ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis Econometrie en Operationele Research

The meal Delivery Routing Problem with independent drivers

M.A. Blom, 506008

Supervisor: B.T.C. van Rossum

Second assessor: R.M. Badenbroek

Erasmus
University
Rotterdam

July 3, 2022

Contents

Abstract

Online meal delivery platforms have become more and more prominent the last few years Hirschberg et al. (2020). They try to deliver orders fast and for the lowest possible cost. This study examines the problem of assigning couriers to orders in the most optimal fashion using a set of predetermined performance measures. This is vital for delivery platforms' profitability and to stay competitive with others. In order to tackle this problem data is used from the paper of Reyes et al. (2018). This study will first replicate this study and then continues by elaborating on its Meal Delivery Routing Problem (MDRP) by adding freelance drivers, these drivers can accept or reject to deliver orders. This study found the same concluding results from Reyes et al. (2018) and when adding freelance drivers costs decrease as more independent drivers came into the system while delivery times and pickup delays increase.

1 Introduction

The last few years online food delivery platforms have grown immensely (Hirschberg et al., 2020). It has become more and more apparent to big businesses that food delivery is the future. Even big supermarket chains are now joining the online food delivery services in rapid pace (*Ook traditionele supermarkten storten zich op flitsbezorging*, 2022). As the field of third-party delivery is exponentially becoming more and more crowded many companies are investing big amounts of money into the industry (Dewey, 2021). This calls for fast and cost-effective methods to deliver food to customers.

The meal delivery routing problem (MDRP) proposed by Reyes et al. (2018) consists of trying to match couriers and orders in a time horizon (a day) while optimising pre-determined performance measures such as delivery times and costs. This MDRP is part of the family of short term pickup and delivery problems and has a highly dynamic and myopic nature.

Developing optimisation methods for the short term pickup and delivery problem is a though challenge. This system is highly dynamic, orders can come in any minute and at any rate, and should be delivered within a short time frame (within an hour). This all while delays can occur at any point in the delivery process. These delays can occur while preparing the order, picking the order up and finally delivering the food. The meal delivery platform should also cope well with sudden and abrupt changes in demand. Besides sudden changes in demand delivery platforms face the problem of geographic inequality, some area's have more restaurants and some area's have more people. This is a problem typical to this industry and should be handled accordingly.

This challenge becomes even harder when realizing that many delivery platforms have opted to use independent contractors to pickup and deliver orders (Reuters, 2016). Using contractors brings a few difficulties, because contractors have more freedom than employees. These contractors can for instance chose to decline an order or to not follow a certain delivery route (given by the optimiser) which would lead to more variability and complexity. Independent contractors can also chose their working hours which results in difficulties around scheduling.

To solve the aforementioned problem this study imposes the research question: “How to operate a meal-ordering platform so that last-mile meal orders are not only delivered within a given time frame but also delivered profitably?” In order to turn a profit with delivery networks, adequate optimisation methods should be developed and scrutinized. Without applying such methods for this myopic pickup and delivery problem online delivery platforms cannot sustain themselves in the long term. Costs and delivery times would be too high, turning profits into losses.
This paper will replicate part of the work of Reyes et al. (2018), by using their proposed rolling horizon matching based algorithm. Then we continue to expand on this algorithm by adding freelance drivers who can make autonomous decisions. They can chose if they want to deliver an order, which makes for a more interesting and real world MDRP. We find that our replication results are in line with the original paper. Also when using freelance drivers we find that costs go down and delivery times go up when freelancers are more present in the system.
This paper’s most important contributions are first of all doing the groundwork around the Meal Delivery Routing Problem, more specifically reproducing the work of Reyes et al. (2018), by giving a definition and a solution of an instance of the MDRP then we proceed by reproducing algorithms and heuristics Reyes et al. (2018) uses. Secondly this study will go over tools to schedule drivers (employees or freelancers) while optimising some predetermined performance measures. Thirdly as an extension to the MDRP this paper explores courier autonomy and how this effects the solution of the general MDRP. By for instance examining what ratio of employees to freelancers maximise profit or minimise delivery times?
This paper continues first by giving a literature review (section 2) into vehicle routing problems which are useful to the MDRP and previous research into the use of freelance drivers. Then we continue with section 3, the problem description. After laying out the problem we examine methods that can be used to solve the MDRP and tackle problems with regards to courier autonomy (section 4). After going over the methodology we present the results of our rolling horizon algorithm presented in the methodology, section 5. The paper ends with section 6 giving concluding remarks and possibilities for further research.

2 Literature review

The MDRP is related to the large collection of Vehicle Routing Problems (VRP’s), more specific the dynamic pickup and delivery problems (dPDP). The VRP’s describe the problem of finding an optimal collection of routes traveled by a fleet of vehicles in order to serve a set of customers, for instance package delivery. While dPDP’s try to solve the same problem but now with the addition of busy and idly periods in the time horizon, dPDP’s also focus specifically on package delivery while VRP’s are more general. These problems have been researched for a long period, where Dantzig & Ramser (1959) were the first to introduce VRP in 1959. The Vehicle Routing Problem is still a hot topic with more recent research like Pillac et al. (2013) and Berbeglia et al. (2010) exploring dPDP and going over basic solutions to these types of problems.
Recent research into dPDP have zeroed in on the moving of people, like dial-a-ride systems (Cordeau

& Laporte, 2007) these are online taxi platforms like Uber, more recent studies by (Agatz et al., 2012). Dial-a-ride systems are of course not the same as our MDRP but they definitely share some valuable characteristics. Both systems are for instance highly dynamic and should be optimised in near real-time, they also operate with delivery urgency (orders should be delivered within given time frame) and driver autonomy which is also interesting for MDRP.

The challenge of delivering food on time (usually in an hour) is often tackled by heuristics where Gendreau et al. (2001) proposed a tabu search heuristic and Coslovich et al. (2006) going over dial-a-ride heuristics with time windows. More recent studies tried using a myopic rolling horizon repeated matching approach Wang et al. (2018) and have produced high quality solutions for the dial-a-ride problem, which is interesting to use in our MDRP case.

Recently a new type of dPDPs have been researched, the dynamic delivery problems (Berbeglia et al. (2010), Agatz et al. (2012), Archetti et al. (2016)). Where Archetti et al. (2016) also uses occasional drivers which could be interesting when researching problems regarding driver autonomy. In these dynamic delivery problems vehicles (couriers in our MDRP case) are delivering multiple orders in the time horizon from a depot (restaurants in MDRP). Solutions to these dynamic delivery problems intuitively have a special structure. Once a courier picks up their order, changing their route is impractical and usually not effective. One could force this structure in the model by implementing an additional constraint, couriers can only handle a pickup if their vehicle is empty. Research by Ulmer et al. (2019) assessed the implications and impact of relaxing this constraint. This study suggests that preemptive returns could increase the number of deliveries per workday, it also showed that preemptive returns are only effective in very specific instances where the changes in route are meaningful enough.

For extensions on the Meal Delivery Routing Problem this research goes over courier autonomy. What are the effects of canceling orders, autonomous moving and self scheduling of the courier on scheduling and profits? Archetti et al. (2016) have researched a VRP with occasional drivers, or more specifically using a fleet of drivers with employees and freelancers. In their research they use a distance measure to decide if an occasional driver is going to accept an order. When an occasional driver delivers that order they get a compensation, first they go over fixed compensation and later over variable compensation. The research of Archetti et al. (2016) also covers using multiple starting points for drivers, in stead of just one single point. The use of multiple starting points could improve profitability.

This paper contributes to following research by diving deeper into the independent driver problem. What percentage should an online meal delivery platform use to minimise cost but keep delivery times low enough so that customers keep coming back? This paper will also validate the results of Reyes et al. (2018) to see if their default rolling horizon matching algorithm is in fact the best at minimising delivery times and costs.

3 Problem Description

The meal delivery routing problem consists of matching couriers to orders over an operating period, while trying to optimise some predetermined performance measure(s). This study imposes the central research question: "How do we operate a meal-ordering platform so that last-mile meal orders are not only delivered within a given time frame but also delivered profitably?" Let us first introduce the meal delivery routing problem (MDRP), and then find out what the main features of this problem are. Then we dive into assumptions made in order to tackle this problem. Lastly going over autonomous couriers and the assumptions made to incorporate them in the system.

3.1 Notation

To start the definition of the MDRP let us introduce some notations also used in Reyes et al. (2018). Let R be the set of all restaurants, where $r \in R$ has location l_r it furthermore has a pickup time s_r which is the time the courier takes to pickup the order and be back on his/her bike. Note that their is also a service time s_r when delivering the order. Let O be the set of all orders, where $o \in O$ has a corresponding restaurant $r_o \in R$ placement time a_o, ready time e_o and drop-off location l_o. Let C be the set of all couriers where $c \in C$ has an on-time e_m, on-location l_c and an off-time $n_c > e_m$. Information about the restaurants and couriers is known before the start of an operation period, information about orders however is only known at the placement time a_o.

3.2 Assumptions

A courier is assumed to be available at time t when $e_d \leq t + \Delta_2$, where e_d is the time when courier d becomes available for a new assignment.
An order is assumed to be ready at time t when $e_o \leq t + \Delta_1$, where e_o is the ready time of order o.
It is assumed that multiple orders from the same restaurant can be combined into bundles with multiple drop-off locations, there is no maximum amount of orders that can fit into a bundle. Ready time of this bundle is the latest of all orders, furthermore the delivery time of this bundle is that of the most optimal found route using euclidean distance. If restaurant r_s is at (x_1, y_1) and order o is at (x_2, y_2) then the travel time would be $\lceil \gamma \times \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \rceil$, where $\frac{1}{\gamma}$ would be the travel speed of a vehicle.
During the operating period travel times between restaurant and delivery location are assumed to be constant over time. Service times either resulting from pick up or delivery are also constant over the operating period.
Couriers can deliver orders after their off-time but cannot pick up orders after their off-time. Furthermore couriers are compensated either at a constant rate p_1 per hour or by compensation fee p_2 earned per delivered order, decided by what results in the highest pay: $\max\{p_1 \times (n_c - e_c), p_2 \times n\}$.

3.3 Autonomous couriers

The aforementioned MDRP is expanded by also using independent couriers alongside the employees $c \in C$ used in the original problem. This set is now substituted by independent/occasional drivers $i \in I$, and the full set of couriers now is $A = I \cup C_{after}$. In this extended problem we use the default settings of the rolling horizon algorithm explained in section 4 Methodology. Note that at any optimisation time t, the size of the fleet does not change compared to the original problem.
It is assumed that independent driver i is willing to make a delivery if $(c_i \times w_s) \times \Omega \leq p_2$. The research of Archetti et al. (2016) also uses a distance measure the decide if an autonomous courier will accept an order. In this equation the parameter Ω is a constant larger than one, ensuring that the independent courier not only breaks even but has a decent hourly salary, Ω can be seen as the profit margin a driver is using. The parameter w_s represents the total minutes to drive to deliver bundle s, calculated using euclidean distance. When driver i is willing to make a delivery it is assumed the driver will always do so when assigned to deliver bundle s. It is furthermore assumed that these drivers follow the route provided by the algorithm.
The ratio of independent drivers to total fleet is assumed to be evenly distributed during the time horizon, at every optimisation time x percent of the total available couriers is independent. This is necessary when this paper examines what would be the optimal ratio of employee to freelancer according to some performance measures.
When multiple independent couriers accept an order a auction is held. Every couriers is assumed to bid at its lowest accepting compensation, $\frac{p_2}{\Omega}$. The order will get assigned to the lowest bidder.

3.4 Performance measures

In order to properly optimise the MRDP we need to define some performance measures. Below the performance measures are enumerated and explained.

1. Percentage of orders delivered: Percentage of total orders that came into the system and orders that were delivered to the consumer.

2. Click-to-door (CtoD): Time between placement time of an order a_o and drop-off time.

3. Ready-to-pickup (RtoP): Time between pickup time e_o and ready time of an order.

4. Orders per bundle: Amount of orders in one bundle for each assignment

5. Cost per order: Total courier payment divided by number orders delivered.

4 Methodology

Now that we have defined the MDRP we can elaborate on methods this study uses to tackle this problem. We start with examining the rolling horizon matching based algorithm. This introductory algorithm uses a simple variant where courier autonomy is not yet considered. After using this model, we move over to a more elaborate model where courier autonomy is taken into account.

4.1 Rolling horizon for the MRDP

As previously stated the MRDP is highly dynamic and myopic. Trying to schedule deliveries in the future is of virtually no use. For this reason it seems that making a delivery schedule 'on the go', so while the operating period is underway, results in the best solutions. A rolling horizon matching based algorithm seems most apt to attack this problem. This algorithm solves every f minutes a matching problem (section 4.1.2) to assign orders to couriers. After this matching procedure has taken place a commitment strategy (section 4.3) decides if these decisions are postponed or communicated to the corresponding courier. If an order is ready way into the future it is likely to be postponed and therefore the algorithm focuses on a subset of upcoming orders, with ready times within the assignment horizon, $U_t = \{o \in O_t : e_o \leq t + \Delta_u\}$, where t is the optimisation time.

4.1.1 Target bundle size and route creation

In an ideal world when an order is ready we deliver it immediately, but in the real world we should realize we have limited couriers and sometimes more orders than drivers available. For this reason it is suitable to have a courier deliver multiple order in a so called bundle.

To know how many orders should ideally be in a bundle we define a target bundle size Z_t. Z_t could be fixed over the operation period but to make our solution more robust we let the target bundle size depend on the optimisation time t. To define the Z_t we use the ratio between orders ready to be picked up and couriers available. The orders ready to be picked up at time t are those in subset U_t, couriers available at optimisation time t are those within interval $t + \Delta_2$. The aforementioned $\Delta's$ are set by a tuning process.

Once Z_t has been calculated we can make the set of bundles to be assigned, S. This set is made by Procedure 1 reproduced from Reyes et al. (2018), described below.

```
input  : $U_{t,r}$ set of upcoming orders at restaurant $r$, sorted by non-decreasing ready times
         $Z_t$ target bundle size
         $k_t^r$ number of couriers available at $r$
output: $S_r$ set of bundles from restaurant $r$ ready for linear assignment model
Target number of bundles to create $m_r = \max\left(k_t^r, \lceil\frac{|U_t,r|}{Z_t}\rceil\right)$
 for $o \in U_{t,r}$ do
    Find route $s \in S_r$ and insertion point $i_s$ for order $o$ at minimum increase in cost. Where cost is
    $\sum_{(p,q)\in s} TravelTime + \beta \sum_{p\in s} ServiceDelay$
    if $|s| > Z_t$ and insertion decreases route efficiency then
       disregard $s$ for order $o$ and find next best route for order $o$
    end
    Insert $o$ in route $s$ at position $i_s$
end
Local search, remove-reinsert
 for $o \in U_t, r$ do
    Remove $o$ from current route
    Given the existing routes in $S_r$ find the optimal route $s$ and insertion position $i_s$
    Reinsert $o$ at that position
end
```

This procedure will bundle upcoming orders from restaurants if they increase route efficiency. Route efficiency is defined as the time per order delivered in that bundle. An order is only inserted into a bundle if it increases route efficiency, this results in lower CtoD averages. Also note that parameter β can be seen as the freshness parameter as it controls the importance of service delays in the algorithm.

4.1.2 The linear assignment model

To assign couriers and routes in a optimal fashion we propose the below explained assignment model. This model will maximise the number of orders delivered minus delays in the pickup process. So in this model we try to assign couriers to bundles in order to get as many orders to clients as possible per time unit.

A simple assignment model is one with bipartite matching, a linear model. Let us first introduce some notation before we elaborate on the model. Let N_s be the number of orders in bundle s. Let $p_{s,c}$ be the pick up time of bundle s assigned to courier c. Let $d_{o,c}$ be the drop-off time of order $o \in s$ when assigned to courier c and let $\gamma_{s,c}$ be $\max\limits_{o\in s}\{d_{o,c}\}$. So $\gamma_{s,c}$ would be the drop-off time of the last order in bundle s. Let θ_o be the maximum ready time of all orders in bundle s. We also introduce a penalty term ω for delays in the pick-up process. Lastly a decision variable $x_{s,c}$, which is one if bundle s is assigned to courier c and zero otherwise. The model then becomes:

$$\max \quad \sum_{c\in C}\sum_{s\in S}\left(\frac{N_s}{\gamma_{s,c}-e_c}-\omega\left(p_{s,c}-\theta_o\right)\right)x_{s,c} \tag{1}$$

$$s.t. \quad \sum_{s\in S} x_{s,c} \leq 1 \qquad \forall c \in C \tag{2}$$

$$\sum_{c\in C\cup\{0\}} x_{s,c} = 1 \qquad \forall s \in S \tag{3}$$

$$x_{s,c} \in \{0,1\} \qquad \forall s \in S, c \in C \cup \{0\} \tag{4}$$

In the objective function we maximise over all couriers $c \in C$ and all bundles $s \in S$. The left term inside the parentheses divides number of orders per bundle by the time it takes to execute the assignment. Note that e_c is the time when courier c becomes available for a new assignment. This first term within parentheses is the number of orders per time unit. The second term inside is a penalty term for delays in the pickup process. We take the pickup time minus the ready time of bundle s, this results in the pickup delay. So in total we maximise the number of order per time unit minus delays in the pickup process. Restrictions 2 and 3 are standard assignment constraints, where restriction 2 ensures that one courier does not have to deliver two or more bundles at optimisation time t. Restriction 3 ensures that every bundles s is assigned to exactly one courier. The set of couriers also uses an artificial courier (notation $\{0\}$) to assign leftover bundles.

4.2 Assignment process

In order to make sure orders do not get infinitely delayed (not assigned) we propose a priority scheme. Where orders are divided into three groups. Group I, the highest priority group, contains orders that cannot be delivered at the target drop-off time. Group II, orders not in Group I which cannot be picked up at their ready time. Group III, orders not in Group I and Group II. Bundles are assigned the highest priority of all orders in the bundle. In this way the highest priority orders can get assigned first instead of not getting assigned because there are not enough drivers available.

4.3 Commitment strategy

For the commitment strategy in the rolling horizon MRDP, this study uses a "lazy" strategy. In this strategy we can make a final, partial or no commitment. A final commitment instructs courier c to pick up and deliver an order/bundle. A partial commitment instructs courier c to drive to restaurant r and wait there for further instructions, this courier is guaranteed to deliver the orders assigned to them by model 4.1.2 and possibly more. No commitment is made when courier c is not ready for a new assignment. For each assignment (s, c), bundle s and courier c, the commitment strategy follows:

1. If c can reach restaurant r_s at time $t+f$ and all orders in s are ready by $t+f$, make final commitment.
2. If c cannot reach the restaurant r_s but is expected to deliver last order before $t+f$ make a partial commitment.

3. If c can reach the restaurant r_s but the orders in bundle s are not yet ready, make a partial commitment.

4. If c cannot start new assignment at time $t + f$ ignore this courier.

5. Exception: If an order in s has been waiting for more than x minutes make a final commitment to deliver this order/bundle.

The idea behind this commitment strategy is that the courier can be matched again at the next optimisation but this time the composition of the bundle may change (become larger). When the courier is busy till $t + f$ waiting till the next optimisation will not effect CtoD or RtoP since the courier can be matched by the next optimisation. We call the commitment strategy, the two-stage additive commitment strategy. Two-stage because we divide going to restaurant and going to client address as two stages. We call this strategy "additive" because if we partially commit a courier c to a bundle s, we force c to deliver at least the orders in s possibly more.

4.4 Courier autonomy

New questions arise when using independent drivers. For instance when do they accept or reject an order and what would be the optimal ratio of independent drivers to employees, in order to minimize Click-to-door or other performance measures? This ratio of independent drivers to employees is defined as follows: Taking the set of available couriers at optimisation time t, $x\%$ is independent rounded downward if this can not be exactly reached. In the next subsections we dive into the methods used to evaluate freelancers performance in the system.

4.4.1 Accepting or rejecting an order

The first question that needs answering is when do these independent couriers accept an order, given that they get paid the same fixed amount per order p_2. The independent couriers have their own vehicle. They are willing to deliver an order if it is not far from their own location. It is assumed that these drivers have a fixed travel cost per minute c_i. The online meal ordering platform wants to deliver orders to optimise some performance measure, like cost, CtoD or RtoP. As mentioned before in section 3.3 a freelancers accepts an order if $(c_i \times w_s) \times \Omega \leq p_2$.

4.4.2 Conflicting order assignments

A situation could arise where two or more independent drivers would like to deliver an order, for this situation this study proposes an auction system. The independent drivers that would like to deliver bundle s can bid on this bundle one time and the lowest bid gets to deliver this bundle. This results in cheaper delivery which is good for the online meal-delivery platform. As previously stated the couriers bid their lowest possible accepting price. This auction system will take place between Procedure 1 and model 4.1.2 the bundle is then taken out of the upcoming orders and is directly committed to the courier with the lowest bid.

This type of system can also be proposed when orders are ready but are not yet picked up. In our system we will increase the compensation price per z minutes delay in the pickup process. If a bundle has multiple orders then all compensation will increase even if only one order in that bundle has z minutes delay. The amount that the order will be increased with is set by a compensation multiplier, a parameter in the model. This way less delays will occur since the orders that are ready are immediately more valuable to deliver for independent drivers. Remember that employees will make the same compensation p_2 regardless of this auction system or compensation increase, only independent drivers participate here. This increase will come in the beginning of the optimisation so that freelancers can adapt their accept/reject strategy accordingly.

The auction system is performed before we assign bundles to couriers in order to lower costs, this could however increase CtoD and RtoP since these freelances could be further away from a restaurant then employees, but on the other hand increasing the compensation price could lead to a decrease in CtoD and RtoP. It is therefore interesting to see what ratio of employees to independent drivers would minimise CtoD, RtoP and costs.

4.4.3 The independent driver model

The new model with independent drivers does not differ much from the original model. We do however need to model the 'accepting'-order-constraint. For this constraint we define a new binary parameter $y_{s,a}$ which is one if driver a is willing to deliver bundle s and zero otherwise. This is of course always one if $a \in C$ but if $a \in I$ it will depend on w_s and Ω. Note that this $y_{s,a}$ is defined before the assignment model is used.

$$\max \quad \sum_{a \in A} \sum_{s \in S} \left(\frac{N_s}{\gamma_{s,a} - e_a} - \omega \left(p_{s,a} - \theta_o \right) \right) x_{s,a} \tag{5}$$

$$s.t. \quad \sum_{s \in S} x_{s,a} \leq 1, \qquad \forall a \in A \tag{6}$$

$$\sum_{a \in A \cup \{0\}} (x_{s,a} \times y_{s,a}) = 1 \qquad \forall s \in S \tag{7}$$

$$x_{s,a}, y_{s,a} \in \{0,1\} \qquad \forall s \in S, a \in A \cup \{0\} \tag{8}$$

The above model uses the set of all drivers A, employees and independent drivers, to maximise orders delivered per time unit. The only added parameter is $y_{s,a}$ which ensures that if driver a is not willing to deliver bundle s it will not get assigned that bundle. The assignments that follow from this model will go through the same commitment strategy proposed in section 4.3. Note that independent drivers do not listen to partial commits so those are left out of the second and third step.

5 Results

This section consists of the results gathered by replicating Reyes et al. (2018). We go over the effects of the optimisation frequency f, the assignment horizon Δ_u, the two-stage additive commitment strategy and lastly the effects of bundling. The default model we use has an optimisation frequency of $f = 5$, allows bundles, it uses the two-stage additive commitment strategy, and parameters $\Delta_1, \Delta_2, \Delta_u$ are all $2{*}f$. This default model is used to examine the effects of the aforementioned variations. Lastly Parameters β and θ are chosen by selecting some values that minimize the CtoD. The dataset used is (0-10)o100t100s2p100 (from the github of Reyes et al. (2018)), which is all parameters on normal and using optimised shift schedules.

After going over the results with regards to replicating Reyes et al. (2018) we continue to examine what effects autonomous couriers have on CtoD, RtoP and costs. The extended model uses four parameters that will be examined, freelance percentage, compensation multiplier, profit margin Ω and z minutes.

5.1 Impact of optimisation frequency

Below the differences between optimisation frequency are shown in relation to the performance measures %undelivered, CtoD, RtoP, cost per order and orders per bundle. As seen in Table 1 using lower optimisation frequencies increases CtoD marginally and increases RtoP by a large portion. This increase in RtoP is due to a much smaller assignment horizon, this went from 10 to 4. This results in less foresight so the model react slower which results in higher RtoP's.

	Optimisation interval				paired differences	
	5 minutes		2 minutes		(f = 2) - (f = 5)	
	mean	std	mean	std	mean	std
%undelivered	0.01	0.02	0	0	-0.01	-0.02
CtoD mean	31.83	1.34	31.84	1.55	0.01	0.22
RtoP mean	1.86	0.22	2.06	0.60	0.20	0.38
cost per order	11.21	1.18	11.24	1.17	0.03	-0.01
orders per bundle	1.11	0.03	1.09	0.03	-0.02	0

Table 1: Performance differences using default optimisation frequency (5 minutes) and a more frequent optimisation (2 minutes)

5.2 Impact of assignment horizon

In Table 2 the differences between assignment horizon are shown. We clearly see an increase in CtoD and RtoP means when increasing the horizon from 10 to 20 minutes. This longer horizon leads to more orders per bundle and a small decrease in cost per order. So depending on what is preferred (CtoD or costs) one can choose to use a 20 min or 10 min assignment horizon. A reason for this rise in delivery times is that with a larger assignment horizon more orders will be bundled, since we see further into the future. This does however also result in lower costs.

	Assignment Horizon				paired differences	
	10 minutes		20 minutes		(u = 20) - (u = 10)	
	mean	std	mean	std	mean	std
%undelivered	0.01	0.02	0	0	-0.01	-0.02
CtoD mean	31.83	1.34	32.52	2.73	0.69	1.39
RtoP mean	1.86	0.22	2.33	1.62	0.47	1.40
cost per order	11.21	1.18	10.88	1.05	-0.33	-0.13
orders per bundle	1.11	0.03	1.13	0.04	0.02	0.01

Table 2: Performance differences using default horizon (10 minutes) and longer (20 minutes) horizon

5.3 Impact of the Commitment Strategy

Table 3 below shows the differences between single- and two-stage commitment strategies. The two-stage additive commitment strategy is better in every performance measure except for a small increase in cost per order. What is also important to note is that orders undelivered more than double (from 0.005 to 0.013), which is not seen in Table 3 below because of rounding. This of course undesirable and therefore the two-stage additive commitment strategy should be used instead of the more simple single-stage commitment strategy. A reason for this 3% and 61% increase in CtoD and RtoP respectively is that there are less orders per bundle. What happens is that every assignment is committed directly, which results in some orders from the same restaurants being delivered separately while they could have been bundled. Less couriers will be available (because of the direct commitments) which leads to more orders having to wait to get delivered.

	Commitment Strategy				paired differences	
	Two-stage		Single-stage		(single) - (two-stage)	
	mean	std	mean	std	mean	std
%undelivered	0.01	0.02	0.01	0.02	0.01	0
CtoD mean	31.83	1.34	32.73	1.52	0.90	0.18
RtoP mean	1.86	0.22	2.99	0.43	1.13	0.19
cost per order	11.21	1.18	11.20	1.17	-0.01	-0.01
orders per bundle	1.11	0.03	1.08	0.03	-0.03	0

Table 3: Performance differences using default two-stage additive optimisation and single stage simple optimisation

5.4 Impact of bundling

A final variation of the rolling horizon matching-based algorithm we used is allowing bundling or not. Below in Table 4 we see the differences when using bundles (default) compared to no bundles. The CtoD is marginally decreased if we do not allow bundles but the RtoP increases around 25%, the cost per order also decrease. So depending on what performance measures are most important to the user one can chose to allow bundles or leave them out of the algorithm. This result is due to the fact that when orders are in a bundle the CtoD increases because if an order is second on the delivery route it will get delivered slower than if it was delivered directly. The perfect route for one order is of course that from the restaurant to the address, not with stops in between. The increase in RtoP is most likely due to the same reason as the commitment strategy. Less couriers are available since no bundles, so this means that orders will have to wait at the restaurant longer.

	Bundling				paired differences	
	Yes		No		(No) - (Yes)	
	mean	std	mean	std	mean	std
%undelivered	0.01	0.02	0.02	0.08	0.01	0.06
CtoD mean	31.83	1.34	31.30	1.50	-0.53	0.16
RtoP mean	1.86	0.22	2.11	0.38	0.25	0.16
cost per order	11.21	1.18	11.10	1.15	-0.11	-0.03
orders per bundle	1.11	0.03	1	0	-0.11	-0.03

Table 4: Performance differences allowing bundles and not allowing bundles

5.5 Impact of independent drivers

To measure the impact of independent drivers this study first adjusts the ratio of employees to independent drivers. To see what the result is on the cost per order (Figure 1), CtoD (Figure 2) and RtoP (Figure 3). After doing this we do a sensitivity analysis on the last three parameters, namely the compensation multiplier (Figure 4), profit margin Ω (Figure 5) and z minutes (Figure 6). The default setting of these variables are 1.2, 1.2, and 2 respectively and 20% for the freelance percentage. For the compensation multiplier and Ω we use a range from 1 to 2 (steps of 0.05) and for the z minutes a range from 0 to 10 (steps of 1). Note that the percentage of freelancers in the figures below is the attained percentage which is lower that the set percentage since we round downward if the exact freelance percentage cannot be attained at a certain optimisation time t (see section 4.4).

In Appendix A the relation between up-sold orders and freelancers are shown (orders with compensation above normal), also for down-sold (orders with compensation below normal) orders.

In the below Figure 1 we see a negative relation between freelance percentage and costs. This is due to the higher likelihood that an auction will take place which will decrease paid compensation. In Figure

2 we see a positive relation between freelancers and CtoD, this is most likely due to the fact that we first cannot partially commit freelancers, also freelancers bid on orders to deliver which focuses more on the price of the order and not on how far away this driver is from the restaurant. Note that also these freelancers will not move after they have delivered their order so they will most likely be further away from the restaurants then employees would be. The same reasoning can be applied when examining the positive relation with RtoP in Figure 3.

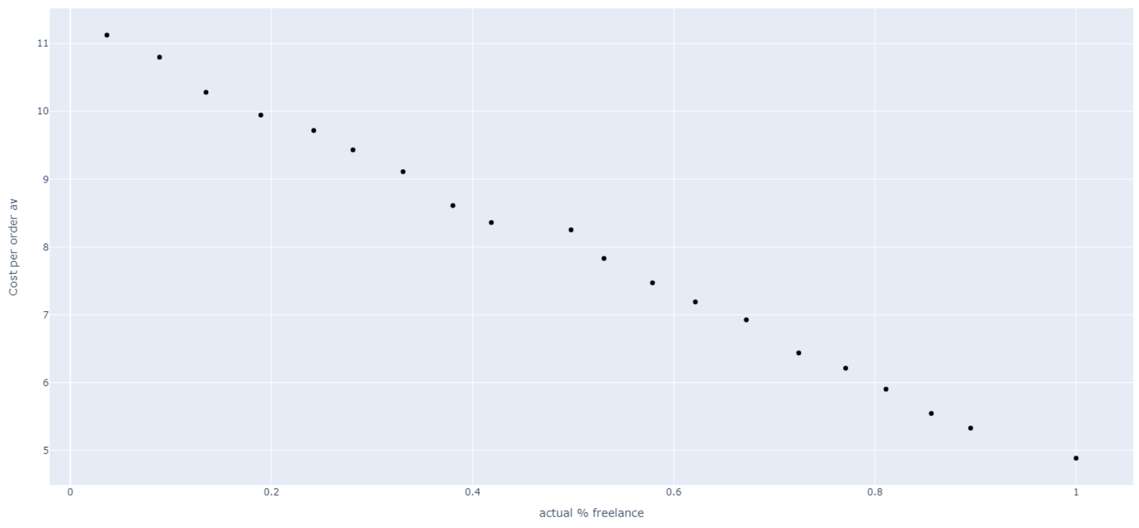


Figure 1: Freelance % in relation to cost per order

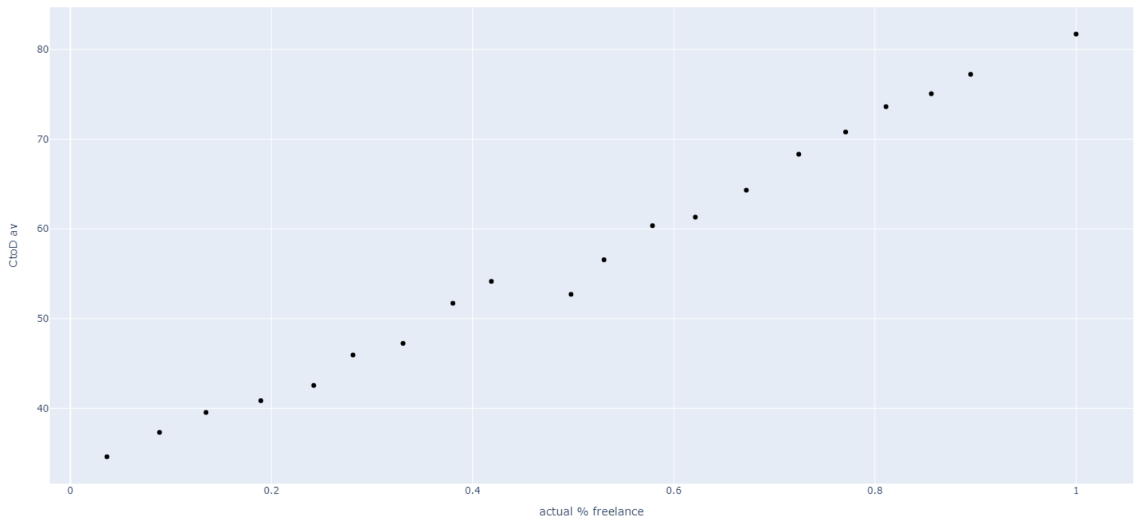


Figure 2: Freelance % in relation to CtoD

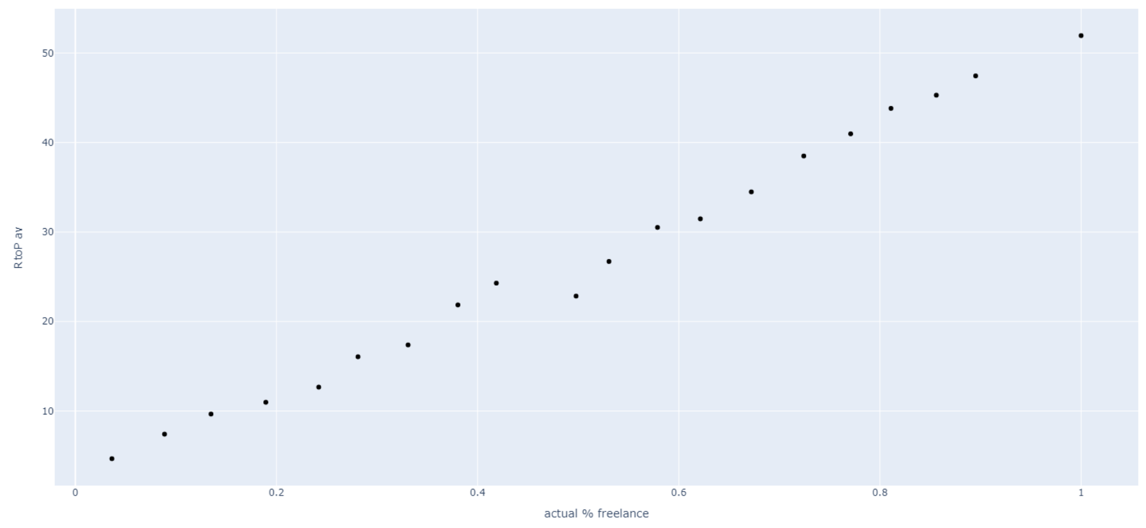
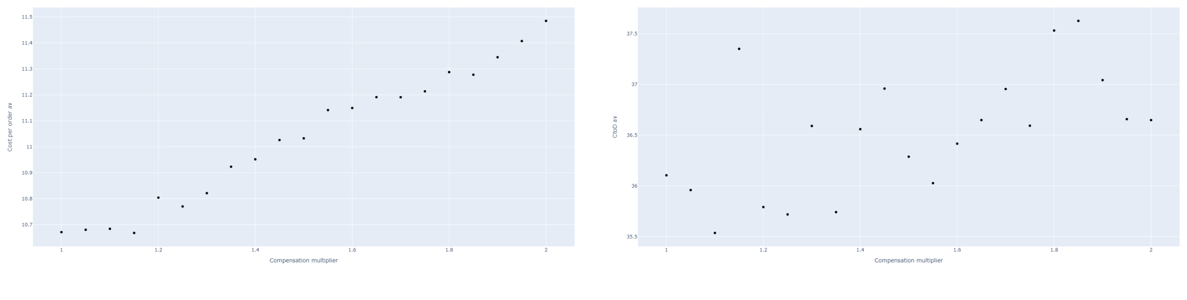


Figure 3: Freelance % in relation to RtoP

In Figure 4 below we see a plot with left (4a) the positive relation between compensation multiplier and costs, and right (4b) the positive relation between CtoD and compensation multiplier although the observations here are more scattered out. The relation with costs is due to the fact that a higher compensation will lead to higher delivery costs all things equal. What is interesting is that a higher compensation multiplier also leads to a higher CtoD, a reverse effect of what we want. It seems that when orders have a higher compensation drivers from further away can now accept this order and deliver it, which leads to a higher CtoD.

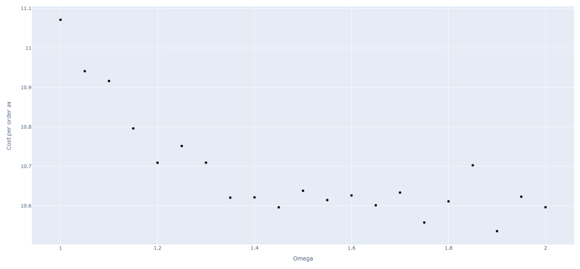


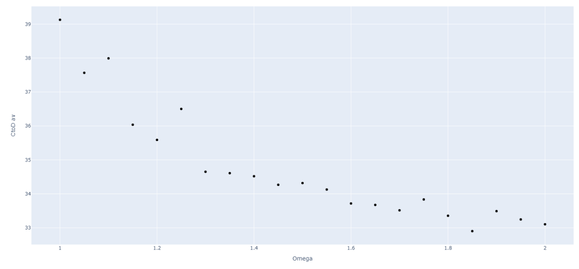
(a) View cost per order

(b) View CtoD

Figure 4: Relation between compensation multiplier, CtoD and Costs

The Figures 5 below show the relation between Ω, costs and CtoD. On the left we see the negative relation between costs and Ω, the effect however starts strong till Ω = 1.4 then the effect dies down in the end. On the right we see the same effect regarding the CtoD, which also starts stronger and dies down in the end. The negative relation with costs is due to the fact that the bids a freelancer makes are lower since these freelancers accept an order only if its closer to them and thus make less costs and will bid lower. The higher the profit margin the closer a driver needs to be to the restaurant to deliver an order (further away leads to more travel costs), so with a higher profit margin only drivers that are closer to the restaurant accept an order and so the CtoD will decrease when Ω increases.

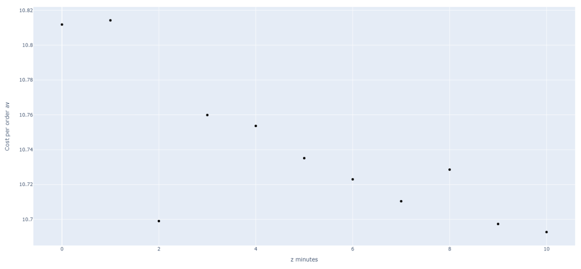


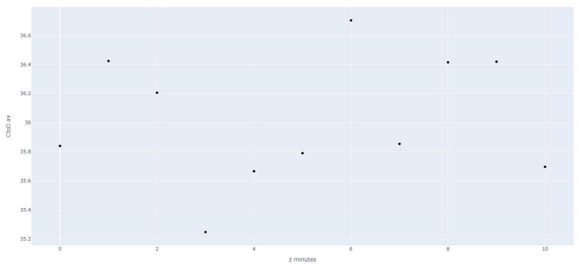


(a) View cost per order (b) View CtoD

Figure 5: Relation between Ω, CtoD and costs

Below Figure 6 shows the relation between z minutes and the performance measures. There seems to be no discernible relation between CtoD and z minutes. The costs however do seem to decrease when z minutes increases. Which would be due to lower compensation prices since orders do not get higher pay as often with a higher z minutes.





(a) View cost per order (b) View CtoD

Figure 6: Relation between z minutes, CtoD and costs

6 Conclusion

This study has imposed the central research question: "How do we operate a meal-ordering platform so that last-mile meal orders are not only delivered within a given time frame but also delivered profitably?" The results from the replication show that we have the same conclusion as Reyes et al. (2018). We found that the default rolling horizon algorithm with a larger optimising interval, lower assignment horizon, the two-stage additive commitment strategy and allowing bundles is the best model for minimising CtoD, RtoP and costs per order. This is in line with our intuition and the original paper.

After the replication we went to extending the model to incorporate freelancers. To answer what percentage of freelancers would be optimal a meal ordering platform should first ask themselves what performance measure they value most. If they want to be the cheapest, more freelancers lower the costs, but if they want faster delivery (low CtoD, low RtoP) lower percentages of freelancers (no freelancers would be best) are better. Note that this paper did not account for the possibility that freelancers move autonomously, this could have a positive effect on delivery times since drivers would be closer to restaurants. It also seems that a low compensation multiplier is best for lower costs and lower CtoD. The parameter Ω, also profit margin, has a negative relation with costs and delivery time, so the higher the profit margin the lower the costs and CtoD. The effect is however stronger at the start till $\Omega = 1.4$ and becomes less strong in the end. The parameter z minutes does not seem to have a relation with CtoD, but it does have a negative relation to costs.

The answer to the central research question is not simple, since we always have a trade-off between costs and delivery time. The best trade-off would be that the online meal delivery platform defines a maximum allowed average CtoD and then allows freelancers up to that level to lower costs. In this situation the platform still has fast delivery but lowers it costs by using freelancers.

This paper seems to say that freelancers should only be used to lower costs but that may not be the case when further research adds the possibility of autonomous moving or better distance measures to ensure CtoD cannot go over a certain threshold. For further research many things can be explored. For instance in order to lower delivery times when using freelancers the possibility of autonomous moving can be explored. Freelancers can move to the most profitable places for them defined by some compensation measures. When independent couriers move to other places they might be closer to future orders which would lower CtoD and RtoP, this could be an interesting direction to research. What is also left out in this paper is the stochasticity of orders. This study uses deterministic ready times, service times and travel times. In the real world this would not be the case and incorporating some distribution for ready times and delivery times would definitely change results and possible change the optimal values of parameters used in our default model and the extended model with freelancers. Lastly we left out the possibility for platforms to exclude some restaurants if they are not profitable, or paying for faster delivery. These two things could be interesting to research since it would increase profits for the meal delivery platform.

References

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, *223*(2), 295–303.

Archetti, C., Savelsbergh, M., & Speranza, M. G. (2016). The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, *254*(2), 472–480.

Berbeglia, G., Cordeau, J.-F., & Laporte, G. (2010). Dynamic pickup and delivery problems. *European journal of operational research*, *202*(1), 8–15.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of operations research*, *153*(1), 29–46.

Coslovich, L., Pesenti, R., & Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, *175*(3), 1605–1615.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management science*, *6*(1), 80–91.

Dewey, C. (2021). *Analysis — the insane $43 billion system that gets food delivered to your door.* WP Company. Retrieved from `https://www.washingtonpost.com/news/wonk/wp/2017/08/08/the-insane-43-billion-system-that-gets-food-delivered-to-your-door/`

Gendreau, M., Laporte, G., & Semet, F. (2001). A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. *Parallel computing*, *27*(12), 1641–1653.

Hirschberg, C., Rajko, A., Schumacher, T., & Wrulich, M. (2020). *The changing market for food delivery.* McKinsey & Company. Retrieved from `https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-changing-market-for-food-delivery`

Ook traditionele supermarkten storten zich op flitsbezorging. (2022). RTL Z. Retrieved from `https://www.rtlnieuws.nl/economie/bedrijven/artikel/5302299/flitsbezorgers-albertheijn-spar-jumbo-gorillas-thuisbezorgd`

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11.

Reuters. (2016). *How amazon is making package delivery even cheaper.* Fortune. Retrieved from `https://fortune.com/2016/02/18/amazon-flex-deliveries/`

Reyes, D., Erera, A., Savelsbergh, M., Sahasrabudhe, S., & O'Neil, R. (2018). The meal delivery routing problem. *Optimization Online*, *6571*.

Ulmer, M. W., Thomas, B. W., & Mattfeld, D. C. (2019). Preemptive depot returns for dynamic same-day delivery. *EURO journal on Transportation and Logistics*, *8*(4), 327–361.

Wang, X., Agatz, N., & Erera, A. (2018). Stable matching for dynamic ride-sharing systems. *Transportation Science*, *52*(4), 850–867.

Appendix

Appendix A: Upsold and Downsold orders

Figure 7 shows the positive relation between freelancers and downsold orders, this is of course due to the fact that more freelancers leads to a higher change that two or more independent drivers accept an order. Below in Figure 8 we see a parabolic line. First the upsold orders go up and then reach a maximum and lower. Most likely due to the fact that when freelancers increase enough there will always be two independent drivers who would like to deliver which would result in a down sell of the order.

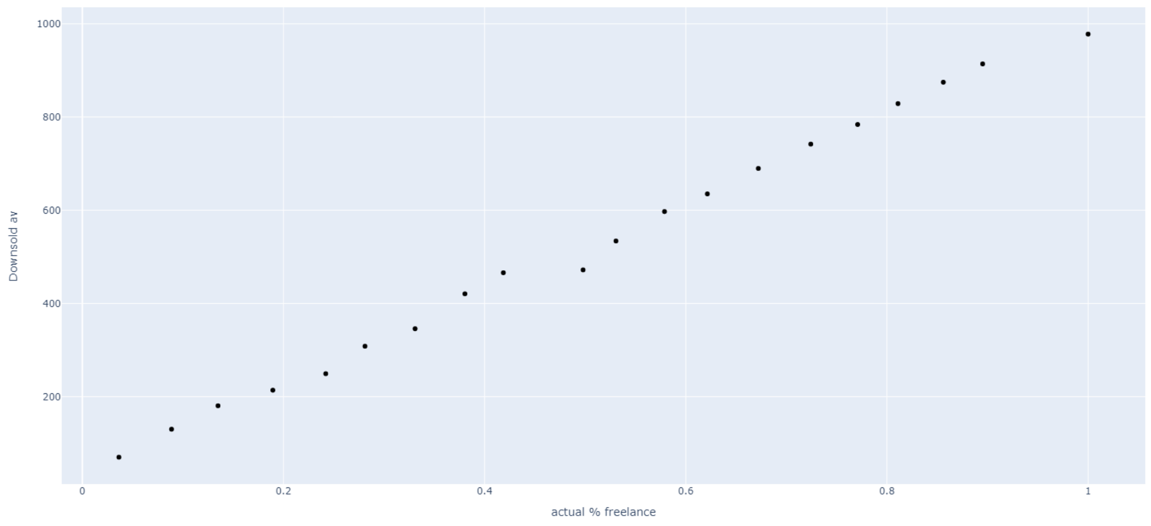


Figure 7: Freelance % in relation to orders sold below asking

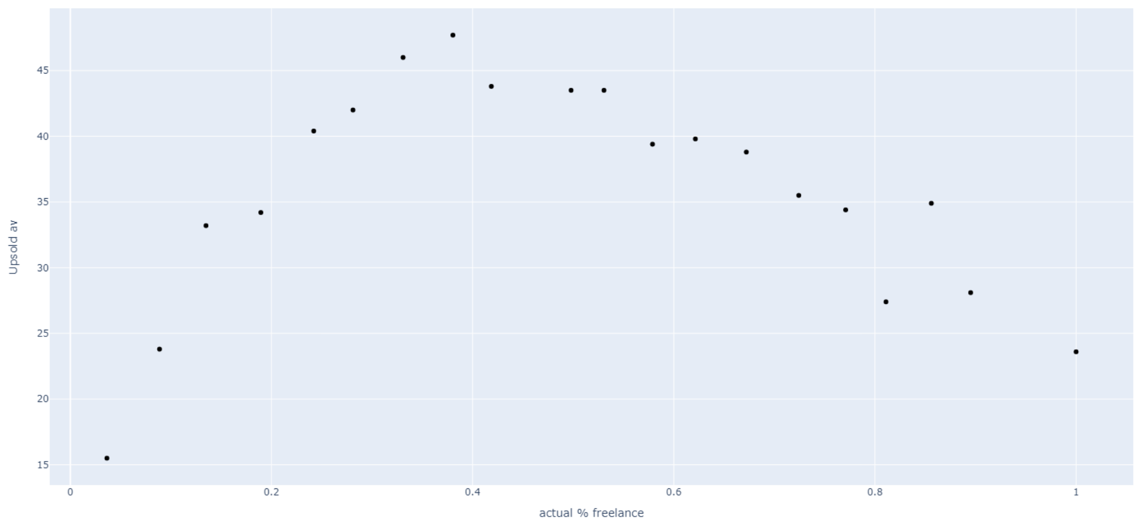


Figure 8: Freelance % in relation to orders sold over asking

Appendix B: Code construction and explanation

The code created for this study is comprised of three parts. First the main code, “rollinghorizon.py”, where we use Procedure 1, model 4.1.2 and the rolling horizon algorithm. The second file “orders.py” houses the Object Oriented Programming language. In this file the Order, Route and Courier object are programmed. In the last file “data.py” the data written by the first file are made into figures and Tables.

Rolling Horizon

There are two methods that execute Procedure 1 and model 4.1.2. They are named as such. If one wants to print the correct results they should change the first for loop (not in the two methods but in general code) to house the variable they want to change. This variable can be either of the four variables, freelance %, profit margin, z minutes and the compensation multiplier. Please not that you should then also change the default setting of the variable you would like to examine. A more precise overview of the code is elaborated in the comments in the code itself.

Object Oriented Programming

There are three classes in this file, first the Order class. This class defines an order, interesting attributes are “self.driver” which is True if order is delivered by an employee and False if delivered by a freelancer. The second class defined a Route or Bundle (used both in the code but mean the same). The last class defines couriers which can be freelancers or employees. All methods of these classes are explained in the comments of the code.

Data

Once you have ran the code in file “rollinghorizon.py” the results are written in a .csv file using the last few line in that file. This .csv file is then read by the “Pandas” package of Python. Using the .describe() method one can see the interesting features of the data such as mean and standard deviation. For making the figures the package “Plotly” is used.