IE492 – Project Report

Optimization of Vehicle-Order Matching Problem via Simulation Framework

Advisor: Mustafa Gökçe Baydoğan

Group Members:

Ahmet Buğra Taksuk-2017402168

Tufan Berk Tuğ – 2017402048

Dorukhan Kılınç – 2017402093

Abstract

With the rise of e-commerce in the last few years, online meal delivery has become more and more popular. To meet the rising demand and compete in this market, meal delivery companies try to develop better strategies to optimize the customer satisfaction. However, it is often costly to test out alternative solutions because it may result in loss of profit, which makes companies hesitant to try new strategies. The aim of this study is to provide a computational simulation framework for the use of testing new strategies to be used in meal delivery systems so that the costly side of real-life testing is avoided, and different strategies can be tested in identical situations with less experimentation times than A/B tests.

The base simulation is taken from a paper by Rojas [1], which was used to simulate meal deliveries in Colombia using OSRM for route calculation. The framework is modified so that it can simulate meal deliveries in Istanbul. The data that is used to prepare the simulation input is taken from Trendyol. With further added features to the simulation, it is possible to implement different order courier assignment strategies, perform demand management, and yield a variety of performance measures such as courier utilization rates and mean delivery time.

Özet

E-ticaret sektörünün yükselmesiyle birlikte, çevrimiçi yemek teslimatı son birkaç yılda giderek daha popüler hale geldi. Artan bu talebi karşılamak ve bu pazarda rekabet edebilmek için yemek dağıtım şirketleri, müşteri memnuniyetini optimize edecek daha iyi stratejiler geliştirmeye çalışıyor. Ancak, bu çözümleri uygulamak genellikle maliyetli, çünkü kâr kaybına neden olabiliyor ve bu da şirketlerin yeni stratejiler denemekte tereddüt etmesine sebep oluyor.

Bu çalışmanın amacı, gerçek hayat testlerinin maliyetli tarafının önüne geçilmesi ve benzer amaçlarla daha kısa deney süresi sağlayan, yemek dağıtım sistemlerinde kullanılacak ve yeni stratejilerin test edilmesinde kullanımına yönelik bir simülasyon ortamı sağlamaktır.

Simülasyon, Kolombiya'daki yemek teslimatlarını simüle etmek için hazırlanan, Rojas’ın [1] bir makalesinden alınmıştır. Bu makaledeki simülasyon modeli rotalama için OSRM aracından yararlanır ve İstanbul'daki yemek teslimatlarını simüle edebilecek şekilde modifiye edilmiştir. Simülasyon girdilerini hazırlamak için kullanılan veriler Trendyol'dan alınmıştır. Simülasyona eklenen ilave özellikler ile farklı sipariş-kurye atama stratejileri uygulamak, talep yönetimi gerçekleştirmek ve kurye kullanım oranları ve ortalama teslimat süresi gibi çeşitli performans ölçümleri almak mümkündür.

1. Table of Contents

2. Introduction

E-commerce is an exploding business sector promising further growth potential in the near future. As a relatively young concept, it can be integrated with other sectors to create demand or share the existing demand. Ridesharing, meal delivery, and online shopping are some examples of such integrations using the power of the internet and e-commerce concept. It can be seen from Figure 1 that the overall e-commerce sales are increasing and also expected to increase over the years.

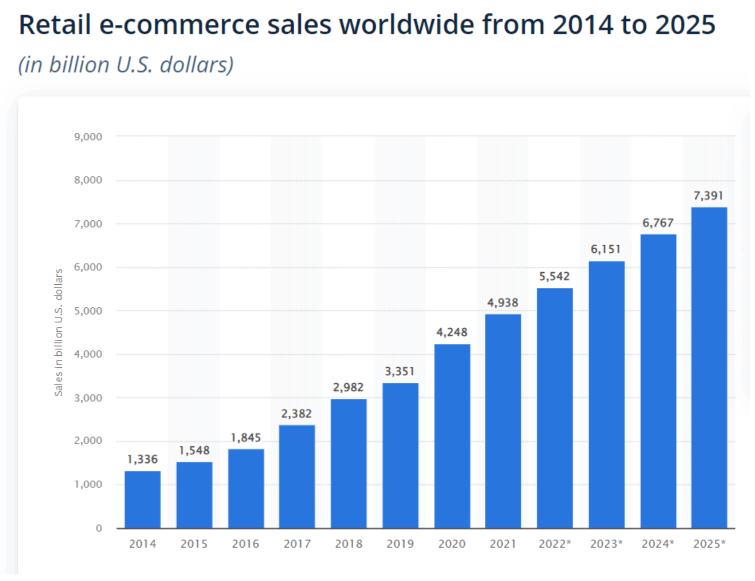


Figure 1- E-commerce sales worldwide

We can see a similar trend for Turkey also in terms of e-commerce growth. E-commerce expenditures for the population between the ages of 18-70 increased to 4,749 TL while e-commerce expenditure per capita increased 69% in 2021 compared to the previous year.

Among the domains of e-commerce with the highest increase compared to the previous year, food and supermarket delivery (13.9 billion TL) is at the second place with an increase of 162% annually. Therefore, it can be concluded that apart from the e-commerce growth in Turkey, Food and supermarket delivery concept is also becoming more and more demanding. (Figure 2)

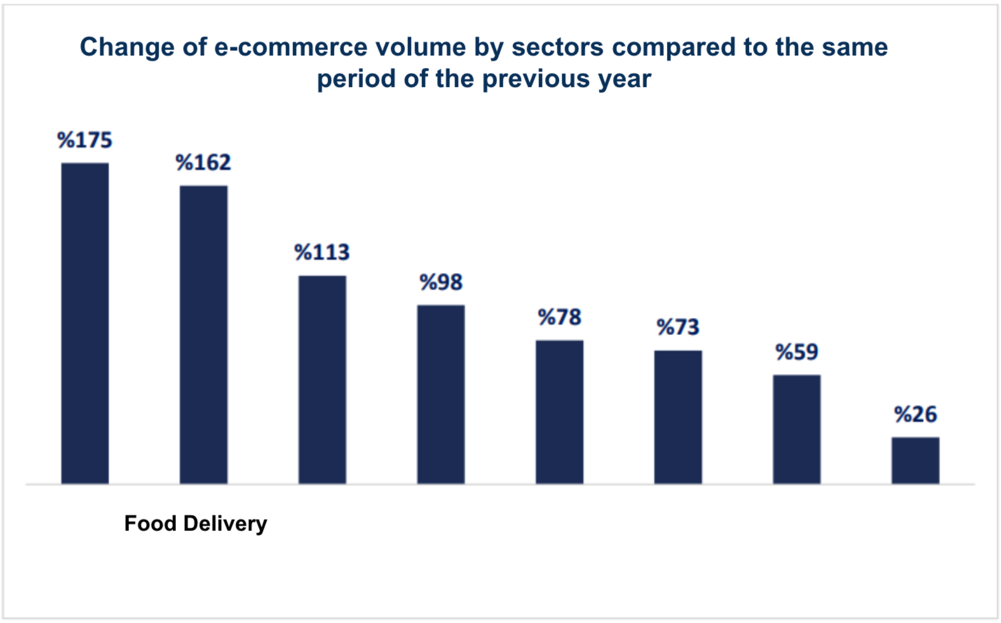


Figure 2- Change in e-commerce volume in 2021 by sectors

Speaking of e-commerce in Turkey, Trendyol is one of the leading companies in Turkey in terms of e-commerce with some other competitors, and the company is the data provider of this project. Trendyol GO, on the other hand, is the branch of Trendyol for the instant delivery service for food and grocery. Similar to its competitors in the food delivery market, they aim to develop a delivery system from every retailer including a variety of markets such as pet shops, butchers, florists for all users in Turkey, promising to deliver in limited time window. While mentioning the limited time, the most competitive aspect of online food delivery is delivering the food as fast as possible. For the Trendyol Go case, also, the main customer satisfaction metric is considered to be whether a delivery is made during the promised time window which is 40 minutes in our case identical to Trendyol's policy. This binary satisfaction metric is the policy performance evaluation metric in both cases.

That being the case, a fast delivery system needs a complex dispatching policy with different sub-policies for each type of item to handle the decisions to be made in this dynamic environment. Dispatching strategy needs a deep analysis in case of new service regions as well as optimization efforts for existing service regions, which can be costly to test in real life. For example, an inefficient set of parameters or a poor matching algorithm may cause loss of demand, which would lead to profit loss. On the other hand, a long time of testing is needed for the results of the experiment to be considered reliable. Lastly, even if there are two valid experiment results, it would be meaningless to compare them because obviously they aren’t from the exact same setting.

For that reason, a simplified simulation of the highly dynamic environment of meal delivery problem that may allow firms to test different strategies and decide on their parameters using controlled environment dynamics, test current algorithms’ performance in different edge cases such as demand shocks, bad weather conditions, or heavy traffic, and run plenty of experiments in shorter time, without jeopardizing customer satisfaction is proposed as solution. Additionally, a demand management module and dynamic speed adjustment are made to improve the simulation system and to make it closer to the real-life environments.

With the proposed simulation framework, it was seen that the simulation is able to work with different types of algorithms(MDRP and MDRP-Graph), different sets of demand management policies and different days where demand and supply changes. Also, it was seen that with the demand management module integrated to the simulation, the ratio of canceled orders due to imbalance in demand and supply is decreased whereas the customer satisfaction increased via radius reduction. Overall looking, Trendyol is able to simulate its highly dynamic environment with the help of simulation to tune their parameters and be ready for unexpected real-life cases.

3. Problem Definition and Assumptions

The problem in our case is basically the Meal Delivery Routing Problem (MDRP). Meal delivery routing problem (MDRP) asks for a matching of given orders with available couriers such that every order is matched to a courier while minimizing/maximizing a set of metrics. In this system, there are customers giving orders using the app and they send their locations with their orders. Then, if allowed, their orders are placed in the corresponding restaurants. Restaurants take the orders, prepare them, and update the status of these orders in the system. Couriers make the deliveries. Dispatcher, which can also be referred to as the company, is the actor in this system who makes the courier-order assignments. It is important to note that MDRP is not a problem that needs to be solved once, it needs to be solved repeatedly as the new orders arrive to the app. (Figure 3)

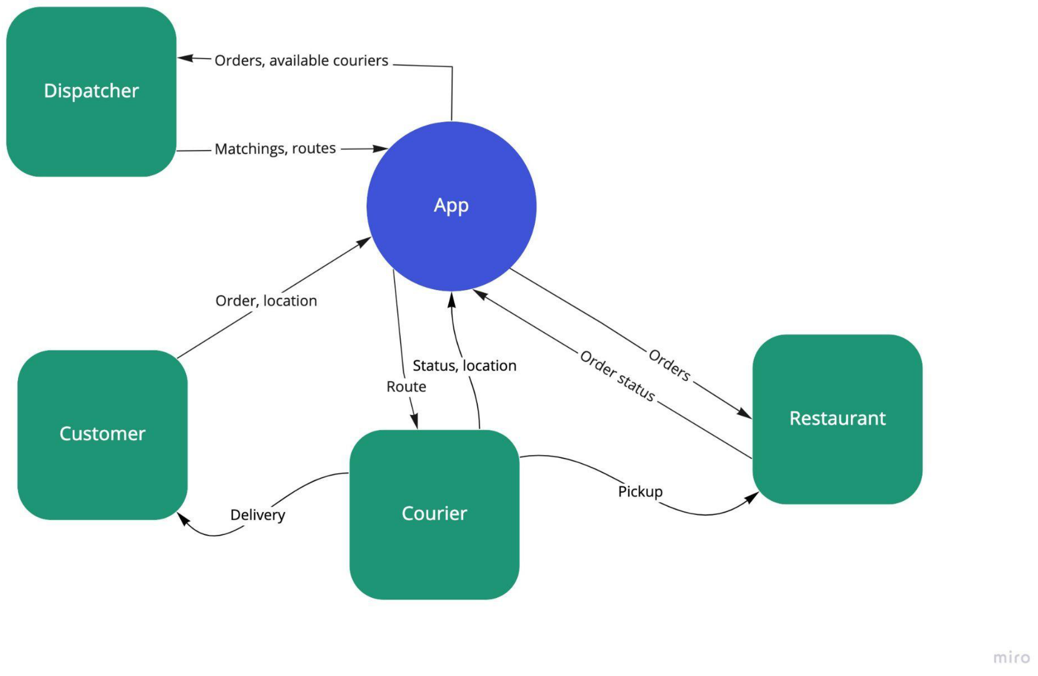


Figure 3- Flow chart of MDRP

There are several choices to be made based on the elements defined above before solving the MDRP, which may affect the overall outcome. Firstly, the algorithm that will be used to solve MDRP is one of the most crucial choices to make since MDRP is an NP-complete problem as stated by Reyes [2]. Therefore, it is impossible to find an exact algorithm solving the problem in a reasonable time, which introduces the choice of which heuristic to use. This project will try to include different heuristics in the solving of the MDRP. Originally, MDRP tries to find a matching such that each courier is assigned to one order. However, a courier can carry more than one order based on its capacity. Such assignments are called bundling and the heuristics that will be used will be allowed to do bundling. However, this will be only limited to the orders that are placed at the same restaurant to prevent the problem from getting too complex. In addition, Trendyol's current algorithm also only follows this rule for bundling.

Secondly, courier constraints are also factors to consider. Different number of couriers serving in a region can increase the customer satisfaction, with a trade-off that there is a risk of underutilized couriers. However, the set of couriers in this study is assumed to be fixed with known parameters. Similarly, the region in which MDRP is solved also changes the solution, which will be assumed to be fixed as well.

Lastly, as Yildiz et al [3] suggests, demand management policies can be used during times with high levels of demand so that the overall customer satisfaction is preserved with some lost orders. One of such policies will be radius reduction, i.e., the reduction of the radius a customer can give orders within. This will help to preserve the balance between active couriers and orders.

4. Literature Review

Regarding MDRP, Reyes [2] is the backbone of our study because it properly defines MDRP, suggests a set of useful metrics to be checked, and gives a heuristic integer programming model to satisfy these metrics. It further proceeds with a basic event-based simulation framework written in C++ to test the performance and make comparisons. However, most of the variables in these frameworks such as meal preparation times, delivery times and average courier speeds are assumed to be deterministic and invariant over time, which is an oversimplification of real life. Thus, we decided to take only the heuristic and the metrics it suggests. Lastly, Reyes [2] emphasizes the effect of how often the MDRP is solved in a dynamic environment. We note this suggestion as a feature to be considered during the setup of simulation.

Regarding the simulation, we found two papers with different simulation frameworks and each defining new heuristics, Rojas [1] and S. Paul [4]. They were both event-based simulations written in Python using public libraries SimPy for simulation and PuLP for linear programming. The main difference between these simulation frameworks was that Rojas [1] introduced probabilistic parameters to the simulation while S. Paul et al [4] assumed deterministic parameters similar to Reyes [2]. In addition, the framework introduced by Rojas [1] was written in an object-oriented manner, which made it easier for additions. Therefore, we decided to move on with the framework introduced by Rojas [1]. In addition to the framework, Rojas [1] also introduces a network flow model for solving MDRP. We will also include this model in our study.

Lastly, regarding demand management, Yildiz et al [3] argues that one of the main strategies in the solving of MDRP in a dynamic environment is the demand management strategy so that during the high demand levels or the cases where the number of couriers is not enough for the satisfaction of the whole demand, the overall customer satisfaction is preserved while minimizing the profit lost. It further suggests radius reduction, which will be discussed in detail.

5. Mathematical Models Used in the Simulation

a. Integer Programming Model

First model is the integer programming model for MDRP suggested by Reyes [2]. The mathematical model and the parameters used in the model with their definitions can be found in Figure 4. One important notion is the notion of a route, that is the route a courier needs to take to deliver all the orders assigned to him. Another important term is the cost variable. Notice that the problem is a maximization problem, therefore the cost variable is associated with a negative cost. Considering this variable, there are two terms contributing to it. First one denotes the total number of orders the courier will carry during his route divided by the duration of his route, which denotes the deliveries in unit time. On the other hand, the second term is a penalization of the delay in the last order. The model tries to maximize their sum using a binary decision variable x taking the value 1 if a courier is assigned to a route. The constraints assure that the assignment rules are not violated. Lastly, a dummy courier is introduced for the unassigned orders. (Figure 4)

- θ: Penalty for delayed drop-off
- $\pi_{s,c}$: pick-up time of route s if assigned to courier c. $\pi_{s,c} \geq \max_{o \in S} e_o$
- $\delta^s_{o,c}$: drop-off time of order o in route s if assigned to courier c.
- e_o: ready time of order o.
- T_c: the time where the courier c becomes available.
- C_t: set of available couriers at time t.
- S_t: set of routes at time t.
- N_s: number of orders to be delivered in route s
- $g_{c,s}$: cost associated with assigning courier c to route s.
- $$g_{c,s} = \frac{N_s}{\max_{o \in s}\{\delta^s_{o,c}\} - T_c} - \theta(\pi_{s,c} - \max_{o \in s}\{e_o\})$$

$$\max \sum_{c \in C_t} \sum_{s \in S_t} g_{c,s} x_{c,s}$$

$$s.t. \sum_{s \in S_t} x_{c,s} \leq 1, \forall c \in C_t$$

$$\sum_{c \in C_t \cup \{0\}} x_{c,s} \leq 1, \forall s \in S_t$$

$$x_{c,s} \in \{0, 1\}, \forall s \in S_t, c \in C_t \cup \{0\}$$

Figure 4- Integer Programming Model

b. Network Flow Model

This model is suggested by Rojas [1]. It consists of the following setting:

- Vertices:
  - Couriers
  - Routes
  - Source (denoted by S)
  - Sink (denoted by T)

- Arcs:
  - (c, s) between courier c and route s with capacity 1, cost g (c, s).
  - (S, c) between the source and courier c with capacity 1, zero cost.
  - (S, s) between the source and route s with capacity 1, zero cost.
  - (s, T) between the route s and the sink with capacity 1, zero cost.

- Problem:
  - Finding the max cost flow from S to T.

Solving this problem, we get the optimal courier-route assignment given corresponding costs. Flow values of the arcs (c, s) correspond to the courier-route assignments while (S, s) arcs for unassigned orders. Since all the couriers have an incoming arc with capacity one, they can only have outgoing flow equal to one. On the other hand, since the problem is a maximum flow problem, we don't need integer constraints, the relaxed version works fine.

6. Simulation Framework

a. Infrastructure

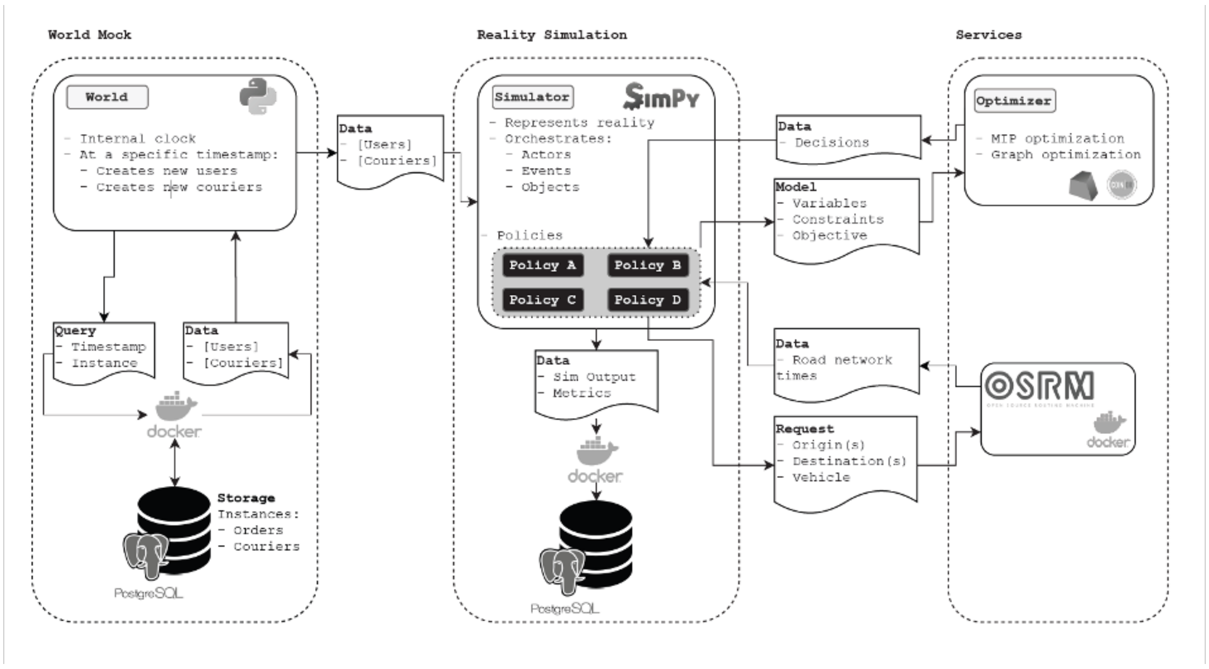


Figure 5- Structure of the simulation framework

Before diving deep into the simulation framework, it would be beneficial to mention about the services that are outside of the simulation but used by it. The first one is a PostgreSQL database that is running on a Docker container. It provides us with a storage area for inputs and outputs of the simulation. The simulation queries the two databases containing orders and couriers each second and creates users and couriers accordingly.

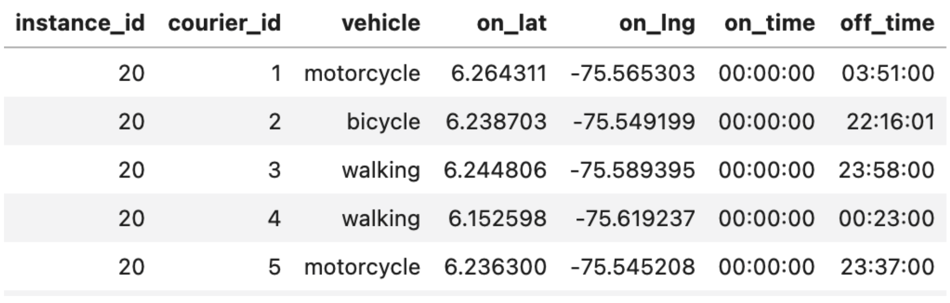
The second one is the OSRM [5] - Open Street Routing Machine which is an open-source tool that yields shortest paths between two locations (1). It utilizes OpenStreetMap and runs on a Docker container. For each movement decision of couriers, the simulation sends a GET request to OSRM API which listens to a port on the local machine. The response of the request contains stops of the suggested route, and haversine distances are calculated between stops and accumulated, which is then used to calculate the time that it takes for courier to travel that specific route.

The third service that we benefit from is the optimization tool. There are two options that are supported by the base version of the simulation: COIN-OR is the free one and GUROBI is the paid one. They are used to solve MIP or graph optimization models for each matching iteration.

The simulation framework that we use is written using Python library SimPy [6]. SimPy is a discrete-event simulation framework (2) that constitutes the base for our model. The code consists of actors and objects, which interact with each other through policies and other methods, and give rise to events. Simulation actors represent the real-life actors like courier, user and dispatcher. They are simply Python classes with the needed attributes and methods. Dispatcher is a singleton class in contrast to other two classes since there is only one decision maker in one subregion.

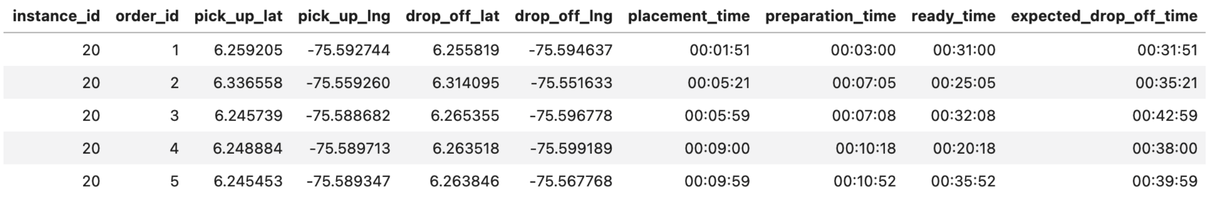
b. Input Types

The expected format of the simulation data input type can be seen in Figure 5 and Figure 6.



instance_id	courier_id	vehicle	on_lat	on_lng	on_time	off_time
20	1	motorcycle	6.264311	-75.565303	00:00:00	03:51:00
20	2	bicycle	6.238703	-75.549199	00:00:00	22:16:01
20	3	walking	6.244806	-75.589395	00:00:00	23:58:00
20	4	walking	6.152598	-75.619237	00:00:00	00:23:00
20	5	motorcycle	6.236300	-75.545208	00:00:00	23:37:00

Figure 6- Courier Instances



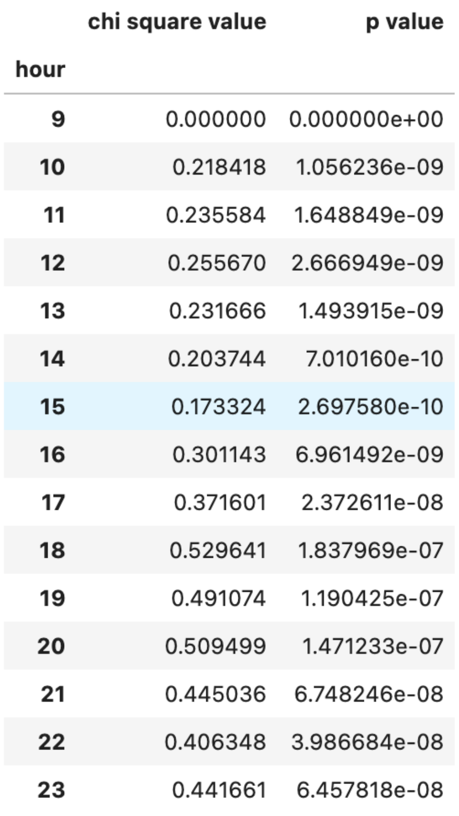
instance_id	order_id	pick_up_lat	pick_up_lng	drop_off_lat	drop_off_lng	placement_time	preparation_time	ready_time	expected_drop_off_time
20	1	6.259205	-75.592744	6.255819	-75.594637	00:01:51	00:03:00	00:31:00	00:31:51
20	2	6.336558	-75.559260	6.314095	-75.551633	00:05:21	00:07:05	00:25:05	00:35:21
20	3	6.245739	-75.588682	6.265355	-75.596778	00:05:59	00:07:08	00:32:08	00:42:59
20	4	6.248884	-75.589713	6.263518	-75.599189	00:09:00	00:10:18	00:20:18	00:38:00
20	5	6.245453	-75.589347	6.263846	-75.567768	00:09:59	00:10:52	00:35:52	00:39:59

Figure 7- Order Instances

In the data gathering part, Python scripts are written to transform Trendyol’s bulk data into simulation input data type.

For the courier instance generation, all vehicle types are assumed to be motorcycles as Trendyol requested. The starting locations (“on_lat”, “on_lng”) of the couriers are chosen at random restaurant locations since they tend to roam around the restaurants as Trendyol mentioned. Their shift hours (“on-time", ``off-time”) is determined according to the active courier number which is aggregated hourly in the following way: When the active courier number in the simulation is less than the courier number in historical data, couriers log on and when it is higher than the number in data they log off.

For the order instances, values in the columns are taken exactly from the historical data; however, the number of orders are generated according to the number of orders of the chosen day with a scale to relax computational burden. For order generation frequency, we conducted an analysis to see whether they are exponentially distributed according to the following chi-square test results and histograms (Figure 7 and 8):



	chi square value	p value
hour		
9	0.000000	0.000000e+00
10	0.218418	1.056236e-09
11	0.235584	1.648849e-09
12	0.255670	2.666949e-09
13	0.231666	1.493915e-09
14	0.203744	7.010160e-10
15	0.173324	2.697580e-10
16	0.301143	6.961492e-09
17	0.371601	2.372611e-08
18	0.529641	1.837969e-07
19	0.491074	1.190425e-07
20	0.509499	1.471233e-07
21	0.445036	6.748246e-08
22	0.406348	3.986684e-08
23	0.441661	6.457818e-08

Figure 8- Chi-Square test results

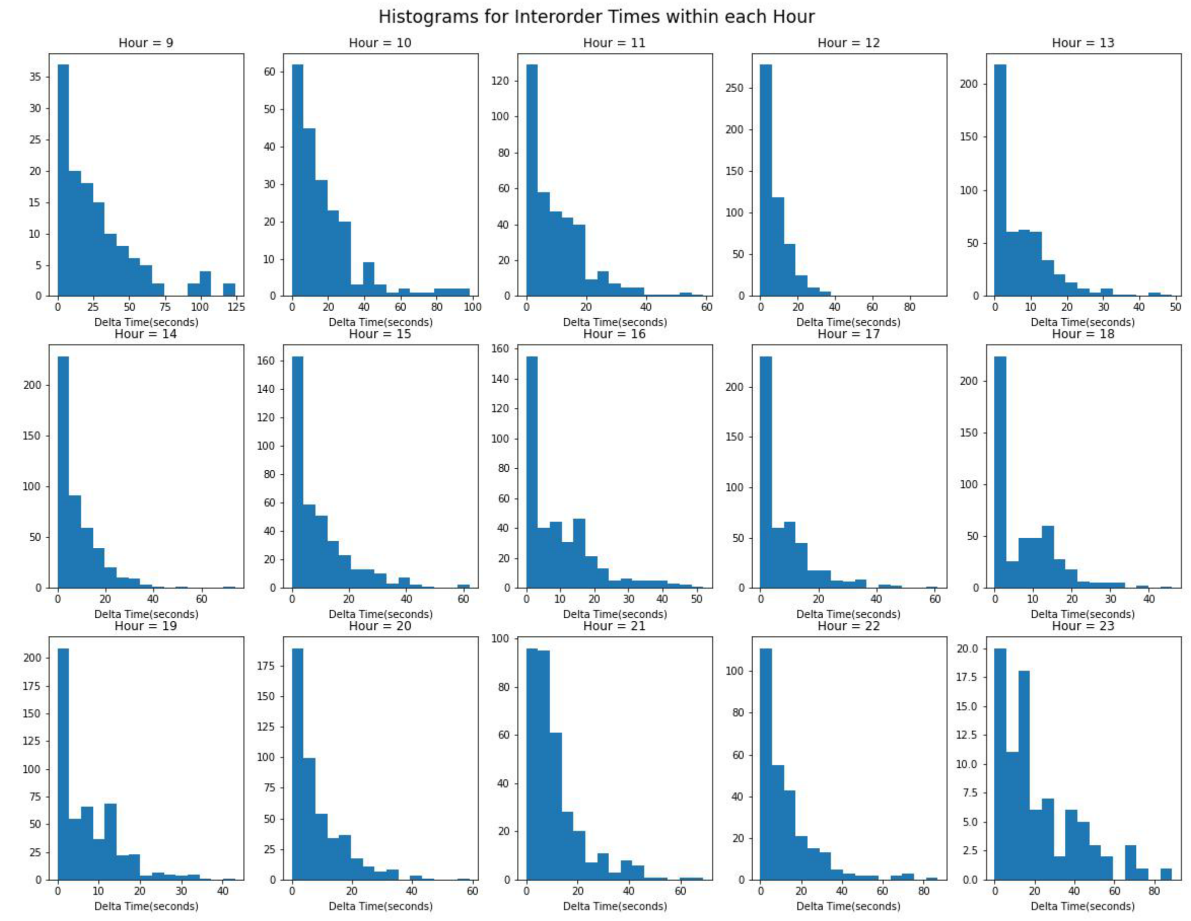


Figure 9- Histograms for interarrival times

c. Actors

i. User

The first and the second version of the basic user flowchart can be seen below. In the base version of the simulation, the user logs on to the system after being created and enters idle state automatically. Then, they submit their order and change their status to waiting state. The only remaining action the user can take is evaluating cancellation if the order is still not assigned to a courier after a specified time has elapsed. If it is the case, they either decide to keep waiting or cancel the order according to a cancellation probability. (Figure 9)

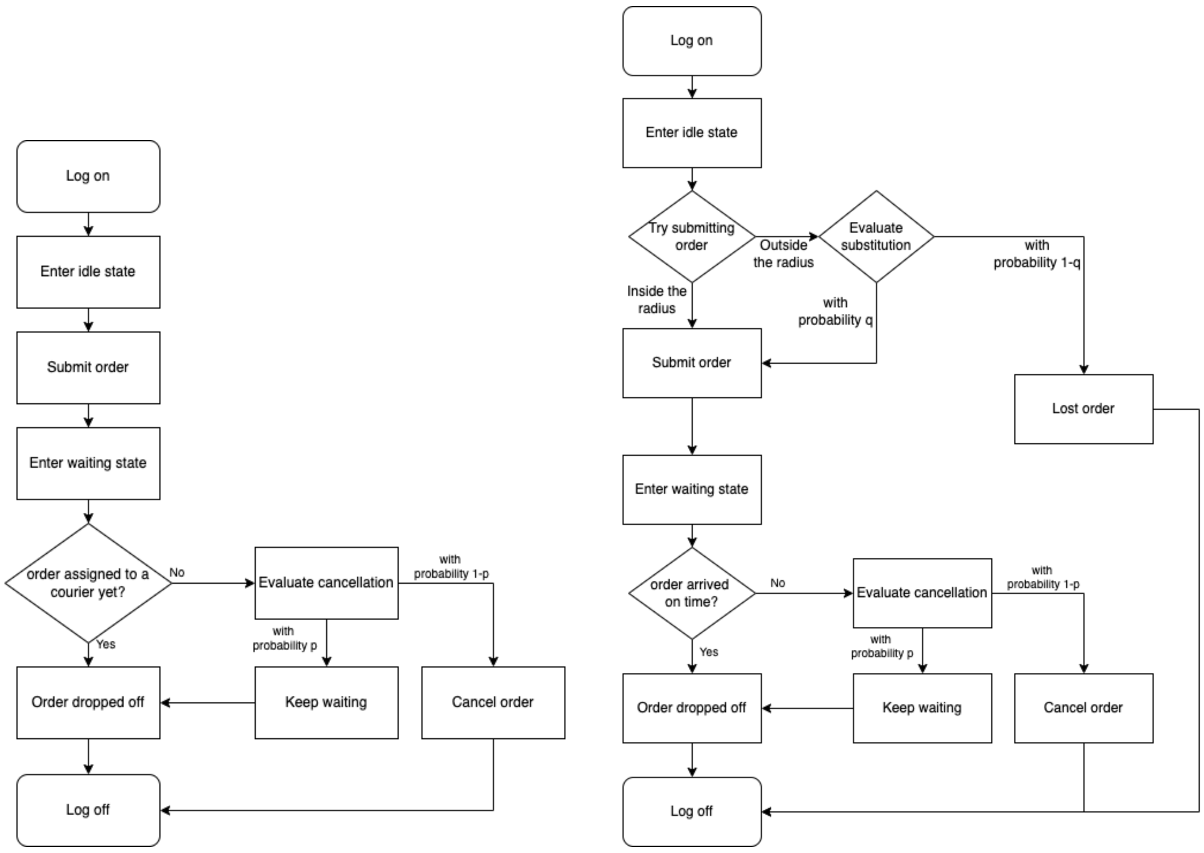


Figure 10- Old and New user flow chart

In the second version, we added another decision made by the user: substitution of the restaurant in the case of the intended restaurant is outside the radius due to demand management of the dispatcher. Here, the user first tries submitting the order from the intended restaurant and if rejected, substitutes to a closer restaurant with a substitution probability. (Figure 10)

ii. Courier

The courier is the actor that picks the orders up from restaurants and drops them off at user locations. They log on to the system at the start of the shift and enter the idle state. It is the state in which the courier waits for order notifications coming from the dispatcher. During this state, an order notification may be notified to the courier. If so, the courier evaluates the notification and may accept the notification with acceptance probability. Otherwise, they reject the notifications and stay in idle state.

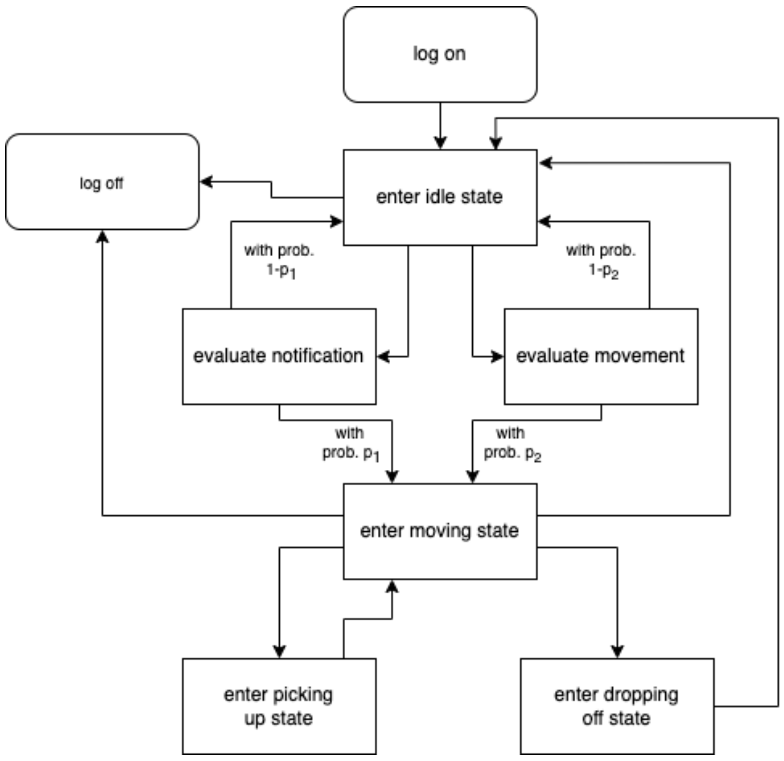


Figure 11- Courier flow chart

The courier may also decide to randomly move during idle state, which is a realistic design element considering courier behaviors when idle. During a delivery task, the courier enters picking up and dropping off states for a uniformly random time. They log off the system at the end of the shift. The only alteration to the courier class is the dynamic movement, which is not a decision but an affecting factor of courier's movement speeds during the day. We used historical data to implement the dynamic speeds. The graph obtained from historical data can be seen below.

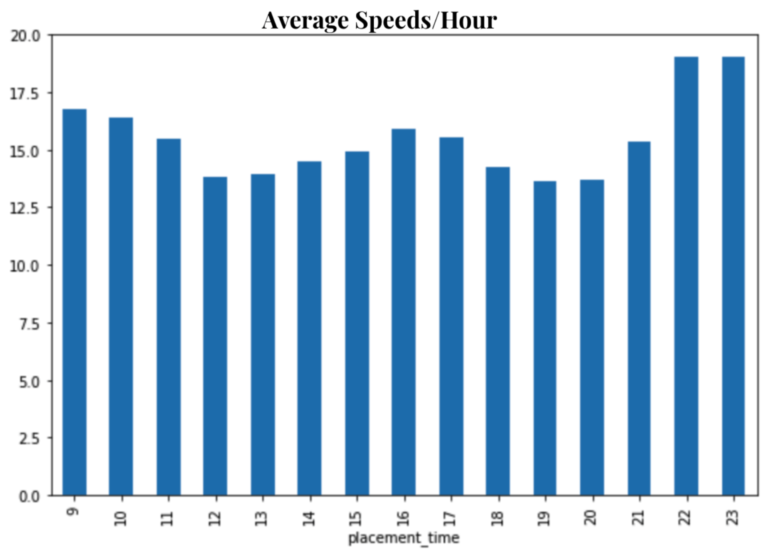


Figure 12- Average hourly courier speed

iii. Dispatcher

The dispatcher is the decision maker of the process, which matches the orders to couriers. During every simulation run, the dispatcher stays in the listening state and listens for courier states and locations, and new orders. The modification for this class is the evaluation of the demand management. For each arriving order, the dispatcher calculates the current congestion as the ratio of the number of unassigned orders to the number of idle couriers. If it is above a certain threshold, permitted order radius is narrowed for users. Order trials outside the permitted radius are notified to the user so that the users can evaluate substitution.

Other than that, the dispatcher tries to match unassigned orders to idle couriers by solving matching heuristics at the end of each time window. The length of the time window can also be changed, which can affect the performance of the dispatcher and customer satisfaction. We could not try different time window lengths due to time constraints.

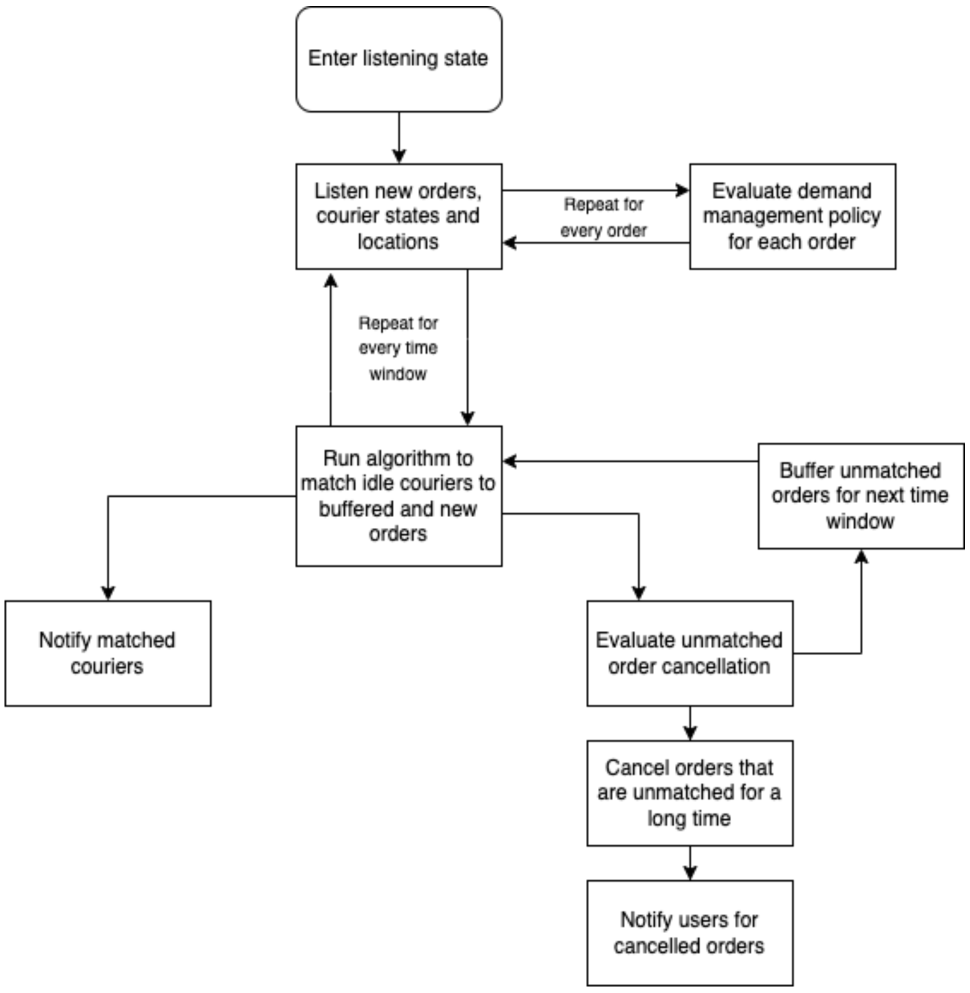

Enter listening state
Listen new orders, courier states and locations
Repeat for every order
Evaluate demand management policy for each order
Repeat for every time window
Run algorithm to match idle couriers to buffered and new orders
Buffer unmatched orders for next time window
Notify matched couriers
Evaluate unmatched order cancellation
Cancel orders that are unmatched for a long time
Notify users for cancelled orders

Figure 13- Dispatcher flow chart

d. Simulation Configurations

There is a Settings class in the simulation in order to pass the configurations to the model easily. This class contains a dictionary consisting of keys and values for each parameter. One can specify the instance ids to run, seed to use for replications, matching algorithms, the start time and the end time of the simulation, users and courier creation intervals, warm up time, actor policies, demand management policies and matching algorithm parameters.
For our simulation runs, we created a grid for congestion threshold and radius limit values, and the matching algorithm to use.

- Congestion thresholds: 0.7, 0.8
- Radius limits: 2.5 km, 3 km
- Matching algorithms: MDRP, MDRP-graph

In addition, we created instances for 3 days:

- 12 May: A casual weekday
- 14 May: A casual weekend
- 18 May: A congested day with insufficient courier numbers

The outputs of runs with these configurations in a specific subzone of Trendyol Go can be seen in 8. section below.

7. Requirements and Limitations

First of all, a simplified open-source simulation framework written in object-oriented manner was a need to implement the necessary policy improvements and other developments in the simulation. For this need, as mentioned in the literature review, the simulation that Rojas [1] developed is chosen since it has many capabilities in terms of adding new policies and it is closer to the real-life examples such as Trendyol.

Another requirement was the data to test the performance of the system, validate the improvements in the simulation and tune the parameters of the simulation. However, Trendyol provided the full data for only one subzone (Besiktas, Istanbul) of its service network for just one day. Then they also provided three days' aggregate data where it enabled us to generate order and courier instances according to that data. Thus, having only three days and one subzone data was not enough to test whether the system improvements are robust enough to work in a more generalized case.

Lastly, the data of three days is given later in the project phase where the time was limited, and the computation power was not enough to handle to test so many alternatives. Therefore, the simulation grid was chosen small to ensure the validation of the improvements.

8. Results and Evaluation

Before moving on to the coefficient analysis of simulation, we can look for overall statistics (Figure 13). Firstly, between two algorithms' performances on tardy jobs ratio, we see that MDRP-graph performs slightly better than MDRP. During weekday orders, decreasing radius from 3 to 2.5, or decreasing threshold from 0.8 to 0.7 does not seem to affect the overall performances. Nonetheless, when it is a busy day like 18 May with a courier bottleneck, it seems reasonable to decrease radius since tardy job ratio considerably decreases. To conclude, in a busy day, most important factor seems to be limiting radius and tuning density threshold may result in even better performances.

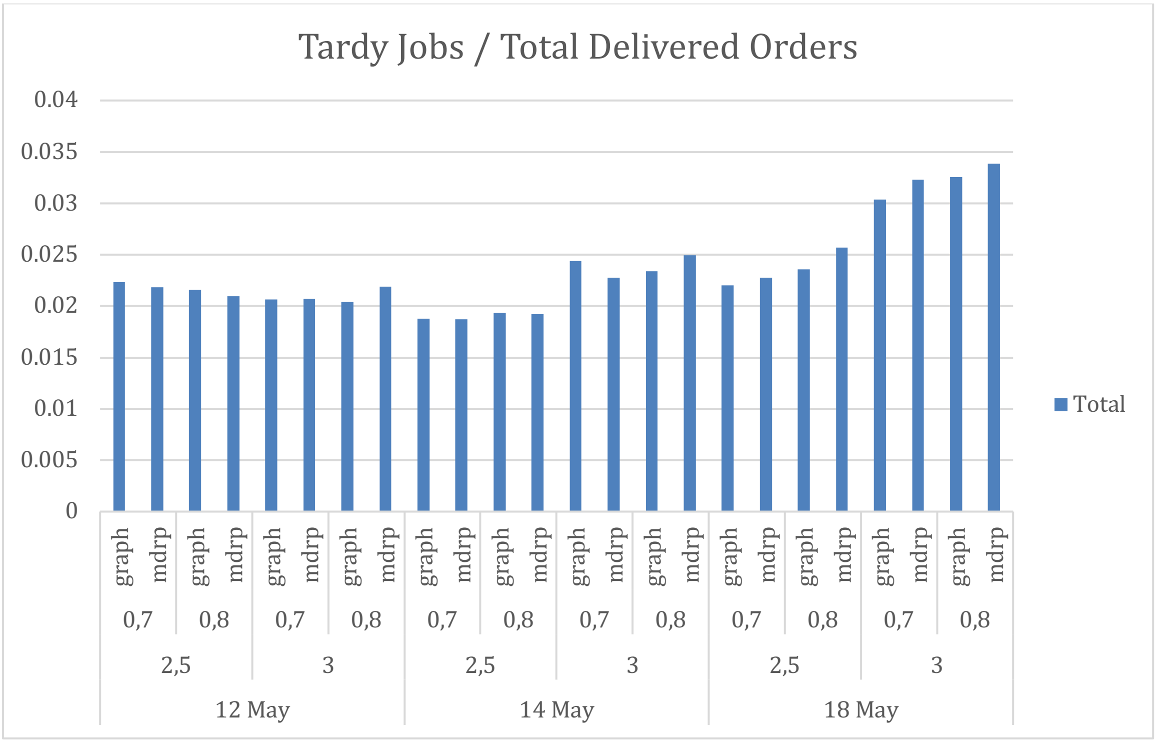
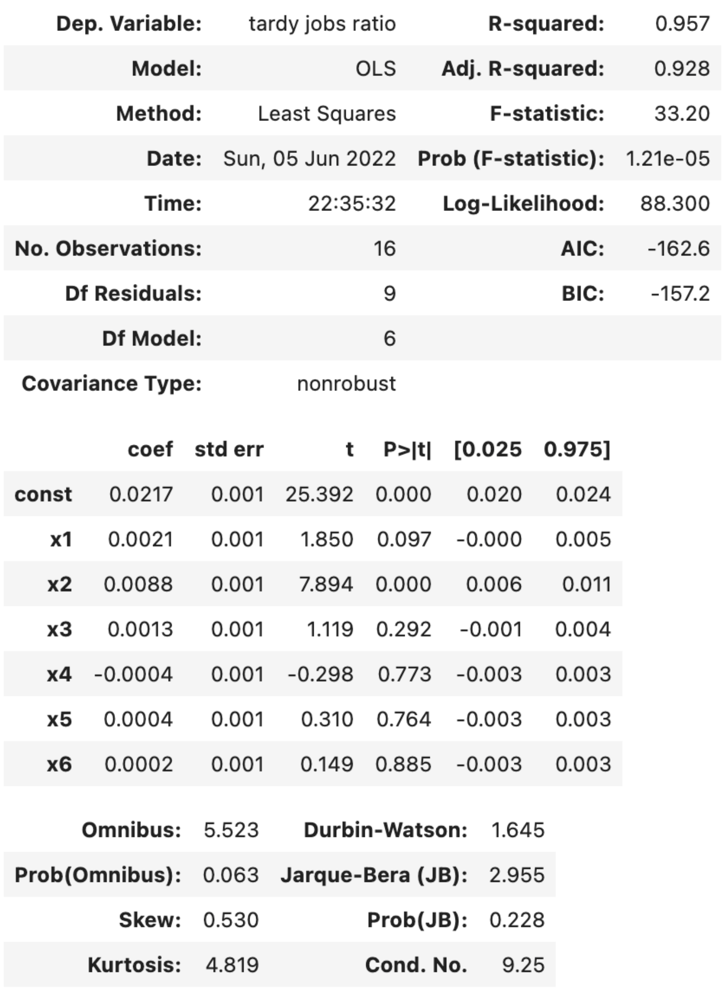


Figure 14- Tardy jobs ratio

To support our claims, we performed coefficient analysis for each day. The following summary table of linear regression model shows that x2 variable (‘limiting radius 3’) significantly increases tardy jobs ratio compared to base case which is limiting radius 2.5 with p-value smaller than 0.001 (Figure 14)



Dep. Variable:	tardy jobs ratio	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.928
Method:	Least Squares	**F-statistic:**	33.20
Date:	Sun, 05 Jun 2022	**Prob (F-statistic):**	1.21e-05
Time:	22:35:32	**Log-Likelihood:**	88.300
No. Observations:	16	**AIC:**	-162.6
Df Residuals:	9	**BIC:**	-157.2
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	0.0217	0.001	25.392	0.000	0.020	0.024
x1	0.0021	0.001	1.850	0.097	-0.000	0.005
x2	0.0088	0.001	7.894	0.000	0.006	0.011
x3	0.0013	0.001	1.119	0.292	-0.001	0.004
x4	-0.0004	0.001	-0.298	0.773	-0.003	0.003
x5	0.0004	0.001	0.310	0.764	-0.003	0.003
x6	0.0002	0.001	0.149	0.885	-0.003	0.003

Omnibus:	5.523	**Durbin-Watson:**	1.645
Prob(Omnibus):	0.063	**Jarque-Bera (JB):**	2.955
Skew:	0.530	**Prob(JB):**	0.228
Kurtosis:	4.819	**Cond. No.**	9.25

Figure 15- Coefficient analysis for May 18

It should also be considered that while increasing customer satisfaction metric (decreasing cancelled orders), we are losing potential sales by shifting cancelled orders to lost orders as can be seen in Figure 15. Cost of cancelled and lost orders can be determined, and decisions can be made accordingly.

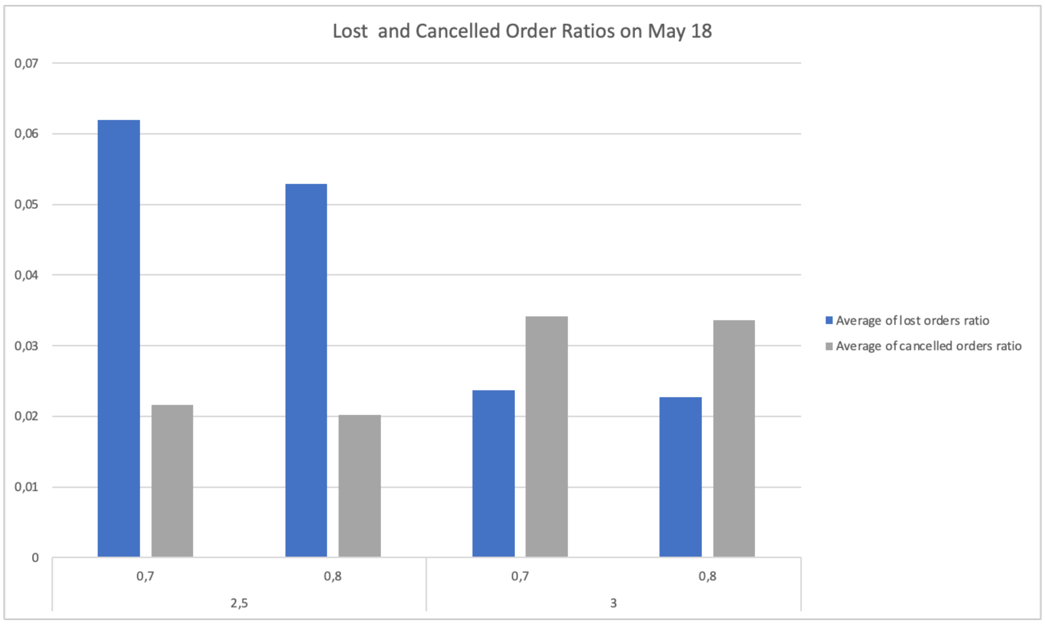


Figure 16- Lost and cancelled orders ratios on May 18

9. Further Improvements

Firstly, as mentioned in the limitations section, more data and computation power can be incorporated so that the simulation results are more meaningful, and different scenarios can be implemented more reliably.

Secondly, assumptions made during the formulation of the problem can be replaced with more generalized ones. By generalized assumptions, we mean the assumption of a fixed set of couriers and courier related parameters and a fixed set of restaurants. On the courier side, courier attributes such as vehicle type and different shift hours may be added to the grid of the parameters to be tuned. In the restaurant side, a restaurant class can be written so that the meal preparation times, and demand management module can be specialized for each restaurant based on how busy they are.

Lastly, from the problem's side, different heuristics can be added so that we have more options to choose from. In addition, recall the Reyes [2] emphasizes that how often the matchings are done can also change the overall performance. Therefore, running simulation with different time windows at which the MDRP is solved can also be done. Further, this time window does not need to be a static variable, instead it can be formulated as a function of the demand level so that the system is not put into so much load when the demand level is low.

10. Conclusion

In this project, we made use of IE tools such as optimization models, distribution fitting, hypothesis testing, statistical inference, discrete event simulation, object-oriented programming to create value for meal delivery sector.

To summarize our findings, it is seen that proposed simulation framework is able to simulate and test real-life scenarios such as Trendyol's case and report necessary metrics without dedicated time and financial costs of A/B testing. It is also seen that with the added demand management module, such e-commerce companies functioning in on-demand meal and grocery delivery can evaluate and tune their parameters. These parameters were density threshold and limiting radius in this study, however the simulation has potential to serve more purposes.

Additionally, we show that decreasing limiting radius more is the most proper way of dealing with courier bottlenecks.

11. References

[1] S. Q. Rojas, "Computational framework for solving the meal delivery routing problem.," 2020.

[2] A. Reyes, "The meal delivery routing problem," *Optimization Online,* vol. 6571, 2018.

[3] B. Y. a. M. Savelsbergh, "Provably high-quality solutions for the meal delivery routing problem," *Transportation Science,* vol. 53.5, pp. 1372-1388., 2019.

[4] P. S. a. G. Doreswamy, "Simulation and optimization framework for on-demand grocery delivery," *Winter Simulation Conference (WSC). IEEE,* 2021.

[5] "Open-source Routing Machine," [Online]. Available: http://project-osrm.org/.

[6] "Discrete event simulation for Python," [Online]. Available: https://simpy.readthedocs.io/en/latest/.

[7] "Worldwide Retail E-commerce Sales," Statista, [Online]. Available: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/ .

[8] T. T. Bakanlığı, "Yıllık E-ticaret Bülteni," [Online]. Available: https://www.eticaret.gov.tr/dnnqthgzvawtdxraybsaacxtymawm/content/FileManager/Dosyalar/2021%20Y%C4%B1l%C4%B1%20E-Ticaret%20B%C3%BClteni.pdf .