i-Deliver: A Crowdsourcing Platform for Delivery-as-a-Service

Theodoros Ioakeimidis¹, Dionysios Kehagias¹,*, Spyros Kontogiannis²,³, Petros Melissaridis¹, Andreas Paraskevopoulos²,³, Georgia Xanthopoulou¹, Christos Zaroliagis²,³

1. Centre for Research and Technology Hellas – Information Technologies Institute, Greece
2. Department of Computer Engineering and Informatics – University of Patras, Greece

*diok@iti.gr

Abstract
Delivery-service enterprises need to achieve high quality of service and at the same time keep their operational costs affordable. The fulfillment of the customer demands depends on various aspects, e.g., timely pickup and delivery of goods, delivery fees, customer support, etc. As a result, the idea of a unified platform for both the deliverers and the customers that could act as a broker and facilitate the delivery management is currently a very important but also very challenging task. We present i-Deliver, a platform that provides a delivery management system which collects all orders (delivery requests with their own time constraints) and available resources (workers with their own vehicles) in one place, and then orchestrates and audits the service of all requests by the available workers in real-time.

Keywords:
last-mile, pick-up and delivery vehicle routing with time windows, traffic prediction, crowdsourcing

1 Introduction
Shared mobility refers to transportation resources (e.g., transportation employees with their own vehicles, or workers) that jointly serve transportation demands (transport requests, or orders), such as ride-sharing, or delivery of goods (e.g., food and beverages, parcels, etc.). A key enabler for practical shared mobility is the mechanism for providing order-servicing routes for workers to serve as many orders as possible. By altering order-servicing routes of the workers in a given route plan, to avoid both under-utilization of resources and unnecessary operational costs, shared mobility manages to mitigate pollution, reduce transportation costs, and provide last-mile delivery. In this work we focus on delivery services, therefore a worker is an active courier with his/her own vehicle; and an order specifies an origin for pick-up and a destination for drop-off of a particular good under certain spatiotemporal constraints. A pickup-and-delivery management (PDM) mechanism proposes a single route per worker, i.e., a sequence of consecutive pickup and delivery locations for orders that are assigned to the worker during their working shift. This should be done in such a way that (i) a maximum number of orders is served, and (ii) the operational cost for having them served is minimized. Moreover, as the orders may be revealed to the management system in real-time, the PDM mechanism should be able to adapt the current routes of the workers so that the new orders are also served, if possible.
This paper presents *i-Deliver*, a crowdsourced PDM platform that gathers all submitted orders in real-time, and orchestrates the tasks of all active workers. The aim is to serve as many orders as possible, in such a way that the overall service cost is minimized but also all the spatiotemporal constraints of both the workers and the orders are absolutely respected. Crowdsourced information by the workers provides the necessary information for the platform to audit the service of each active order, if not rejected.

## 2 Architecture Outline

The *i-Deliver* platform is a PDM software for serving pickup-and-delivery requests (*orders*), auditing order records, client support, etc., all in one place. It is utilized by a fleet of delivery drivers (*workers*), enterprises employees and clients. All these stakeholders provide the platform with data in real time, such as the workers’ profile, order details (e.g., pickup and delivery locations and deadlines, etc.), through a crowdsourcing mechanism. It is also in charge of orchestrating the service of orders by the workers, exploiting traffic-prediction and demand-forecasting algorithms, and always respecting spatiotemporal constraints for both the workers and the orders. Towards this direction, the platform has been designed according to a layered architecture, where each layer is responsible for a certain procedure, and has the ability to interact with the underlying layer. In short, the *i-Deliver*’s layered architecture consists of: (a) the presentation layer (User Interface); (b) the communication layer; and (c) the delivery-service application layer (i.e., backend services and data subsystem) containing all the necessary backend services. The architecture diagram of the *i-Deliver* platform is presented in Figure 1. In what follows, the key functionalities of the modules of all layers of the *i-Deliver* platform are described.

![Figure 1: The architecture of the i-Deliver platform.](image)

## 3 Presentation Layer

The *i-Deliver* mobile application for delivery drivers provides the following features: (a) Authentication of users by signing in with their own username, password and domain. The domain defines the stores, which will be available to serve and the global settings. (b) Access to user location: The application requests permission, when running either in the foreground or in the background, to periodically send user’s location and speed to the backend. (c) Initiation of working shift: The shift can be started by the
worker’s entering the current indication of kilometers and the licence plate of their vehicle, as well as the store which will be served. The list of available stores depends on the domain. (d) List with the active (i.e., not yet served) orders allocated to the worker, which appears so long as the working shift is still active. The list is refreshed periodically, the refresh rate is determined by the backend. In addition, a sound alert informs the worker for the assignment of a new order. Finally, five different colours are used to help users to distinguish the status of delivery requests. (e) When the worker chooses a particular order assigned to them, the details are shown. Its status can be identified as (1) “go to pick up from store”, (2) “arrive at store to pick up”, (3) go to deliver to customer, (4) “delivered”, or (5) “completed”. (f) Show on-screen available actions of a delivery request: The available actions are (1) “delay on delivery time”, (2) “cancelation of delivery”, (3) “failure to delivery”, or (4) “emergency request”. (g) A calculation of optimal route for serving delivery requests assigned to a worker is available, for minimizing the delivery time. Also, navigation to the delivery points using Google Maps is provided. (h) The shift can be terminated by entering the final indication of kilometers of the vehicle, expenses and comments of the worker. (i) The historical data for the completed delivery requests during the last shift can be viewed. (j) A new delivery request can be created. (k) The preferences can be shown on-screen with the following options: (1) projection of delivery requests (detailed list or short list); (2) projection only the requests of current timeslot; (3) selection of store which will be served; (4) selection of route; (5) request to delete all user’s personal data (GDPR). Some screens from the i-Deliver mobile application are presented in Figure 2 and Figure 3.

4 Communication
The i-Deliver API is based on the REST (REpresentational State Transfer) design principles and mainly offers the following core functionalities: (a) access/role management and (b) delivery request assignment. Regarding access/role management, the API provides endpoints for creating, updating and deleting all the data entities of the i-Deliver system.

The manipulation of these entities is performed with the use of HTTP methods (i.e., GET, POST, PATCH and DELETE). The main user roles in the i-Deliver system are the following: (1) Store: the role that represents a physical store and needs to assign delivery requests to delivery drivers. (2) Delivery drivers: the entity responsible for delivering orders to customers; users having this role communicate with the API solely through the i-Deliver mobile app. (3) Customer: the role that represents a person that creates delivery requests. The system supports both registered and unregistered customers.

Each role has access to different parts of the resources of the i-Deliver platform through the API. The authorization rights of each user role are determined by providing proper credentials via the Bearer Token authentication mechanism. Regarding the allocation of orders to workers, the API is the interface through which the corresponding allocation algorithms are invoked. In particular, the i-Deliver API has the following types of endpoints for delivery request management: (1) retrieve the assignments suggested by the allocation algorithms; (2) verify the results suggested by the algorithms; (3) retrieve the confirmed results. The API provides all required input to the algorithms and decouples the i-Deliver users from the complex internal functionalities of the delivery requests assignment process.
i-Deliver: A crowdsourcing platform for delivery as-a-service

Figure 2: (a) Login and provision of access to the device location; (b) listing of the delivery requests assigned to the delivery driver; (c) detailed view of delivery request; (d) steps of the delivery request processing.

Figure 3: i-Deliver mobile app - Route planning for delivery

5 Backend Module for Traffic Prediction & Demand Forecasting

One of the most important challenges faced by businesses operating in the field of goods distribution is
the effective and efficient delivery of their products. Traffic forecasting (combined with other optimization techniques) can improve the process of receiving and delivering goods, by accurately predicting the delivery times of goods using information for the real traffic conditions of delivery networks. One of the key technological innovations of i-Deliver is its traffic forecasting module.

The traffic forecasting module of the i-Deliver platform integrates a set of forecasting algorithms that can predict the state of an entire traffic network (i.e., of all the roads of the network) at multiple time in the future. These algorithms use data aggregated from the traffic network in order to generate their predictions. In particular, the data in their original form are GPS traces derived from the i-Deliver mobile application for the delivery drivers, and include as information the id of the recording, the recording time, the id of the delivery driver, the location (in terms of coordinates) of the delivery driver at the time of the recording, and the speed of the delivery driver’s vehicle at the time of the recording. These data records are initially matched to real (prespecified) road segments of the traffic network by the means of map-matching algorithms. Then, they are matched to specific time intervals, thus forming traffic time series (i.e., time series of speed values). Each interval in these time series has a length of five minutes.

After the traffic time series of all road segments of the network have been formed, the traffic network is segmented into groups of roads segments using the k-means clustering algorithm. In this clustering process, the road segments are represented by their spatial features, i.e., the coordinates of their start and end nodes. The clustering process also facilitates the imputation of missing values, in the cases where there are no values for several time intervals of the traffic time series of multiple road segments.

When the traffic network has been segmented into clusters, a different instance of each traffic forecasting algorithm is constructed for each different cluster. The traffic forecasting algorithms used by the i-Deliver platform are the following: (a) a persistence model, (b) a moving average model, (c) a linear regression model, (d) a k-nearest neighbors regressor, and (e) a multilayer perceptron regressor. These models were subjected to a comparative evaluation process in terms of forecasting accuracy (measured using the root mean square error (RMSE) and the mean absolute percentage error (MAPE) metrics). The benchmarking results indicated that the linear regression model achieves the best forecasting accuracy, in most forecasting scenarios, for all groups of road segments. Hence, this is the model used for generating predictions by the traffic forecasting module of the i-Deliver platform.

6 Backend Module for Pickup & Delivery VRP with Time Windows

The core service for a delivery-as-a-service platform is the well-known Vehicle Routing Problem (Parragh, Doerner, & Hartl, 2008) (Toth & Vigo, 2014). In our case we use the variant of VRP with Pickup & Delivery Points and with Time Windows (VRP-PD-TW) (Parragh, Doerner, & Hartl, 2008), which determines how a given set of pickup and delivery requests is to be served by a prespecified fleet of deliverers, under certain spatiotemporal constraints.

6.1 VRP Pickup & Delivery with Time Windows: Problem Statement

An instance of VRP-PD-TW consists of a set R of orders (the pickup-and-delivery requests to be served), a set W of workers (the deliverers), the corresponding set H of their working shifts, and a set C of spatiotemporal constraints and parameters that regulate the context in which the service of each order is feasible and acceptable. The output consists of a set P of routes, one per active worker, in the
underlying road network with time-dependent traversal-times at each road segment, which aim to minimize the aggregate cost to service all the orders. Each such path, that is associated with a specific worker \( w \in W \), is required to be feasible with respect to all the relevant spatiotemporal constraints for the orders assigned to \( w \), in such a way that it does not violate any temporal restrictions of \( C \) (concerning earliest-pickup and latest-delivery deadlines for the orders) or \( H \) (concerning the working shifts and the capacities of the involved vehicles). The cost of each solution is assessed through a customizable objective function. The primary objective is to serve as many orders as possible, without violating any of the spatiotemporal constraints related to them. Given that, as an optimization criterion we consider the minimization of either the aggregate distance, or the total travel duration of the involved workers to serve these orders. A secondary objective is to provide balanced (or at least justified by an appropriate rewarding scheme) amounts of work to the workers.

A VRP-PD-TW instance consists of a collection of orders, \( r_1, \ldots, r_{|R|} \in R \) and a set of workers, \( w_1, \ldots, w_{|W|} \in W \). A request is represented by a tuple, \( r_i = (t_i, c_i, l_i, p_i, d_i, t^e_{p_i}, t^d_{d_i}) \), where \( t_i = \) release-time of the order, \( c_i = \) load capacity requirements, \( l_i = \) required type of vehicle, \( p_i = \) pickup point, \( d_i = \) delivery point, \( t^e_{p_i} = \) earliest-pickup time and \( t^d_{d_i} = \) latest-delivery time. Analogously, a worker is represented by a tuple \( w_j = (b_j, f_j, t^s_{b_j}, t^d_{f_j}, g_j, y_j) \), where \( b_j = \) starting point of \( j \)’s shift, \( f_j = \) ending point of the shift, \( t^s_{b_j} = \) start-time of the shift, \( t^d_{f_j} = \) end-time of the shift, \( g_j = \) maximum vehicle capacity and \( y_j = \) vehicle type. A set \( C \) contains some spatiotemporal feasibility constraints: i) each order is served by a single worker, who will first pick up the good from an origin and will then deliver it to a destination, using an eligible type of vehicle for the good and respecting the corresponding deadlines for the delivery; ii) each good has capacity requirements for its storage in the worker’s vehicle; iii) all workers have specific vehicles with given maximum capacities for storage and cannot be assigned any goods that exceed the capacity of their vehicles, at any time.

For the needs of the backend service (abbreviated as bPnD) responsible for the provision of solutions to VRP-PD-TW instances, the instance is first represented by a “pickup-and-delivery” (PD) graph. Based on this graph, a mixed-integer linear program (MILP) is then constructed to guarantee that all the spatiotemporal constraints are respected by any feasible solution, while optimizing the selected objective function. The representation of the VRP-PD-TW instance as a PD graph, \( G = (V, E) \), involves the encoding of the entire set of pickup, delivery, and potential movement events, using the set of nodes \( V \) representing the pickup and delivery events, and the set of edges \( E \) indicating the potential movements of workers from one event to the next. Each pickup or delivery event is accompanied by its actual coordinates in the underlying road map. When a worker traverses an edge, s/he is “charged” according to two cost criteria: i) the distance from the tail to the head of the edge, independently of the used vehicle; and ii) the traversal-time to move from the tail to the head of the edge (based on historical traffic data), which depends on the vehicle type (motorbike, car, truck) and the arrival-time of the deliverer at the tail of the edge. The dataset regarding the actual road network is provided by the OpenStreetMap\(^1\) service. The travel-time metric is provided by OpenStreetMap and the traffic prediction service of i-Deliver.

For the construction of the PD graph \( G \), a preprocessing step is initially carried out in the underlying

\(^1\) https://www.openstreetmap.org
road network. First, the required geographical points (i.e., starting/ending points of shifts, pickup/delivery points of orders) are identified in the road network. This procedure involves finding the nearest points on road segments of the road network. The search is efficiently implemented using an R-tree (Guttman, 1984), utilizing the indexing and classification of the coordinates in geographic partitions in the form of rectangular cells. Second, we conduct sequential shortest-path tree computations to provide a set of (one-to-many) optimal routes among these points of interest, minimizing either the distances or travel-times from each node of $G$. We use executions of Dijkstra’s algorithm (Dijkstra, 1959) for the (static) distance metric, and state-of-art travel-time oracles (Kontogiannis, Papastavrou, Paraskevopoulos, Wagner, & Zaroliagis, 2017) for the (departure-dependent) travel-time metric. The resulting paths in the road network are represented in a condensed manner, as edges of $G$. Eventually, the construction of the PD graph (Figure 4) is carried out in such a way that: a) each node in the graph represents a distinct geographical point on the road network, at which a time-dependent event of \{arrival/departure\} of a worker, a \{pick-up/delivery\} of an order, or \{start/end\} of a working shift may take place; and b) each edge on the graph represents the transitions and successive visits between two nodes. It is worth mentioning at this point that, for any pair of nodes in $G$, we create two parallel edges between them, one representing the minimum distance (with a possibly suboptimal travel-time) and one representing the (time-dependent) minimum travel-time (with a possibly suboptimal distance) between them. The nodes of $G$ are grouped into type-b nodes representing the beginning of a worker’s shift, the type-p nodes representing pickups of goods that have been ordered, the type-d nodes representing deliveries of goods to the consumers, and the type-f nodes representing the end of a worker’s shift: $V = \{ b_i: i \text{ is a worker’s shift from } H \} \cup \{ p_j: j \text{ is an order from } R \} \cup \{ d_j: j \text{ is an order from } R \} \cup \{ f_i: i \text{ is a worker’s shift from } H \}$. The following edges of $G$ are then considered: $E = \{ (b_i, p_j): i \text{ is a worker’s shift from } H, j \text{ is an order from } R \} \cup \{ (p_j, d_k): j, k \text{ are orders from } R \} - \{ (d_j, p_j): j \text{ is order from } R \} \cup \{ (d_j, f_i): i \text{ is a worker’s shift from } H, j \text{ is an order from } R \}$.

Figure 4: Representation of a VRP-PD-TW instance with the PD graph (left) and the MILP (right).
For each pair \((u, v) \in V \times V\), there exist two parallel edges from \(u\) to \(v\), one for each of the two optimization criteria (distance, or travel-time), corresponding to two Pareto-optimal \((u, v)\)-routes in the underlying road network, per different vehicle type: For each vehicle type \(h\), the first \((u, v)\)-route, \(\pi^1_{h,u \rightarrow v}\), is length-optimal, while the second \((u, v)\)-route, \(\pi^2_{h,u \rightarrow v}\), is travel-time-optimal. When solving the VRP-PD_TW instance at hand, for each of these two parallel edges, at most one may be selected as part of the solution, depending on the optimization criterion that we consider and considering all the feasibility constraints, since only one deliverer can serve an order.

The inclusion of two Pareto-optimal routes per pair \((v, u) \in V \times V\) and different type of vehicle serves in case that the objective is the travel-distance minimization. In such case, it is necessary to take both routes \(\pi^1_{h,u \rightarrow v}\) and \(\pi^2_{h,u \rightarrow v}\) into account, because the primary objective is to construct a feasible solution with the maximum number of satisfied orders, that respects all spatiotemporal constraints. When only the distance-minimal routes are not sufficient to provide such a feasible solution, due to violation of some time constraints, the alternative of (some) travel-time-minimal routes allows the model to overcome this infeasibility and provide at least a feasible solution with the maximum number of served orders, exploiting only the necessary number of these travel-time-minimal routes to recover feasibility. E.g., in the example of Figure 5 a worker with a motorbike arrives at \(u\) at 8:00. The arrival at node \(v\) via the path \(\pi^1_{h,u \rightarrow v}\) is at 8:45, but the latest-delivery-time deadline is until 8:30. On the other hand, via the path \(\pi^2_{h,u \rightarrow v}\) the worker arrives at \(v\) at 8:27, i.e., before the deadline, traveling for a slightly longer distance. Therefore, the only option for this worker to move from \(u\) to \(v\) in \(G\), is via the route \(\pi^2_{h,u \rightarrow v}\). The representation of the VRP-PD-TW instance as a mixed-integer linear program (see Figure 4) consists of a set of linear inequalities that determine: i) the set of feasible solutions based on the constraints of the instance; and ii) the objective cost function that determines the optimal one(s). The coefficients of the linear inequalities are described by a matrix \(A\) and a vector \(b\) of constants, in the form \(Ax \leq b\). The rows correspond to spatiotemporal constraints and the columns correspond to continuous, or discrete (decision) variables. The decision variables determine which edges of \(G\) are used in the solution, as part of the collection of paths that serve the orders, or indicator variables for assigning orders to workers. The continuous variables determine actual pickup and delivery times, as well as the vehicles actual capacities while serving orders.

### 6.2 Algorithms for VRP-PD-TW

The aim of the bPnD service is twofold: (i) to partition the orders among the active workers; and (ii) to determine a set of non-overlapping and independent routes in \(G\), one per worker, corresponding to (possibly overlapping) routes in the road network, which specify the exact way that the workers serve the orders assigned to them, in such a way that there is no violation of the spatiotemporal constraints of the instance, while also minimizing (exactly or approximately) the cost of the provided solution, as determined by the objective function. The process of constructing a feasible solution is done in two
modes: a) the **offline mode**, in which all the orders and working shifts are known in advance; and b) the **online mode**, where both orders and workers are revealed in real time to the solver, which must then adapt the current solution appropriately. The **offline scheduler** solves the entire mixed-integer linear program constructed for the instance at hand, using the “Brunch and Bound” method (Land & Doig, 1960). The **online (re)scheduler** is used along with the PD graph, to support the dynamic operations of: a) updating the pool of active workers, upon the entry of new workers who just started their working shift; and b) scheduling new orders that are revealed in real-time.

In the online mode, the rescheduling algorithm must confront a constantly changing state, as new working shifts may start at any time, some others will terminate, new orders may be released and some of the already allocated orders will be completely served by some workers. The focus of the **bPnD** service is to provide, as fast as possible, immediate decisions on how to adjust the allocation of the still pending orders to the available workers, in a very efficient manner. The proposed **plain-insertion algorithm** adopts and extends an idea in (Tong, Zeng, Zhou, Chen, & Xu, 2022): upon the release of a new order, it actually tries the unilateral extension of each worker’s scheduled path, so as to add this order to their schedule in a locally-optimal but also feasible manner, and then simply allocates this new order to the workers that can fit it in their schedule with the minimum additional overhead to the objective value. The experimental evaluation (see Section 0) has shown that the use of the constraint-based “plain insertion” algorithm is quite efficient not only for the online mode, but also as a heuristic method for the offline mode as well (handling the orders one by one, in an arbitrary sequence).

### 6.3 Experimental Evaluation of VRP-PD-TW

We evaluated our algorithms using historical data of actual orders for a particular working day (Wednesday, 3/11/2022) in the Greek city of Ptolemaida. Our experiment evaluates the performance of the online rescheduling service of **bPnD**. For comparison purposes, the real-time, human-decided, and GPS-recorded delivery routes from the i-Delivery platform’s database were considered as the baseline. The experiment is performed in relation to the set of active deliverers and the pickup-delivery orders that appeared during the time-windows 1:[06:00 - 8:00], 2:[08:00 - 10:00], …, and 9:[22:00 - 23:59].

The experimental evaluation was conducted over three different scenarios (for each of the nine time-windows): (1) **Raw**: only the basic restrictions from the database are considered. (2) **Constraint I**: the load requirement of each order is set to 1 unit, and the maximum load capacity of all deliverers’ vehicles is set to 3 units. (3) **Constraint II**: active deliverers who have travelled less than 1.5 times the average distance travelled by all the deliverers so far, are being prioritized towards being assigned a newly released order, so that a more balanced schedule for the active schedulers is produced.

Our results (cf. Figure 6) show that the **plain-insertion (re)scheduling** online solver of **bPnD** computes a solution that achieves a significant improvement percentage in the overall cost, both in terms of both travel-distance and travel-time metrics, compared to the human-aided routes that were stored in the i-Deliver database. Naturally, the incorporation of additional constraints into the scenarios negatively affects the cost minimization, as the number of acceptable solutions decreases to meet the constraints.

### 7 Summary and Future Work

We presented i-Deliver, a crowdsourcing platform for delivery-as-a-service, along with the key
i-Deliver: A crowdsourcing platform for delivery as-a-service

functionalities of its constituent modules. The platform is currently deployed at different cities and its real-world operation will be monitored and assessed in the forthcoming months.

Figure 6: Percentage improvement in total length and travel time depending on the problem constraints.

Acknowledgements

This work was co-financed by the European Regional Development Fund of EU and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation (call RESEARCH–CREATE–INNOVATE) under contract no. T2EDK-03472 (project “i-Deliver”).

References


