framework with real-life instances. The model is first implemented and then simulated under the environment that is composed of movement on the actual street map, cancellation available and information from time to time.

5.1 Instances

10 real-life instances from Rappi are provided, each containing operations for the whole day (from 00:00 to 23:59) in a specific city.

We have used these 10 instances for experiments with a random seed value of 10. Every instance is composed of two main parts: courier data and order data. The instances are labelled from 0 to 9, and their sizes are sorted in ascending order by the number of couriers.

In courier data, the information contains the vehicle type (walking, bicycle, motorcycle and car), the location(latitude and longitude) and the available time slot (from on time to off time). Two locations (pickup and drop-off), as well as four moments (placement, preparation, ready and expected drop-off time) related to orders, are included in order data.

Instance	0	1	2	3	4	5	6	7	8	9
Courier	396	415	430	455	941	959	1156	1185	1876	1908
Order	539	844	538	735	1462	1376	2407	2959	4516	4389

Table 5.1: *Number of orders and couriers in the 10 different instances*

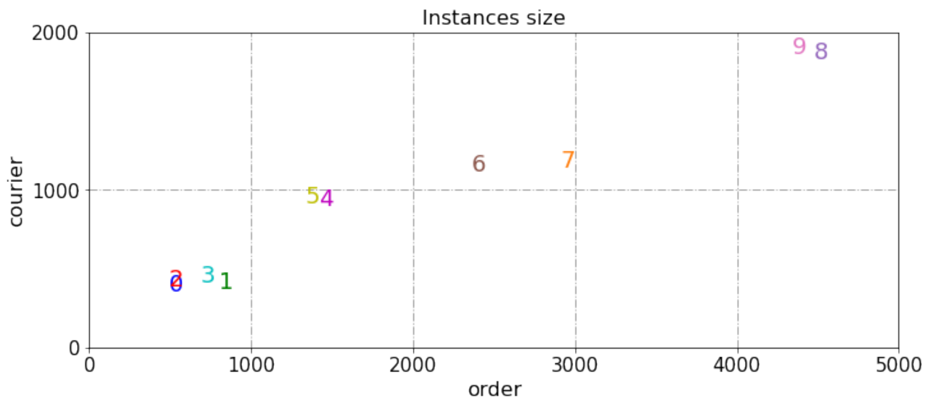
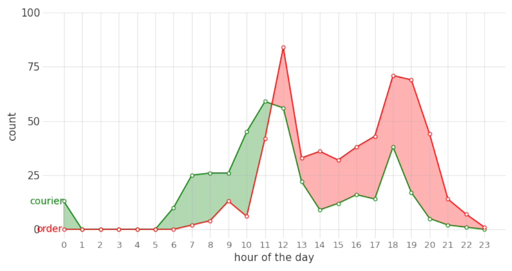


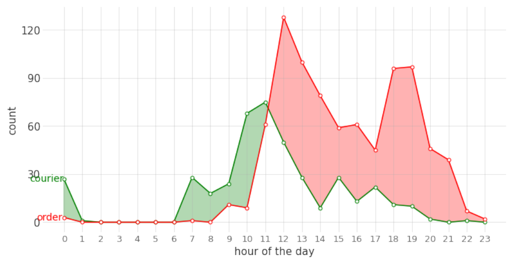
Figure 5.1: *Number of orders and couriers in the 10 different instances*

As shown in the above figure and table 5.1, taking 2000 orders and 1000 couriers as a threshold, instances 0-5 are less than this, while instances 6-9 are over.

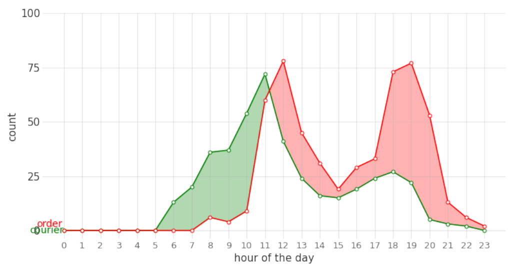
To directly see the arrivals of the orders and couriers, the distribution is shown in the figure 5.2 below by aggregating them per hour of the day. The arrival of orders in line with actual life showed two peaks, lunch and dinner. The distribution of the registered couriers also shows two peaks but is relatively steady throughout the day. More courier vehicles are registered ahead of the peak order period to increase their likelihood of being assigned to an order.



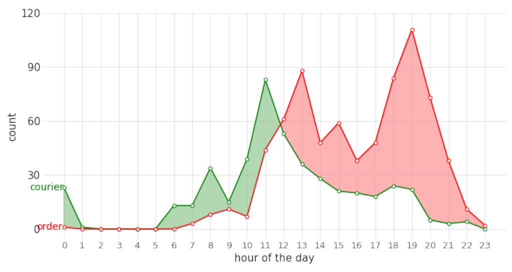
(a) *Instance 0*



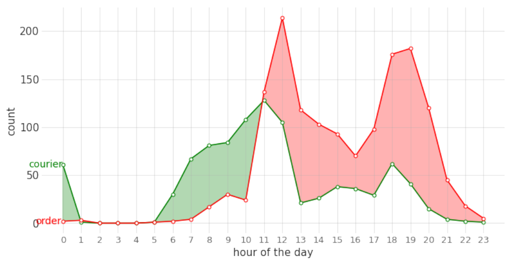
(b) *Instance 1*



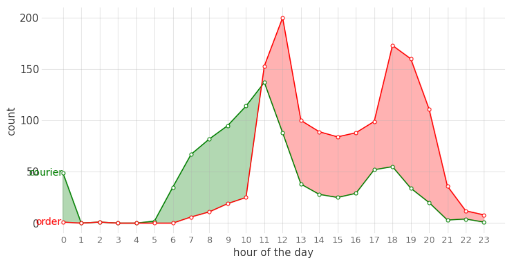
(c) *Instance 2*



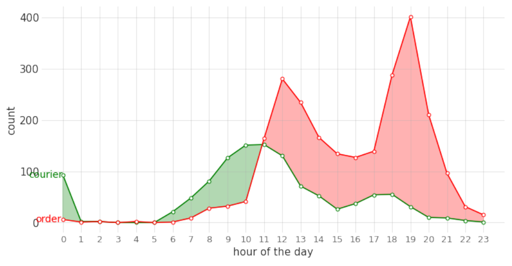
(d) *Instance 3*



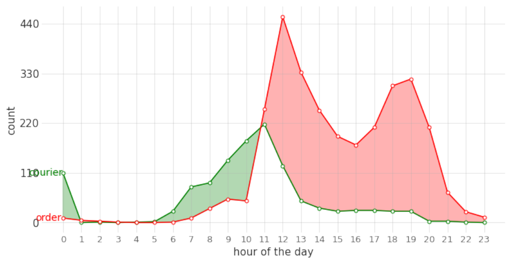
(e) *Instance 4*



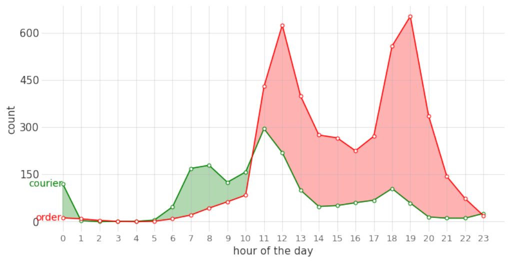
(f) *Instance 5*



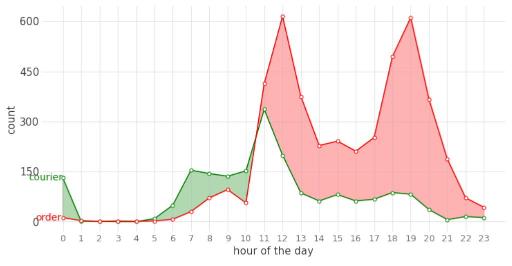
(g) *Instance 6*



(h) *Instance 7*



(i) *Instance 8*



(j) *Instance 9*

Figure 5.2: *Arrivals of couriers and orders in the 10 different instances*

5.2 Implementation

We implement a mixed-integer programming (MIP) assignment model as well as the myopic matching policy, and integrate them into an open-source available computational framework[38]. The row generation model (Algorithm 2) is also implemented.

With the help of Gurobi, the variables of routes and couriers are built and then followed by a built objective function (Equation 3.1). We limit that one order is assigned to only one courier within the matching scope limit, which is done by the implementation of the two types of constraints: one for the route, the others for the courier (Equation 3.2/3/11). The myopic matching policy (Algorithm 3) is executed at the beginning of every time window, which involves the functions of generation and grouping routes, bundles of orders calculation, and costs for solutions procession.

The computational framework[38] has plenty of useful built-in functions and services. We utilized the OSRM (Open Source Routing Machine)[39] service to simulate the model with the given instances, which means the movement of the vehicle is completely along the real-life road map, and the traffic rules and congestion levels are consistent with real life, which will change with time and space. An example of a bicycle route on OSRM is below.

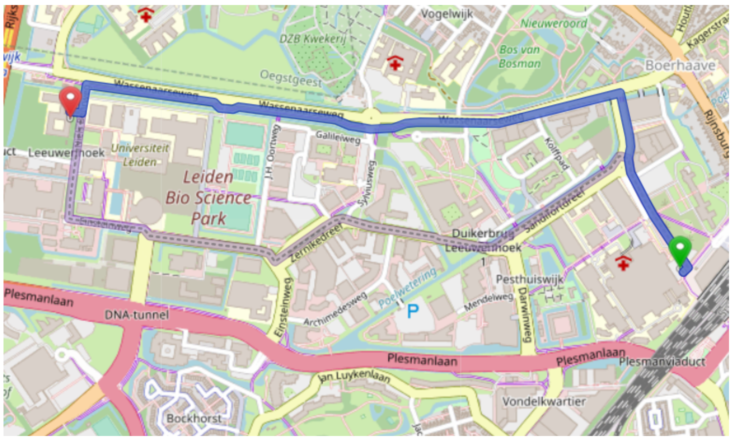


Figure 5.3: Example of the movement on OSRM service

It is necessary to have some functions to first identify the vehicle type and then locate the vehicles from time to time during the experiments. Rules that create disturbances for the dispatch system, such as order cancellation policies, are also implemented.

5.3 Experiment

Experiments are carried out in all different instances under the same settings of the lunchtime simulation scenario. The complete settings can be found in Appendix B, and the information on the hardware and software can be found in Appendix C.

In the beginning, all the instances are loaded into the embedded database. Next, we test and simulate the model to assign and match orders to the routes of crowd-sourced vehicles. During the experiments, two containers in Docker are required: one container where the movement, pickup and drop-off actions of vehicles run; the other one embedded with the database simulates from the start time to the end time with time windows and performs the assignment algorithm and optimizations. Finally, when all simulations are done, the query tool is used to search and export results by SQL sentences for the subsequent procedures.

We simulate the implemented model and then compare it with the network flow[38] MDRP[1] model. Results are analyzed by performance metrics and computational efficiency in the following section.

5.4 Result

5.4.1 Performance Metrics

Meal delivery routing problems involve several stakeholders (the dispatcher, restaurants, users, and couriers), each with its own objectives and worries. Thus, several performance metrics are required to measure the quality of the model from different aspects.

Time Criteria

All the time performances over instances 0-9 are shown in the box plots (Figure 5.4). Here are the descriptions of the six time criteria and analyses of sub-figures under each criterion:

(a) **Click to door (CtD) time**: the difference between the drop-off time of an order and its placement time;
Figure (a) shows the user needs to wait for about 40 minutes to get the order after placing it.

(b) **In store to pick-up (StP) time**: the interval between the time of pick-up and the time the courier arrives at the store, which measures the quality of the courier's assignments and the precision of the system at arrival times;
The median time interval between the arrival of the courier and the pickup is 11 minutes shown in figure (b).

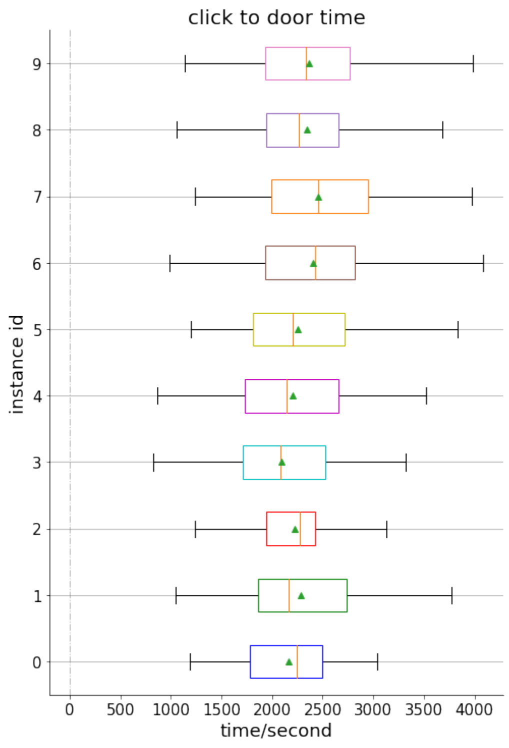
(c) **Ready to pick (RtP) time**: the difference between the pickup time of an order and its ready time;
In figure (c), a meal box is supposed to wait 6 to 7 minutes after it is ready, and then to be picked up by a courier.

(d) **Ready to door (RtD) time**: the difference between the drop-off time of an order and its ready time;

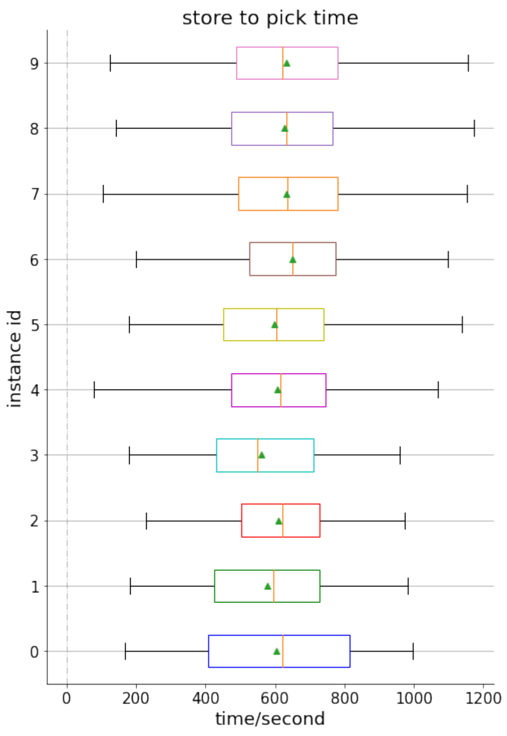
Figure (d) indicates the meal box can be delivered to the user's location approximately 1000 seconds after it is ready.

(e) **Click to taken (CtT) time**: the time difference between when a courier accepts an order and the time it was placed, which reflects the effects of pre-position and acceptance;
As for the click to taken time in figure (e), things are a little different: 750 seconds is enough for a courier to accept a placed order of instances 2 and 3; For other instances, it requires 1000 seconds.

(f) **Lateness**: the degree to which an order arrives later than expected can be used to gauge how well the user experience and the solutions are being provided;
From figure (f), we could see that almost over three quarters of orders are delayed than expectation. Half of the orders are no later than 500 seconds as expected.

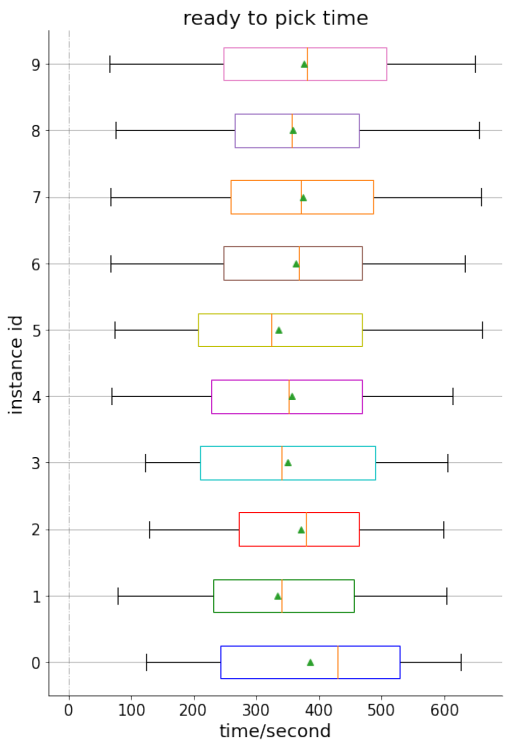


(a) *Click to door (CtD) time*

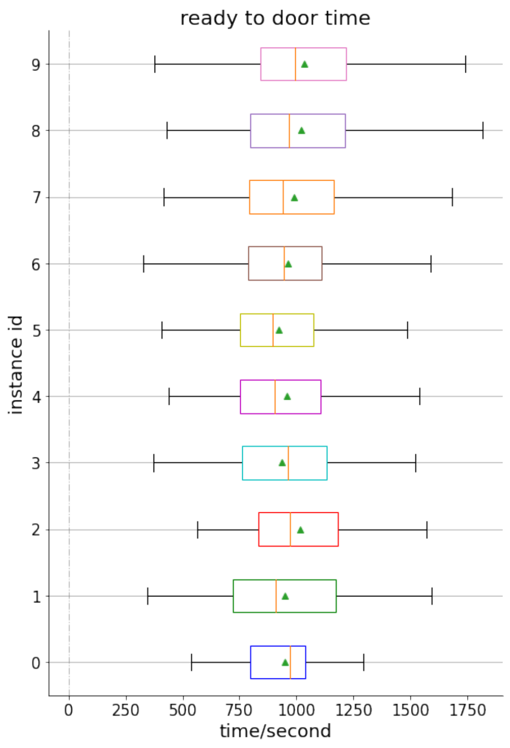


(b) *In store to pick-up (StP) time*

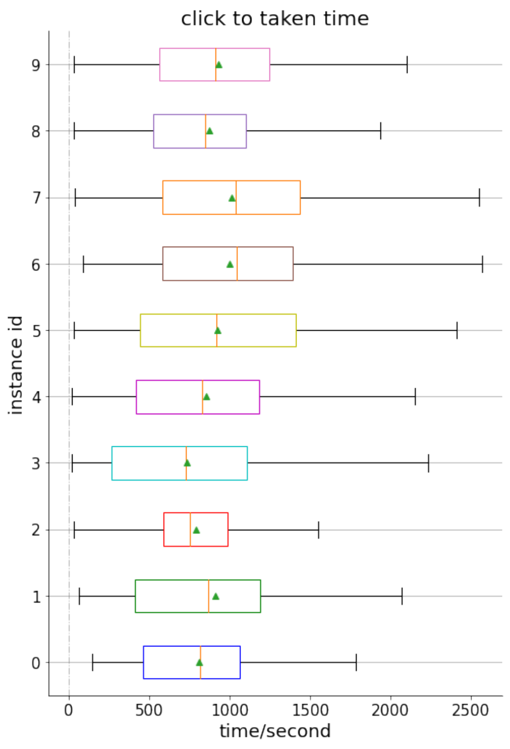
Figure 5.4: *Performance metrics box plots*



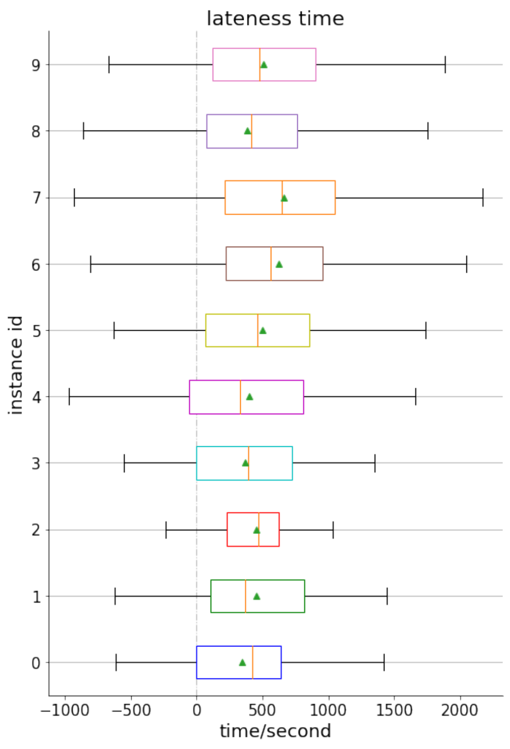
(c) Ready to pick (RtP) time



(d) Ready to door (RtD) time



(e) Click to taken (CtT) time



(f) Lateness

Fill Rate

Fill rate, also known as order fulfillment rate, is the percentage of orders that can be shipped from available stock without any lost sales or out of stock. Fill rate can be calculated by counting the number of orders that were fulfilled and dividing it by the total number of placed orders:

$$Fill\ Rate = \frac{Total\ Orders\ Fulfilled}{Total\ Orders\ Placed} * 100\% \tag{5.1}$$

For each placed order, it is either fulfilled or cancelled. Order cancellations can come from the dispatcher or the users. Couriers accept the order stochastically with a probability, which obeys a uniform distribution between the minimum and the maximum probability. Orders without any possible assignments to an available courier will be cancelled after a certain period. When the set maximum waiting time is exceeded, both the user and the system can cancel the order randomly with a set probability. The settings of parameters can be found in Appendix B.

The fill rates of the implemented MIP model for all instances are shown in Table 5.2.

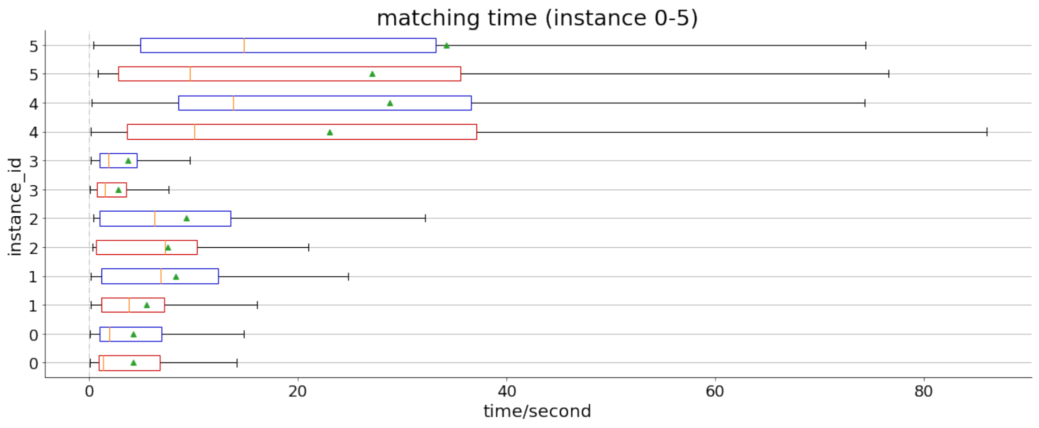
	Model		
Instance	MIP	MIP (with scope)	Network flow[38]
0	55.73%	**85.48%**	81.96%
1	65.82%	**86.07%**	55.69%
2	**66.23%**	53.84%	66.23%
3	**72.72%**	65.15%	69.69%
4	**73.50%**	59.50%	68.50%
5	68.50%	61.19%	**68.81%**
6	**89.51%**	58.63%	68.01%
7	**80.38%**	50.67%	51.63%
8	**68.62%**	50.67%	51.63%
9	55.35%	**55.61%**	46.64%
Average	**69.43%**	62.95%	62.51%

Table 5.2: *Fill rates of the models*

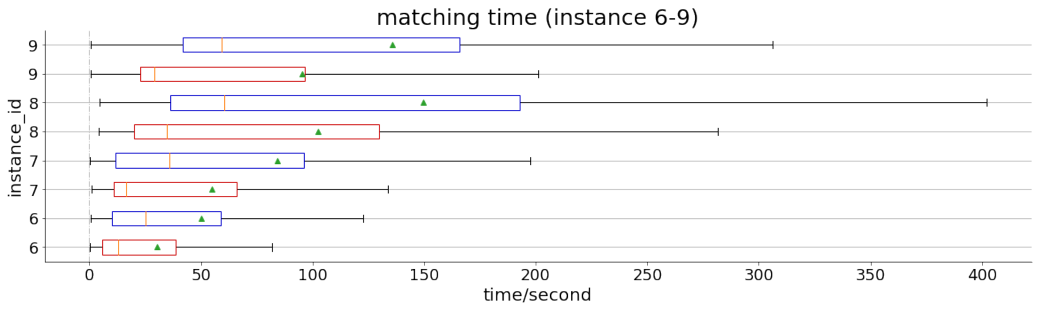
The mixed integer model has fill rates greater than 50% on all instances, with an average of 69.43%, which outperforms the network flow model in the simulations. On top of the MIP model, a limitation of 4km on the matching scope was applied, which still performed better than the network flow model on fill rate.

5.4.2 Computational Efficiency

Due to the simulation being carried out on the OSRM, the movement and positioning of the vehicle take time. The time of matching(Figure 5.5) and routing(Figure 5.6) is visualized in the box plots.



***(a)** Instance 0-5 (orange: MIP; blue: Network)*

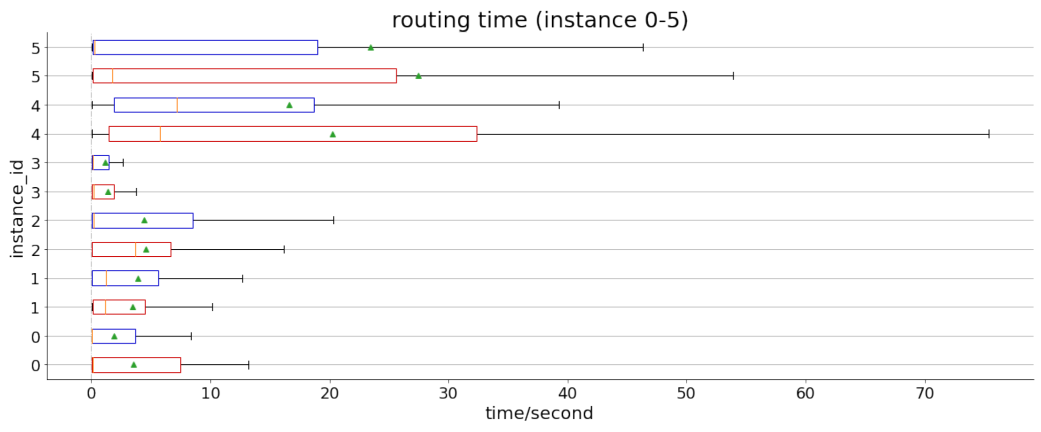


***(b)** Instance 6-9 (orange: MIP; blue: Network)*

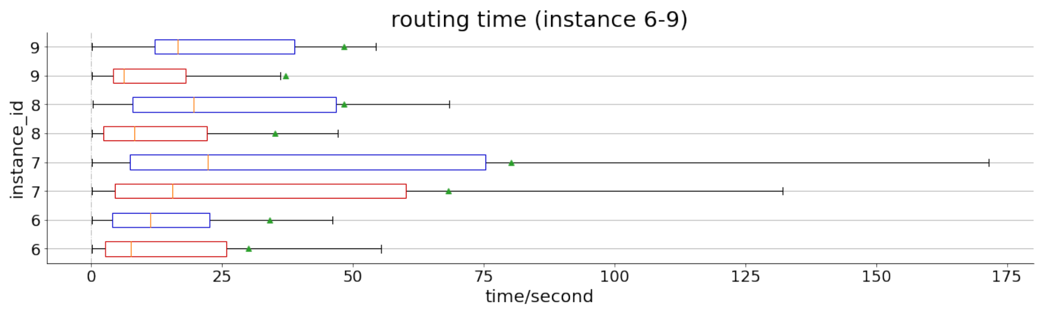
***Figure 5.5:** Matching time box plots*

For all instances, the medians, means and the first quartiles of the computation time of the MIP model during the matching process are smaller than those of the network flow model. Except for instances 4 and 5, so do the third quartiles.

The usage of the matching scope limit significantly reduces computational complexity in matching process, because we do not consider assigning orders to the vehicle out of the set range.



(a) *Instance 0-5 (orange: MIP; blue: Network)*



(b) *Instance 6-9 (orange: MIP; blue: Network)*

Figure 5.6: *Routing time box plots*

In general, the MIP model performs better in routing time. In instances 6-9, the medians, means and the first quartiles are all 10 to 20 seconds smaller. Despite both models requiring similar time for routing on smaller datasets like instances 0-5, sometimes MIP takes 1 to 2 seconds more.

We could conclude that MIP performs better in computational efficiency. The larger instances require longer computational time in both operations of matching and routing. The sum of matching and routing time is smaller than the length of the time window of 4 minutes, which means the operations of the dispatcher can be done within one time window.

Instance	MIP	Network	Improvement
0	7.735	6.151	-0.25
1	8.931	12.193	0.26
2	12.066	13.766	0.12
3	4.167	4.869	0.14
4	43.266	45.443	0.04
5	54.604	57.642	0.05
6	60.208	84.339	0.28
7	123.209	164.498	0.25
8	137.655	197.877	0.30
9	132.547	183.813	0.27
Average Improvement			14%

Table 5.3: *Means of computational time for all instances in seconds*

Chapter 6

Discussion

This study presents several mathematical formulations for Meal Delivery Routing Problem and algorithms to solve the problem. The research starts from the Vehicle Routing Problem to the concepts of MDRP and formulates it by the procedure of decomposition. This decomposition formulation encourages the usage of crowd-sourced vehicles and leaves small space for dedicated vehicles to handle some special cases to ensure all the orders could be assigned. The row-generating model succeeds because it involves more information in the variable definitions, avoiding the complex constraints of modeling the relationships between them[26]. We believe it could balance embedding the information into definitions and rendering the formulation efficiently in computation.

Since it is hard to integrate the row generation model into the computational framework, we carry out the simulations by implementing the MIP assignment model with the myopic matching policy; and compare the results with the network flow MDRP model[38].

Finding a set of routes that optimize performance metrics by matching vehicles with orders could be seen as the solution to the problem. From the results under time criteria(Figure 5.4), we could see that the experiments of MIP model over all instances have good performances on lateness: The medians are around 500 seconds and some orders could be sent to their destination earlier than expectations. If we increase the quality of service requirements and reduce the system's tolerance for delay, the performance of the model results will decrease under various indicators.

In all, The research includes MDRP formulations with crowd-sourced vehicles, which focuses on assigning the orders to crowd-sourced couriers using a row-generation model and a MIP model with time windows. Both models can make use of the information at the current time point to match the couriers' routes with the orders. Compared with the MDRP network flow model, the MIP model reduces the average computation time by 14% on all instances, while achieving a 7% higher average fill rate.

Future work could be to implement and integrate the more complex row generation model of crowd-sourced vehicles into the framework for the simulation. It is also a good way to create a new framework which could contain different stages and types of vehicles, allowing fast simulations with a larger scale of data. More complex objective functions which not only consider cost but carbon emission could be modelled. There seems to be great potential in self-adapted time windows and row-column generations in extended formulations. Other stochastic factors in crowd-sourced delivery that people are concerned about could be taken into more consideration. Besides the aspects of time and cost, some optimizations on the topological space of the algorithm can also be involved in future research. Not only do we need progress in abstracting and describing problems, but we also have a long way to go in weighing the quality and cost of solutions.

Bibliography

[1] D. Reyes, A. L. Erera, M. W. P. Savelsbergh, S. Sahasrabudhe, and R. J. O'Neil, *The Meal Delivery Routing Problem*, 2018.

[2] G. B. Dantzig and J. H. Ramser, *The Truck Dispatching Problem*, Management Science **6**, 80 (1959).

[3] S. Irnich, P. Toth, and D. Vigo, *Chapter 1: The Family of Vehicle Routing Problems*, pages 1–33, 2014.

[4] P. Toth and D. Vigo, *1. An Overview of Vehicle Routing Problems*, pages 1–26.

[5] B. Golden, S. Raghavan, and E. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43, 2008.

[6] B. Eksioglu, A. Vural, and A. Reisman, *The vehicle routing problem: A taxonomic review*, Computers & Industrial Engineering **57**, 1472 (2009).

[7] C. Lin, K. L. Choy, G. T. S. Ho, S. H. Chung, and H. Y. Lam, *Survey of Green Vehicle Routing Problem: Past and Future Trends*, Expert Syst. Appl. **41**, 1118–1138 (2014).

[8] M. Marinaki and Y. Marinakis, *A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands*, Expert Systems with Applications **46**, 145 (2016).

[9] B. Golden, A. Assad, L. Levy, and F. Gheysens, *The fleet size and mix vehicle routing problem*, Computers & Operations Research **11**, 49 (1984).

[10] A. W. Siddiqui and M. Verma, *A bi-objective approach to routing and scheduling maritime transportation of crude oil*, Transportation Research Part D: Transport and Environment **37**, 65 (2015).

[11] M. Talebian Sharif and M. Salari, *A GRASP algorithm for a humanitarian relief transportation problem,* Engineering Applications of Artificial Intelligence **41**, 259 (2015).

[12] D. S. Lai, O. Caliskan Demirag, and J. M. Leung, *A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph,* Transportation Research Part E: Logistics and Transportation Review **86**, 32 (2016).

[13] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, *A review of dynamic vehicle routing problems,* European Journal of Operational Research **225**, 1 (2013).

[14] H. N. Psaraftis, M. Wen, and C. A. Kontovas, *Dynamic vehicle routing problems: Three decades and counting,* Networks **67**, 3 (2016).

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

[16] S. Mitrović-Minić and G. Laporte, *Waiting strategies for the dynamic pickup and delivery problem with time windows,* Transportation Research Part B: Methodological **38**, 635 (2004).

[17] K. Lund, O. Madsen, and R. J.M., *Vehicle Routing Problems with Varying Degrees of Dynamism,* (1996).

[18] A. Larsen, O. Madsen, and M. Solomon, *Partially dynamic vehicle routing—models and algorithms*, Journal of the Operational Research Society **53**, 637 (2002).

[19] R. R. 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 **253**, 614 (2016).

[20] C. Yan, H. Zhu, N. Korolko, and D. Woodard, *Dynamic pricing and matching in ride-hailing platforms,* Naval Research Logistics (NRL) **67**, 705 (2020).

[21] S. Voccia, A. Campbell, and B. Thomas, *The Same-Day Delivery Problem for Online Purchases*, 2015.

[22] M. A. Klapp, A. L. Erera, and A. Toriello, *The One-Dimensional Dynamic Dispatch Waves Problem,* Transportation Science **52**, 402 (2018).

[23] S. Voccia, A. Campbell, and B. Thomas, *The Same-Day Delivery Problem for Online Purchases,* Transportation Science **53** (2017).

[24] M. A. Klapp, A. L. Erera, and A. Toriello, *The Dynamic Dispatch Waves Problem for same-day delivery*, European Journal of Operational Research **271**, 519 (2018).

[25] M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak, *The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times*, Transportation Science **55**, 75 (2021).

[26] B. Yildiz and M. Savelsbergh, *Provably High-Quality Solutions for the Meal Delivery Routing Problem,* Transportation Science **53**, 1372 (2019).

[27] Z. Steever, M. Karwan, and C. Murray, *Dynamic courier routing for a food delivery service,* Computers & Operations Research **107**, 173 (2019).

[28] F. D. Hildebrandt and M. W. Ulmer, *Supervised Learning for Arrival Time Estimations in Restaurant Meal Delivery,* Transportation Science **56**, 1058 (2022).

[29] C. Archetti, M. Savelsbergh, and M. Speranza, *The Vehicle Routing Problem with Occasional Drivers,* European Journal of Operational Research **254** (2016).

[30] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and D. Laganà, *The Vehicle Routing Problem with Occasional Drivers and Time Windows*, pages 577–587, 2017.

[31] L. Dahle, H. Andersson, M. Christiansen, and M. G. Speranza, *The pickup and delivery problem with time windows and occasional drivers*, Computers & Operations Research **109**, 122 (2019).

[32] W. Liao, L. Zhang, and Z. Wei, *Multi-objective green meal delivery routing problem based on a two-stage solution strategy,* Journal of Cleaner Production **258**, 120627 (2020).

[33] K. Gdowska, A. Viana, and J. P. Pedroso, *Stochastic last-mile delivery with crowdshipping,* Transportation Research Procedia **30**, 90 (2018), EURO Mini Conference on "Advances in Freight Transportation and Logistics".

[34] A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk, *Crowdsourced Delivery-A Dynamic Pickup and Delivery Problem with Ad Hoc Drivers*, Transportation Science **53**, 222 (2019).

[35] I. Dayarian and M. Savelsbergh, *Crowdshipping and Same-day Delivery: Employing In-store Customers to Deliver Online Orders*, Production and Operations Management **29**, 2153 (2020).

[36] A. Sampaio, M. Savelsbergh, L. P. Veelenturf, and T. Van Woensel, *Delivery systems with crowd-sourced drivers: A pickup and delivery problem with transfers*, Networks **76**, 232 (2020).

[37] F. Torres, M. Gendreau, and W. Rei, *Vehicle Routing with Stochastic Supply of Crowd Vehicles and Time Windows*, Transportation Science **56**, 631 (2022).

[38] S. Q. Rojas, *Computational Framework for Solving the Meal Delivery Routing Problem*, (2020).

[39] D. Luxen and C. Vetter, *Real-time routing with OpenStreetMap data*, in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 513–516, New York, NY, USA, 2011, ACM.

Appendix A

Notations

Table A.1: Notation for *Meal Delivery Routing with Crowd-sourced Vehicles*

Sets:	
O	Set of orders, $o \in O$
V	Set of all vehicles, $V = C \cup D$, $v \in V$
C	Set of crowd-sourced vehicles (CV)
D	Set of dedicated vehicles (DV)
N^v	Set of locations that vehicles could reach, $N^v = N_w \cup \{\ell_v, \overline{\ell}_v\}$
N_w	Set of locations of all work-packages, $N_w = \{\ell_o : o \in O\}$
T_o	Set of continuous time intervals during which it is possible to start a work package
W	Set of work-packages: $W(v)$ can be performed by a vehicle v; $W(o)$ contains order o
R	Set of routes, $(r, v) \in R$
Variables:	
x_w^v	Binary variable equals to 1 if the work-package w is performed by the vehicle v
z_v, y_v	Binary variable equals to 1 if CV or DV v is used
τ_o	Drop-off time for an order o

Continued on next page

Table A.1: Notation for *Meal Delivery Routing with Crowd-sourced Vehicles* (Continued)

$y^c_{r,v}, y^d_{r,v}$	Binary variable equals to 1 if the r^{th} feasible route of CV or DV is used
$\alpha^c_{r,w,v}, \alpha^d_{r,w,v}$	Binary variable equals to 1 if it is feasible for route r of CV or DV v to deliver the work-package w
Parameters:	
d_w	Distance of the work-package from start to end
s_w, f_w	Start location, end location related to the work-package
σ_w, ϕ_w	Start time, end time related to the work-package
δ^o_w	Drop-off time for the order in the work-package
ℓ_o	Drop-off location
$\ell_v, \overline{\ell}_v$	On location, off location of the vehicle
h	Location of the depot for DV
u^o	Service time associated with the delivery of an order at a customer location
p_1	Price per unit of CV based on travel distance
p_2	Compensation per work-package of CV above the basic compensation threshold
p_3	Monetized cost per unit of DV based on travel distance
F_d	Fixed cost of each DV
b	Basic compensation threshold for one CV
q_c	Capacity of CV that is the maximum number of work-packages all CV capable or willing to deliver
q_d	Capacity of DV that is the maximum number of work-packages all DV capable to deliver
ϱ^{max}	Upper bound for click-to-door time

Appendix B

Simulation Settings

SEED	10
OPTIMIZER	gurobi
SIMULATE	08:00-13:00
USERS	08:00-12:00
COURIERS	08:00-11:00
WARM UP TIME	9,000s
DISPATCHER WAIT TO CANCEL	45min
ROLLING HORIZON TIME	4min
DISPATCHER PREPOSITIONING TIME	1h
DISPATCHER PROSPECTS MAX DISTANCE	4km
DISPATCHER PROSPECTS MAX ORDERS	3
DISPATCHER PROSPECTS MAX STOP OFFSET	10min
DISPATCHER PROSPECTS MAX READY TIME	4min
DISPATCHER READY TIME SLACK	10min
DISPATCHER DELAY PENALTY	0.4
COURIER ACCEPTANCE RATE	$U(0.4, 1)$
COURIER WAIT TO ACCEPT	15s
COURIER MOVEMENT PROBABILITY	0.4
COURIER WAIT TO MOVE	30min
COURIER EARNINGS PER ORDER	4
USER WAIT TO CANCEL	30min
USER CANCELLATION PROBABILITY	0.6
USER SERVICE TIME	$U(2, 5)$min
RESTAURANT SERVICE TIME	$U(2, 10)$min
ORDER TARGET DROP OFF TIME	40min

Appendix C

Computer and Software Information

Computer	
Intel(R) Core(TM) i5-10200H CPU @ 2.40GHz	
RAM	16.0 GB
OS	Windows 10 x64
Software & Packages	
Python	3.7.0
gurobipy	10.1.0
Docker	4.11.1
colombia osrm	
postgres	
numpy	1.19.1
pandas	1.1.1
haversine	2.3.0
python-geohash	0.8.5
snowflake	0.0.3
alembic	1.4.3
simpy	4.0.1