Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

SIMULATION AND OPTIMIZATION FRAMEWORK FOR ON-DEMAND GROCERY DELIVERY

Siddhartha Paul
Goda Doreswamy

Data Science, Swiggy, Bundl Technologies
Bangalore, KA 560103, INDIA

ABSTRACT

This paper presents a generic two-stage optimization model and a simulation framework for on-demand grocery delivery. The proposed simulation framework can be used to reproduce any historical day's behaviour or evaluating various what-if scenarios. The two-stage optimization model's objective is to minimize the Cost Per Delivery (CPD) and maximize Customer Experience (CX). In the first stage, Last-Mile (LM) delivery optimization is modelled as a Pickup and Delivery Problem with Time Windows (PDPTW). Both static and dynamic variants of the PDPTW model are tried out and compared to their performances. A Just-In-Time (JIT) heuristic is integrated with PDPTW to minimize waiting time. The second stage solves the First-Mile (FM) delivery optimization using a multi-objective assignment model for trading off between the CPD and CX. The proposed framework is simulated with actual order data and it was found that the the dynamic PDPTW model results in significant savings in CPD while maintaining a good CX.

1 INTRODUCTION

The e-grocery industry is booming due to the current pandemic situation. Customers prefer to order online their daily needs, including groceries, to avoid unnecessary social interactions. Consequently, in metro cities, customers are getting used to this unparalleled convenience of online delivery. The trend is expected to remain the same even after the COVID pandemic. Currently, the e-grocery market value is estimated at $2.9 billion in India in 2020. The annual growth rate is predicted to be over 30 %. Many companies are in the race to capture market share and unlock the growth potential. On-demand grocery delivery is one of the competitive business services to provide a great customer experience. This paper focuses only on the on-demand grocery delivery, where customer demands need to be satisfied within a specific time (aka promised Service Level Agreement or SLA) from the ordered time. On-demand service's operational challenges are much higher than any scheduled delivery services due to the short decision-making period. It is very important for the delivery system to be efficient. This efficiency can be directly translated into business profits or customer satisfaction. The challenge is to provide timely delivery of orders with minimal Cost Per Delivery (CPD). The timely delivery is defined as the order is delivered within SLA. The Customer Experience (CX) is a function of delay.

The CPD comprises travel pay (i.e., FM and LM pay) and wait-time pay. The waiting time is defined as the time spent by a Delivery Executive (DE) at the store waiting for the order to be prepared. This problem maps to a Multi-Depot Pickup Delivery Problem with Time Windows (MDPDPTW) (Ramos et al. 2020). In MDPDPTW, each DE location can be treated as a depot, stores as the pickup locations, and customers as the delivery locations. However, the large scale of the problem and the necessity of a near-real-time solution make it infeasible to solve through MDPDPTW. Moreover, the MDPDPTW is not very helpful in minimizing the waiting time at a store. For example, if some orders have longer preparation time, because they have too many items to be packed, they could be assigned later. Consequently, DE will not be spending significant waiting time at the store for the order to be ready. The classic MDPDPTW with constant drop

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

cost does not help in postponing long preparation time orders. Ideally, during peak hours (i.e., when the system has fewer DEs as compared to the number of orders), longer preparation time orders should be dropped in order to assign urgent orders and maintain the SLA. In this paper, we propose an optimization framework to handle these challenges of the e-grocery delivery business.

The e-grocery delivery is quite similar to on-demand food delivery. There is limited literature on the on-demand meal delivery routing problem. Reyes et al. (2018) has formally defined the Meal Delivery Routing Problem (MDRP) and relevant crucial business metrics. The authors have proposed an optimization model for overcoming the scale challenge and real-time solution. In this paper, a system-wide standard batch size (i.e., bundles of orders) is decided based on the DE availability and active orders. Then, orders are batched using a parallel-insertion algorithm. The assignment of the batches is modelled using an Integer Linear Programming (ILP) problem. However, a system-wide standard batch size may not be appropriate and should depend on the type of orders. For instance, during peak hours, there will be a low number of DEs compared to the number of orders and this algorithm will set large batch size to compensate for that. Consequently, the Order to Delivery (O2D) time will be significant and food will lose freshness. Ulmer et al. (2021) have studied the problem of restaurant meal delivery under uncertainty in order-ready times. Authors have formulated the problem as a Markov Decision Process to postpone orders having large order-to-prepared time. This paper has provided a good literature review on the MDRP problems and compared their objectives, scalability, postponement etc. Liao et al. (2020) have modelled the MDRP problem as a multi-objective ILP. The goals are to maximize CX as well as DE balance utilization and minimize carbon footprint. DE balance utilization ensures that all DEs get a similar number of orders (i.e., promoting equal earning). The problem is solved in a two-stage approach. In the first stage, an initial number of drivers are found via the Non-dominated Sorting Genetic Algorithm II (NSGA-II). Also, the initial routes are formed via PCA and the k-means algorithm. In the second stage, the initial solution is improved via the Adaptive Large Neighborhood Search (ALNS) algorithm. This approach's performance is compared with Simulated Annealing (SA) and Tabu Search (TS) algorithms. However, this approach is demonstrated with only 100 orders and may not provide real-time solutions for large-scale problems. Kottakki et al. (2020) have presented a multi-objective assignment problem for the First-Mile (FM) delivery optimization. In this paper, the CPD and the CX trade-off are modelled as a weighted combination. Moreover, a penalty function is used for trading off the cost of assigning and not assigning a batch. Authors have implemented the proposed model in a food-tech company and compared their results against a base algorithm with actual data. Paul et al. (2020) have addressed both the FM and Last-Mile (LM) delivery optimization problem for on-demand food delivery. The proposed solution is experimented with real data to provide a real-time solution for large scale of orders. Moreover, this paper proposed an algorithm to perform the batching and assignment in an integrated manner by providing all the feasible candidate batches to the assignment problem. The assignment problem is modelled as a multi-objective optimization problem. Further, the Just-In-Time (JIT) concept is introduced in this paper to minimize the food waiting time. Finally, this paper concluded that JIT and integrated batching assignment results in significant savings in CPD while maintaining CX. Unlike food, Alonso-Mora et al. (2017) have studied the batching and assignment problem for on-demand high capacity ride sharing. This paper has proposed heuristics and an exact approach for solving the ILP to minimize travel delay and re-balancing idle vehicles. Despite having a lot of similarity with the on-demand ride-sharing, the on-demand food delivery problem is relatively complex because of the additional requirement of JIT. In e-grocery delivery optimization literature, most of the work is focused on scheduled delivery (Pan et al. 2017; Leyerer et al. 2020; Liu et al. 2020; Vazquez-Noguerol et al. 2020). Therefore, these problems are more flexible in terms of solution time.

The on-demand e-grocery delivery problems are relatively less complex than food delivery in terms of scale. We can split a city into multiple zones containing multiple stores, given the e-grocery delivery is happening at a store level. Thus, the problem can be broken down into a smaller scale and solved without much interaction. The smaller problem scale motivates trying out Vehicle Routing Problem (VRP) variants to obtain higher-order optimal batches. In this paper, we have split the e-grocery delivery problem

into two stages. The LM optimization is performed in Stage 1 via a Pickup Delivery Problem with Time Windows (PDPTW). In Stage 2, the FM optimization is modelled as a multi-objective optimization problem to minimize the DE travel distance, waiting time at the restaurant, and delivery within SLA. A Discrete Event Simulation (DES) framework is described to compare the performance of the proposed algorithms and a few policies for e-grocery delivery. The objectives of this paper are: (i) to develop a simulation (DES) framework for simulating the e-grocery delivery system, (ii) to develop a novel two-stage optimization method for solving the delivery cost optimization problem, (iii) to compare two variants of the vehicle routing problem for solving the LM delivery optimization, (iv) to build a JIT heuristic for waiting time minimization, and (v) to compare the algorithms in proposed simulation framework with real order data.

2 MODEL

This section starts by introducing the e-grocery delivery optimization problem with suitable examples. In the later part of this section, the DES, LM optimization, and FM optimization frameworks are discussed.

The overall e-grocery delivery optimization problem is solved on a rolling horizon basis. In any given decision cycle (aka cron), the system has a set of customer orders (i.e., $O = \{O_1, O_2, O_3, \cdots, O_n\}$), a corresponding set of stores (i.e., $S = \{S_1, S_2, S_3, \cdots, S_m\}$), and a set of available DEs (i.e., $DE = \{DE_1, DE_2, DE_3, \cdots, DE_k\}$). The customer locations corresponding to order set O are denoted by C (i.e., $C = \{C_1, C_2, C_3, \cdots, C_n\}$). In every cron, the system needs to decide on the routing of the orders, DE assignment, dropping the high-waiting-time orders that could be assigned in a later decision cycle.

Figure 1 shows the process view of the system. In this instance, four customer orders are placed from three different stores (C1 and C2 from S1, C3 from S2, and C4 from S3) and four DEs are available for assignment. The LM optimization model returned two batches ($B_1 = \{O_1, O_2\}$, $B_2 = \{O_3, O_4\}$) with the sequence: (i) $S_1 \longrightarrow C_1 \longrightarrow C_2$ and (ii) $S_2 \longrightarrow S_3 \longrightarrow C_3 \longrightarrow C_4$. Later, in the FM optimization stage, DE_1 gets assigned to B_1 and DE_3 gets assigned to B_2. The dotted line indicates the distance to be travelled in the absence of batching. Batching more orders saves the LM distance travelled per order and hence CPD.

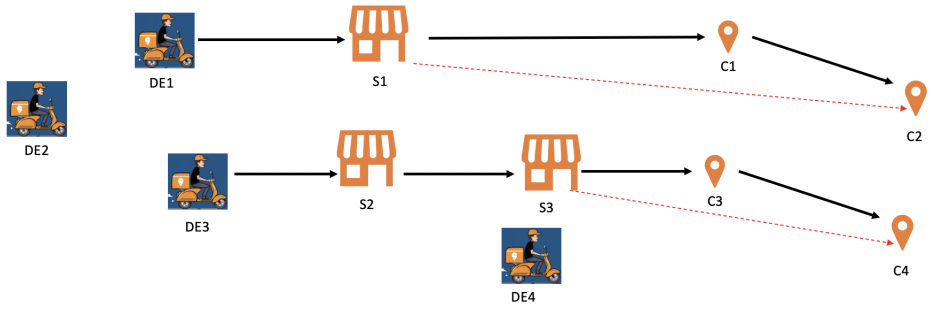


Figure 1: Process view of on-demand grocery delivery.

Figure 2 shows the key events in a decision cycle (i.e., cron). In the first step, the system considers all new orders and the orders not yet picked up. Once batches are formed using the LM optimization algorithm, the batches participate for assignment to the DEs (i.e., FM optimization). A couple of past orders are completed in the current cron, and a few DEs become available for new orders.

2.1 Simulation Framework

The grocery delivery system is modelled using Discrete Event Simulation (DES) methodology and implemented via *SymPy* (Matloff 2008) in *Python*. Figure 3 shows the conceptual diagram of the proposed simulation framework. Figure 4 shows the timeline view of the grocery delivery system. Once a customer order is received, the store confirms the order. Upon confirmation, immediately, the order starts getting packed. Time prediction models are called for Order to Packed (O2P) time. The goal is to make a

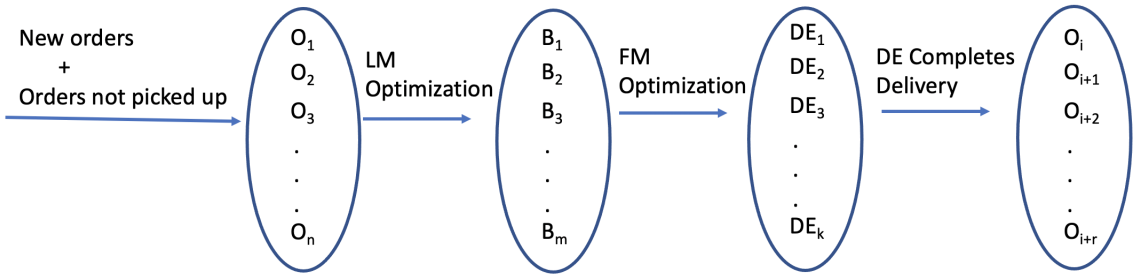


Figure 2: Decision cycle view,

Just-In-Time (JIT) assignment based on this predicted O2P so that DE does not wait at the store. The system runs a batching algorithm (i.e., for LM optimization) and an assignment algorithm (i.e., for FM optimization) in the back end to decide when the order should be assigned (Order to Assigned Time or O2A), which DE to assign, and what route sequence to follow. Once the order is assigned to a DE, the DE confirms the order and starts travelling towards the store.

The DE reaches the store after FM time (calculated based on the Haversine distance between the DE location, store location, and DE speed). Once DE goes to the store, the order is picked up after a service time if the order is ready. If O2P is underestimated, the DE spends additional waiting time at the store. After order pickup, the DE delivers the order to the customers in the pre-decided routing sequence. The critical events in this simulation are: new order arrival, store confirmation, DE assignment, DE confirmation, DE arrival at store, order pickup, DE reached customer location, and order delivered. The key states of the system are: DE is free, DE is busy, order is active, order is getting packed, order is marked ready, order is completed.

This simulation framework is helpful to reproduce, compare with historical data, and answer various *what-if* questions.

2.2 Optimization Framework

In this section, the overall optimization (i.e., MDPDPTW problem) is split into two stages, viz. LM delivery optimization and FM delivery optimization, to handle the scale, complexity, waiting time issue, and the trade-off between CPD and CX. The LM delivery optimization is achieved through solving a Pickup Delivery Problem with Time Windows (PDPTW) that minimizes the CPD. This optimization returns a set of routes (aka batches) of orders. The FM delivery optimization problem is handled through a multi-objective assignment problem. The FM delivery optimization tries to assign nearby DEs to a batch that minimizes the FM distance and waiting time at the store. Further, FM optimization ensures timely delivery of the order to maximize CX.

2.2.1 LM Delivery Optimization

Two variants of VRP for LM optimization viz. static PDPTW (aka batching till assignment) and Dynamic PDPTW (DPDPTW) are tried out in this paper. The DPDPTW problem is performed till DE arrival at the store (i.e., batching till arrival). The static PDPTW considers all unassigned orders (i.e., the fresh orders received in current cron and previously unassigned orders) and perform routing of these orders. It is noted that the more orders we receive, the more feasible route combinations are created. Hence, the PDPTW problem can find a more optimal solution with a higher-order volume. The DPDPTW (aka batching till arriving) problem is motivated based on this intuition. In the DPDPTW problem, the routing of already assigned orders is explored with the new incoming orders by keeping a partial sequence of the route unchanged. The reason for locking a partial sequence (say, $S_2 \longrightarrow S_3$) of the route ($S_2 \longrightarrow S_3 \longrightarrow C_3 \longrightarrow C_4$) is to resolve any DE communication problem. Once a DE is assigned, the DE starts travelling towards the

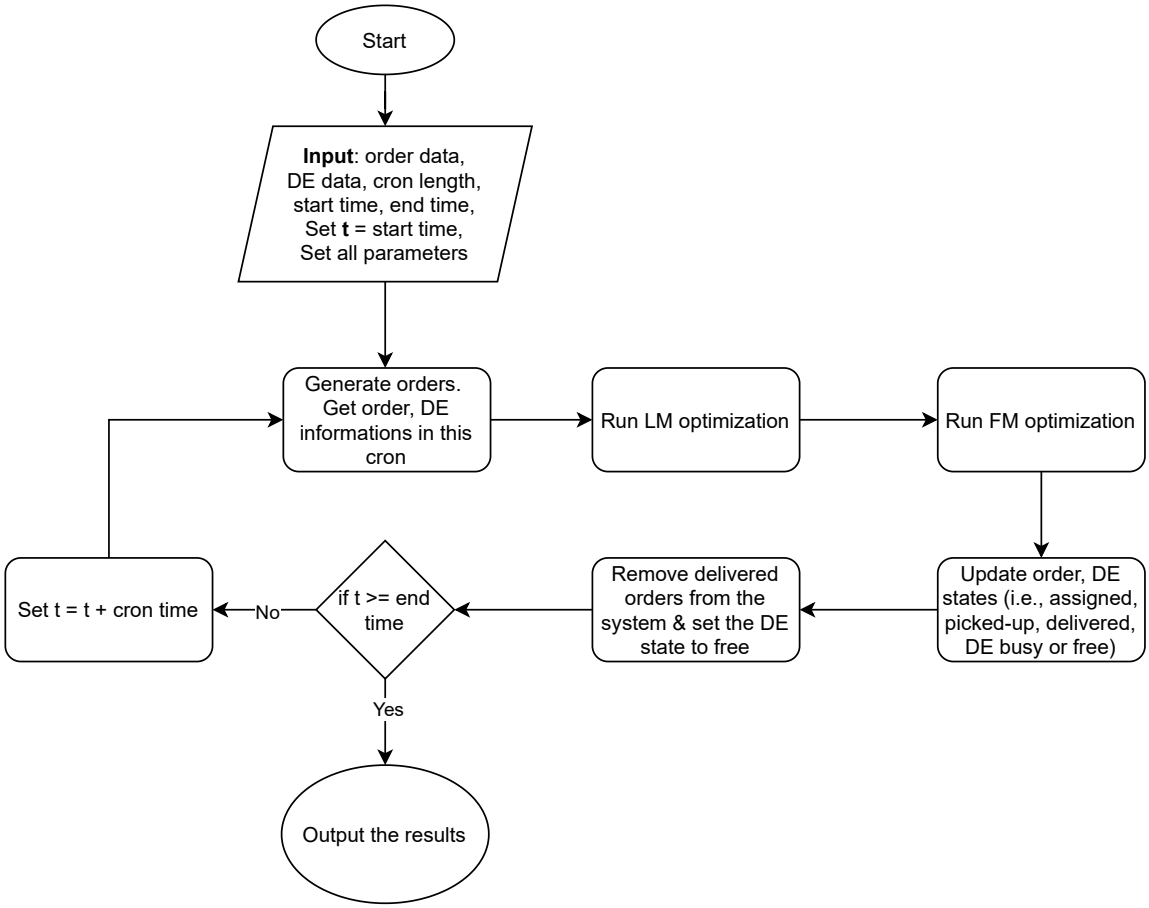


Figure 3: Conceptual diagram of simulation framework of on-demand grocery delivery.

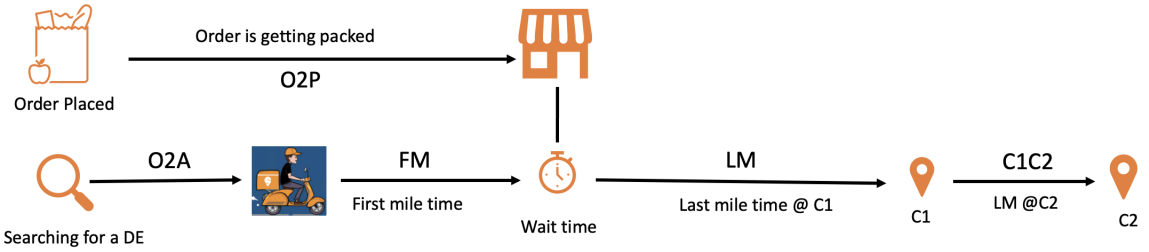


Figure 4: Timeline view of on-demand grocery delivery.

store, so the next pickup task (or set of tasks within next t minutes) of the route is locked and remaining tasks are free to get reshuffled. This node locking can be done using the *applylocks* method on *Google OR-Tools*.

The LM delivery optimization problem is solved and compared using two approaches: (i) PDPTW (when we solve the problem batching till assignment) and (ii) DPDPTW (when we solve the problem batching till arrived and initialize with the previous solution). This problem is implemented in *Python* using *Google OR-Tools* routing optimizer (https://developers.google.com/optimization).

We are unsure about the DE location and fleet capacity for the time slot in the LM optimization stage. Figure 5 shows a virtual depot (say 0) is assumed where all DEs are located. LM optimization has a set of pickup (stores) and drop (customer) locations (see directed arcs). We assume the distance

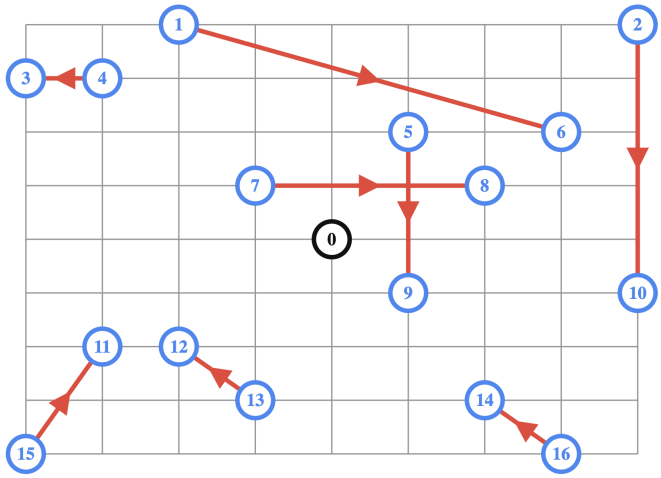


Figure 5: Conceptual diagram for PDPTW (Source: https://developers.google.com/optimization/routing/pickup_delivery).

from the virtual depot to any pickup location as a constant parameter viz. *DE independent FM* and from any drop location to the depot is 0. The travel time from the virtual depot to any pickup location is based on the *DE independent FM* distance, DE speed, and estimated assignment time. The decision variable is

$$x_{i,j,k} = \begin{cases} 1, & \text{if DE k visits location j immediately after location i} \\ 0, & \text{otherwise.} \end{cases}$$

For the objective function, the classic VRP model minimizes the total distance travelled or a weighted combination of the total distance travelled and the maximum length of a route (ensuring every vehicle will travel a similar distance). As the payout scheme for grocery stores is not a linear function of the distance or the time, the node-to-node distance matrix is modelled as a replica of the CPD. Further, a longer distance (as payout schemes compensate for the same) travel is preferable over the use of more drivers. With O2P prediction models, most of the orders will be ready to pick up at stores. Hence, a DE can pick up multiple orders from a store by spending service time on just one order. This gives significant savings in waiting time and hence in CPD. However, at the drop locations or different pickup locations, the DE has to spend a specific service time. We denote the total number of DEs by *"K"* and the total number of nodes (pickup, drop, and depot) by *"N"*. Table 1 shows the edge cost function based on the type of nodes connected to the edge.

Table 1: Node to node cost matrix.

From Node	To Node	VRPCost
Depot	Pickup	Constant FM Cost + Store Service time cost
Pickup1	Pickup 1	0
Pickup 1	Pickup 2	Travel Distance Cost+ Store Service time cost
Pickup	Drop	Travel Distance Cost + Customer touchpoint cost
Drop	Drop	Travel Distance Cost + Customer touchpoint cost
Drop	Pickup	Large Cost (10000)
Drop	Depot	0

$$\min \ Z = \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} VRPCost_{i,j,k} \times x_{i,j,k} \tag{1}$$

Equation (1) shows the optimization objective as minimizing of the total route cost in a given cron.

The following practical business constraint and Time Windows (TW) constraints are used in the PDPTW model:

1. *Weight Constraint:* DE can carry a maximum W kg weight.
2. *Item Constraint:* DE can have a maximum of It items.
3. *Capacity Constraint:* DE should not carry more than k orders at any point in time.
4. Constant *service time* per order spent at every customer and per batch at every store for pick up.
5. *Item Group Constraint:* Orders containing items from different batching groups should not be batched. This is done by setting the node-to-node cost from different groups to a very large number (10,000) to make batching among them infeasible.
6. *Last In First Out (LIFO) Constraint:* Few orders contain items that can be batched only when picked last and delivered first. This can be handled by making all distances from the pickup node of those orders very large so that it is directly routed to the drop node first.
7. Any rejected order from PDPTW should be activated immediately for assignment and removed from future batching crons.
8. *Time Window Constraints:* See Table 2.

We define,

- ready = 1 if order is marked ready, else 0;
- assigned = 1 if the order is assigned, else 0;
- lock = 1 if the order pickup node is locked, else 0.
- α: A factor to approximate OpenStreetMap (OSM) distance by Haversine distance.

$$\text{TTRS} = \frac{\alpha \times \text{Haversine distance (DE's current location to the store)}}{\text{DE Speed}} \tag{2}$$

with TTRS being the expected time to reach a store from the current DE location.

It is noted that "*revised SLA*" is the predicted order-delivered-time based on the recent routing sequence. "*Safety Buffer (SB)*" is the parameter intended to increase compliance by targeting a lower SLA than promised.

Table 2: Time windows.

Time Windows	**On-Demand static/dynamic PDPTW**
Earliest Pickup Time (EPT)	lock× (confirmed time+ expected time to reach store from current DE location+ service time) +(1-lock)× max{(1-ready)× (ordered time+O2P)+current_time× ready, current_time}
Earliest Drop Time (EDT)	EPT +LM+service time
Latest Drop Time (LDT)	lock× max{max{ordered time + SLA - SB, revised SLA}, EDT+1} +(1-lock)× max(ordered time + SLA - SB, EDT + 1)
Latest Pickup Time (LPT)	max(LDT - LM -service time, EPT + 1)

It should be noted that any infeasibility in time windows would result in the rejection of that order in the PDPTW module and activated as a single order batch (see Constraint 7). However, any infeasibility of TW constraints in the initial solution of the DPDPTW would make the whole problem infeasible. Consequently, the optimization will fail. An exception handler algorithm is written to remove or update the infeasible orders from the initial route to make the problem feasible. The primary reason for the TW constraints' infeasibility are prediction errors and disruptions (e.g., unexpected delay in assignment due to DE unavailability, prediction error in O2P, etc.).

For the solution algorithm, the popular *savings algorithm* (Clarke and Wright 1964) is used as a construction heuristic to find the initial solution. Afterwards, a meta-heuristic search algorithm (*guided_local_search*) (Voudouris and Tsang 1999) is applied to improve the initial solution. As the final output, we provide the batch ids, order ids, route sequence, and task completion time (i.e., pick up, drop) in the batch.

This paragraph discusses a *JIT heuristic for postponing an order* and a batch to leverage its optimal batching potential till the promised SLA time. The generic, static, or dynamic PDPTW solution does not guarantee that the last order in a batch will be delivered close to the SLA time (say 45 min). To ensure the final order is delivered close to the SLA time, we propose the following heuristic. For every order in a batch, based on the SLA (say 45 min), SB (say 5 min), and PDPTW recommended delivery time of the order (say 30 min), calculate the buffer time (i.e., 45-5-30). Then, take the minimum buffer time of all orders in the batch. Then, add this buffer time to the PDPTW recommended pickup time of the first order in the batch. Finally, adjust for the "*DE independent FM time*" and *expected assignment time (A2A)* to calculate the batch's *activation time*. Hence, the *activation time* of a batch for assignment would be as follows:

$$\begin{aligned}\text{activation time} = \min_{o \in O}(&SLA - SB - \text{recommended delivery time} \\ &+ \text{recommended pickup time} - \text{A2A} - \text{DE independent FM time})\end{aligned} \tag{3}$$

A batch is sent to the assignment model (i.e., FM optimization) only when the *current time* is more than the *activation time*.

2.2.2 FM Delivery Optimization

The FM delivery problem is formulated as an multi-objective assignment problem, similar to Kottakki et al. (2020). The terminologies and notations are kept similar to Paul et al. (2020). The *parameters* are:

$A2D_{b,d}$: Assigned to Delivery time for batch b by DE d.
$W_{b,d}$: Waiting time for batch b by DE d.
$O2P_o$: Order to Pick up time for order o.
Q : Factor accounts for the differences in travel and waiting time pay.
$CX_{b,d}$: Customer Experience for batch b by DE d.
$Fam_{b,d}$: Familiarity of batch b and DE d.
OET : Order Elapsed Time.
AOV : Average Order Value.
P_b : Penalty of batch b based on the OET and AOV.
α, β : Parameters for penalty of batch b.
$O2D_{o,d}$: O2D time for order o by DE d.
SLA_o : SLA time for order o.
V : A factor to transform cost into time scale.
I : Monetary benefit from early shipping.
A, N : Used for scaling the monetary cost of delay.

The *decision variable* is

$$x_{b,d} = \begin{cases} 1, & \text{if batch } b \text{ is assigned to DE } d \\ 0, & \text{otherwise} \end{cases}$$

and the *objective function* is

$$Cost_{b,d} = A2D_{b,d} - Q \times W_{b,d} - V \times CX_{b,d} - V \times P_b \tag{4}$$

$$\hat{Cost}_b = \hat{A2D}_b - Q \times \hat{W}_b - V \times \hat{CX}_b \tag{5}$$

$$P_b = AOV \times \min(\alpha \times e^{(\beta \times OET)}, 1) \tag{6}$$

$$CX(delay, O2D) = \begin{cases} \dfrac{I \times 10}{SLA - 10}, \text{delay } \leq -10 \\ \dfrac{-I \times delay}{O2D}, -10 \leq \text{delay } \leq 0 \\ -\dfrac{A \times delay \times e^{0.1 \times delay}}{N}, \text{delay } > 0 \end{cases} \tag{7}$$

$$A2D_{b,d} = \underbrace{FM_{b,d}}_{FirstMile} + \underbrace{W_{b,d}}_{Max(O2P_b) - FM_{b,d} - O2A_{o,d}} + \underbrace{LM_{b,d}}_{LastMile} \tag{8}$$

$$O2D_{o,d} = \underbrace{O2A_{o,d}}_{\text{Order to Assignment}} + A2D_{o,d} \tag{9}$$

$$\min \text{ Z } = \sum_{b \in B} \sum_{d \in D} Cost_{b,d} \times x_{b,d} + \sum_{b \in B} \hat{Cost}_b \times (1 - \sum_{d \in D} x_{b,d}) \tag{10}$$

The following *constraints* apply:

$$\sum_{b \in B} x_{b,d} \leq 1, \quad \forall\, d \in D \tag{11}$$

$$\sum_{d \in D} x_{b,d} \leq 1, \quad \forall\, b \in B \tag{12}$$

$$x_{b,d} \in \{0,1\}\ \forall\, b \in B, d \in D \tag{13}$$

Equation (4) represents the cost of assigning a DE d to a batch b combining CPD and CX. Equation (5) represents the cost of not assigning a batch b in the current cron. The parameters (denoted with hat) in Equation (5) are estimated as an average of the existing edges. Equation (6) denotes the penalty of not assigning a batch based on the elapsed time and order value. Equation (7) represents the customer experience for a given delay in the order. The objective function (Equation (10)) is the trade-off between the cost of assigning a batch in the current cron vs not-assigning in the current cron. The constraints (Equations (11) and (12)) are straight forward and ensuring that a batch is assigned to only one DE and vice-versa.

3 EXPERIMENTS AND RESULTS

The whole simulation framework is implemented in *Python* using the open-source simulation package *SymPy* (Matloff 2008). The LM and FM optimizations are implemented using the open-source *Google OR-Tools* and *PuLP*, respectively.

For this experiment, we have considered a sample set of real order from a zone. The model parameters are set based on a business understanding or historical data. We mask some of the business-sensitive parameters (e.g., O2P, W, It, service time, cron time, A2A) for confidentiality. Similar to Paul et al. (2020), we have chosen Safety Buffer (SB) = 5 minutes, LM-Optimization run time = 30 seconds, DE Speed = 12 kmph, Q = 0.2, V = 0.6, I = 50, A = 500, N = 2000, α = 1.3. We fixed the maximum batch size k as 3 for this experiment. The CPD is defined as a function of FM distance, LM distance, Waiting Time, and customer touch point pay.

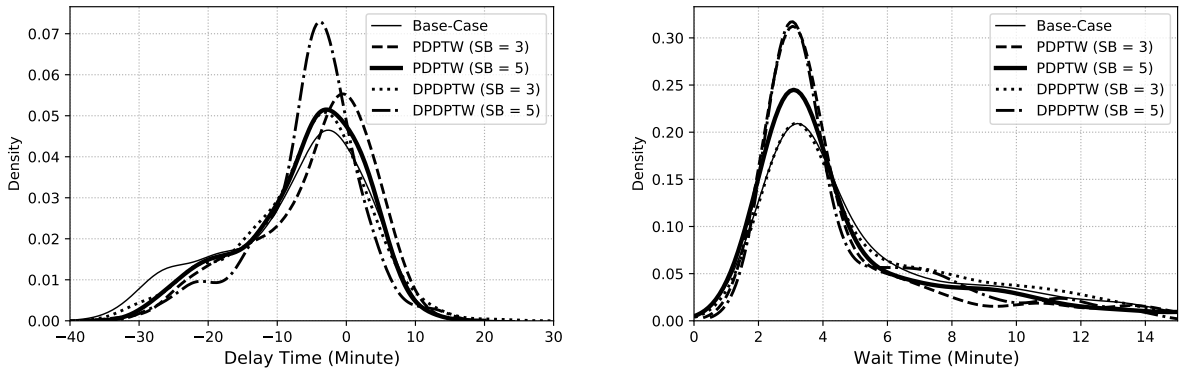
Table 3 shows the simulation results comparing two variants of PDPTW with the *Base Case* in terms of key business metrics. The *Base Case* is simulated based on the existing algorithm used for LM optimization. We limit the paper's scope to use *Base Case* only for comparison and validation purpose and, hence, not discuss it in detail. The business-critical metrics are shown in the form of relative increase (i.e., gain) or decrease (i.e., savings) compared to the *Base Case*. Both static and dynamic variants of PDPTW models are compared against the *Base Case* for two different settings of the *SB* parameter and for the same number of orders. The effect of *SB* parameter variation on any particular model is intuitive. As the Safety Buffer (SB) increases, the system target SLA decreases. Consequently, the SLA compliance increases (e.g., -12.87 % to -3.72 % for 2 minutes increase in SB in the PDPTW model), but at the cost of higher CPD (e.g., 11.07 % to 6.17 % for 2 minutes increase in SB in the PDPTW model). The comparison between *Base Case*, PDPTW (SB =5 min), and DPDPTW (SB =5 min) shows that the PDPTW gives the best CPD (11.07 % savings), whereas the DPDPTW gives the best SLA compliance (5.77 % gain). Further, there is a significant saving in waiting time through the JIT heuristic.

Table 3: Comparison of PDPTW variants.

	PDPTW (SB = 3 min)	**PDPTW (SB = 5 min)**	**DPDPTW (SB = 3 min)**	**DPDPTW (SB = 5 min)**
Batch Size (% Increase)	14.43	3.09	10.82	-12.37
DE Used (% Decrease)	12.03	3.80	3.80	-12.66
Batching (% Increase)	7.95	1.22	3.98	-21.10
O2D Time (% Decrease)	-7.60	-6.75	-3.32	-5.57
FM Time (% Decrease)	3.55	-10.66	-10.15	-38.58
waiting time (% Decrease)	29.60	26.35	14.80	11.55
LM Time (% Decrease)	9.18	4.77	10.23	-0.70
Compliance (% Increase)	-12.87	-3.72	0.34	5.77
FM Distance (% Decrease)	9.66	-3.59	8.23	-17.73
LM Distance (% Decrease)	11.59	6.07	12.95	-0.91
CPD (% Decrease)	11.07	6.17	9.80	-1.48
Assigned Effeciency (% Increase)	12.97	6.65	9.49	-2.85

Therefore, SB for DPDPTW is reduced to 3 minutes from 5 minutes and this variant gives the best CPD and CX (i.e., compliance). The gain in compliance and savings in CPD is 0.26 % and 9.8 %, respectively. The PDPTW with SB = 3 minutes performs very poorly in terms of compliance (12.87 % decrease). The CX and CPD gains lower than the *Base Case* are not acceptable. Hence, we primarily compare *Base Case*, PDPTW (SB = 5 min), and DPDPTW (SB = 3 min). Among the three models, the DPDPTW has the highest batching percentage, average batch size, compliance, lower LM, and lower CPD. This better compliance is possible in the DPDPTW, because of its ability to reshuffle batches till the last moment before DE arrival at the store. This enables to break off any bad batches that may breach the SLA. The Assigned Efficiency (AE) metric denotes the average number of orders delivered per assigned hour. The AE indicates that DPDPTW being the most efficient algorithm.

Figure 6a compares the delay distribution of all models. This indicates that PDPTW (SB=5 min) and DPDPTW (SB=3 min) have a very similar distribution, but the DPDPTW has a relatively sharp decay in the positive delay axis. Figure 6b shows that the majority of waiting time probability mass lies between two and four minutes. Base Case and DPDPTW (SB = 3 min) have a similar distribution.



(a) Delay time distribution

(b) Waiting time distribution

Figure 6: Comparison of delay and waiting time.

4 SUMMARY

A Discrete Event Simulation framework and a two-stage optimization model have been developed in this paper for the on-demand e-grocery delivery problem. The two-stage optimization model comprises an LM-optimization and an FM-optimization module. The proposed optimization model has experimented with real order data in simulation. The LM optimization problem is modelled using two variants of the PDPTW problem, viz. static and dynamic. Further, a JIT heuristic is developed to minimize the waiting time at the store. The DE assignments to batches are handled through the FM optimization by trading off between the CPD and CX. The simulation results showed that the JIT heuristic saves significant waiting time. Further, the PDPTW variants significantly reduce the average LM distance per order. Consequently, the 9.8 % saving in CPD is possible without compromising on compliance (CX). This gain is achieved through the dynamic variant of PDPTW (i.e., DPDPTW) with a safety buffer of three minutes. The DPDPTW model considers the system's current state and locks some of the tasks scheduled within the next few minutes. The DPDPTW reshuffles the non-fixed tasks and orders in a batch until the DE arrives at the store. This process helps in optimizing the batches (i.e., routes) further in terms of CPD and CX.

The proposed model can handle the current scale of orders and provide a real-time solution. The model is getting implemented in production. As future work, we are trying to build intelligence for handling large scale orders within the proposed framework as the business grows.

ACKNOWLEDGMENTS

We sincerely thank our colleagues Vidya Nand, Sreepathy R Naik, and Bharath Nayek for their help in various stages of this work.

REFERENCES

Alonso-Mora, J., S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. 2017. "On-demand High-capacity Ride-sharing via Dynamic Trip-Vehicle Assignment". *Proceedings of the National Academy of Sciences* 114(3):462–467.

Clarke, G., and J. W. Wright. 1964. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". *Operations research* 12(4):568–581.

Kottakki, K. K., S. Rathee, K. M. Adusumilli, J. Mathew, B. Nayak, and S. Ahuja. 2020. "Customer Experience Driven Assignment Logic for Online Food Delivery". In *2020 IEEE International Conference on Industrial Engineering and Engineering Management*. December 14th–17th, Singapore, 827–831.

Leyerer, M., M.-O. Sonneberg, M. Heumann, and M. H. Breitner. 2020. "Shortening the Last Mile in Urban Areas: Optimizing a Smart Logistics Concept for E-Grocery Operations". *Smart Cities* 3(3):585–603.

Liao, W., L. Zhang, and Z. Wei. 2020. "Multi-objective Green Meal Delivery Routing Problem Based on a Two-stage Solution Strategy". *Journal of Cleaner Production* 258:120627.

Liu, D., Z. Deng, X. Mao, Y. Yang, and E. I. Kaisar. 2020. "Two-echelon Vehicle-routing Problem: Optimization of Autonomous Delivery Vehicle-assisted E-grocery Distribution". *IEEE Access* 8:108705–108719.

Matloff, N. 2008. "Introduction to Discrete-event Simulation and the Simpy Language". *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August* 2(2009):1–33.

Pan, S., V. Giannikas, Y. Han, E. Grover-Silva, and B. Qiao. 2017. "Using Customer-related Data to Enhance E-grocery Home Delivery". *Industrial Management & Data Systems* 117(9):1917–1933.

Paul, S., S. Rathee, J. Matthew, and K. M. Adusumilli. 2020. "An Optimization Framework for On-Demand Meal Delivery System". In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. December 14th–17th, Singapore, 822–826.

Ramos, T. R. P., M. I. Gomes, and A. P. B. Póvoa. 2020. "Multi-depot Vehicle Routing Problem: A Comparative Study of Alternative Formulations". *International Journal of Logistics Research and Applications* 23(2):103–120.

Reyes, D., A. Erera, M. Savelsbergh, S. Sahasrabudhe, and R. O'Neil. 2018. "The Meal Delivery Routing Problem". *Optimization Online*. http://www.optimization-online.org/DB_HTML/2018/04/6571.html, Accessed March 16, 2021.

Ulmer, M. W., B. W. Thomas, A. M. Campbell, and N. Woyak. 2021. "The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times". *Transportation Science* 55(1):75–100.

Vazquez-Noguerol, M., J. Comesaña-Benavides, R. Poler, and J. C. Prado-Prado. 2020. "An Optimisation Approach for the E-grocery Order Picking and Delivery Problem". *Central European Journal of Operations Research*. https://doi.org/10.1007/s10100-020-00710-9.

Voudouris, C., and E. Tsang. 1999. "Guided Local Search and its Application to the Traveling Salesman Problem". *European Journal of Operational Research* 113(2):469–499.

AUTHOR BIOGRAPHIES

SIDDHARTHA PAUL is an Operations Research Scientist at Swiggy. At Swiggy, as part of the core logistics team, he has been primarily working on developing low latency and scalable solutions for the hyper-local logistics optimization problems for both food and grocery. Before joining Swiggy, he was a Data Science consultant at antuit.ai where he worked with clients from logistics and agriculture industry. He has received his M.Sc. and Ph.D. in Operations Research from the Department of Industrial Engineering and Operations Research, Indian Institute of Technology Bombay. His broader research interests include, but are not limited to, simulation, optimization, mathematical modelling, and statistical analysis. His e-mail address is paulsiddhartha0@gmail.com and his web address is https://paulsiddhartha.wordpress.com/.

GODA DORESWAMY is AVP, Data Science at Swiggy handling delivery logistics optimization for both food and new initiatives in grocery and pick-up-and-drop use cases. Previously she worked with xto10x, establishing the right data science journey and practice in the startup ecosystem as GM, Data Science. Prior to that, she was at Ola for 2 years as Principal Data Scientist leading the machine learning and optimization algorithms behind pricing and matching for on demand ride sharing. She worked at Sabre for 10 years before Ola on a variety of problems in Airline Pricing and Revenue Management. She holds an undergraduate and Masters degree from IIT Madras. She has been ranked as one of "Top 10 Data Scientists in India" by Analytics India Magazine in 2018. Her e-mail address is goda.doreswamy@swiggy.in.