Novel online routing algorithms for smart people-parcel taxi sharing services

Son Nguyen Van\(^1,2\) | Nhan Vu Thi Hong\(^3\) | Dung Pham Quang\(^1\) | Hoai Nguyen Xuan\(^4\) | Behrouz Babaki\(^5\) | Anton Dries\(^6\)

\(^1\)Hanoi University of Science and Technology, Hanoi, Vietnam  
\(^2\)Academy of Cryptography Techniques, Hanoi, Vietnam  
\(^3\)Faculty of Information and Technology, University of Engineering and Technology, VNU, Hanoi, Vietnam  
\(^4\)Faculty of Information Technology, HUTECH University, Ho Chi Minh, Vietnam  
\(^5\)HEC Montréal, Montreal, Quebec, Canada  
\(^6\)KU Leuven, Leuven, Belgium

Abstract
Building smart transportation services in urban cities has become a worldwide problem owing to the rapidly increasing global population and the development of Internet-of-Things applications. Traffic congestion and environmental concerns can be alleviated by sharing mobility, which reduces the number of vehicles on the road network. The taxi-parcel sharing problem has been considered as an efficient planning model for people and goods flows. In this paper, we enhance the functionality of a current people-parcel taxi sharing model. The adapted model analyzes the historical request data and predicts the current service demands. We then propose two novel online routing algorithms that construct optimal routes in real-time. The objectives are to maximize (as far as possible) both the parcel delivery requests and ride requests while minimizing the idle time and travel distance of the taxis. The proposed online routing algorithms are evaluated on instances adapted from real Cabspotting datasets. After implementing our routing algorithms, the total idle travel distance per day was 9.64% to 12.76% lower than that of the existing taxi-parcel sharing method. Our online routing algorithms can be incorporated into an efficient smart shared taxi system.

KEYWORDS
Internet-of-Things application, online routing algorithm, routing problem, taxi demand prediction, taxi-parcel sharing

1 | INTRODUCTION
Transportation in urban cities has become a worldwide problem owing to the rapidly increasing global population. To reduce traffic congestion and environmental concerns, the number of vehicles on the road network can be reduced by sharing mobility [1,2]. Ridesharing models attempt to optimize the design of schedules and routes in dial-a-ride problems (DARPs) [3–5]. As DARPs are NP-hard, they are usually solved by heuristic approaches [3,5–7].

Smart ridesharing services can be realized through the convergence of location-based devices, geographic information systems, global positioning systems, and wireless communication. In practice, the transport demands of people and goods tend to overlap during rush
hours. Although the integration of logistics and transportation into multiple modes of transportation with ridesharing has received much attention, the number of studies remains limited [8,9]. Even fewer studies have sought an efficient solution to the share-a-ride problem (SARP) of people and goods over relatively short transportation distances [10,11]. Most similar to our study are conceptual and mathematical models of dynamic SARP (DSARPs) for taxi-sharing services [12–14]. Since the first DSARP model by [12], this problem has been solved by two heuristic algorithms [13,14]. Services accumulate information such as pickup and drop-off locations along with time constraints pertaining to the users’ requests. This information is integrated with other data for later statistical analysis, service customization, and decision-making. The quality of delivery services greatly depends on the performance of the routing techniques. A smart taxi-routing system based on a historical data analysis can improve the operational efficiency of drivers and optimize the overall travel efficiency. However, it appears that most researches have optimized only the total route distance of all current events on the schedules. Few studies have considered where the driver should go when the scheduled tasks are completed [10,11].

The present study tackles the online taxi-share scheduling and routing problem (O-TSSP), in which a taxi travels to a specified parking place when no service request appears on the schedule. Our model alleviates the deficiencies of the DSARP model by adding a set of capacitated parking places. Unlike the DSARP model, our model assumes that not all parcel requests are known in advance but instead supports real-time bookings for both parcel and people deliveries. A parcel request will be inserted in the delivery schedule only if it does not interrupt passenger travel, which is assumed to be continuous at all times. Fixed-schedule events are implemented along the shortest path, but requests received in real-time require a different routing mechanism. Applying a spatiotemporal Poisson point process, we first predict the locations of the road network at which requests are likely to be issued at a particular time of day. Applying the learned information to the routing problem, we attempt to maximize the overall travel efficiency while minimizing the idle time of a driver. The first algorithm, named online taxi scheduling framework (OTSF), directs a taxi to the nearest parking location from the last drop-off point. The second algorithm, named OTSF with demand prediction (OTSF-DP), allows the driver to select another parking place with a high frequency of requests. Based on historical bookings, the success rate is determined as the ratio of the received number of requests at that parking location to the total number of requests issued within a radius of that location. The algorithm also finds the optimal route from the origin to the parking site that maximizes the probability of receiving a new request. This optimization reduces the idle time and travel distance of a taxi sent by the company. Finally, we evaluate the efficiency of the proposed OTSF and OTSF-DP algorithms on a real dataset and compare their results with those of DSARP methods [12] under different parameter settings. The DSARP method was then adapted for applying the predicted information on parking places for unloaded taxis. In the experimental evaluation, our algorithms reduced the total idle travel distances per day by 9.64% to 12.76% from those of the DSARP algorithms, raising the profit from 2520 to 5730 USD per day. Our algorithms can be applied in smart people-parcel taxi sharing systems that effectively utilize the taxi space, limit the number of vehicles on the road, and (most importantly) provide customers with the same quality as traditional cab services.

The remainder of this paper is organized as follows. Section 2 presents a model of the O-TSSP problem and Section 3 details our proposed method for predicting the information of future requests. Our new anticipatory methods for solving the O-TSSP are described in Section 4. Experimental results and comparisons are given in Section 5. Section 6 concludes the paper.

2 | STATIC TAXI-SHARE ROUTING PROBLEM

2.1 | Problem description

A taxi system receives a large number of people- and goods-transportation requests during fixed working hours. Each request contains information about the pickup location, drop-off location, and the valid time window between the two locations. A taxi routing system can be static or dynamic. In static mode, the system assesses the available information of the requests and performs scheduling and routing to serve the maximum number of requests at the minimum total cost. However, real-time requests are dynamic and each taxis’ status and route must be continuously monitored by the system. A dynamic instance can be considered as a series of static instances composed of currently pending requests. When all requests are fulfilled, the taxi is directed toward an identified parking place. By analyzing the collection of historical requests, the system can predict the probability of a request being issued at a certain time and space and route the taxi to that location.
The O-TSSP problem can be defined as a pickup-delivery vehicle routing problem with time windows on a graph. Let $G = (P^*, E)$ be a directed graph with a set of vertices $P^*$ describing four types of spatial points in a geographical region, namely, parking points, pickup points, and drop-off points pertaining to taxi requests, and road points. The set $E$ consists of arc-ed edges connecting pairs of vertices. The routing problem aims to fulfill all requests while reducing the total idle travel distance of the taxis. The system must also recommend the best route by which a taxi driver without a load maximizes its chance of receiving a new transportation demand when the taxi is still available. In the example of Figure 1, a taxi without a load can reach the parking point from its last drop-off point by three candidate routes.

### 2.2 Problem formulation

This subsection describes the O-TSPP problem in detail. The inputs are as follows:

**Input**

- $[t_s, t_e]$: the working-time interval of the routing system (accurate to one second).
- $R$: a set of taxi requests. Each request $r \in R$ is associated with a parameter $tp(r)$ determining the type of request: a ride request from a passenger ($tp(r) = 1$) or a request for a parcel delivery ($tp(r) = 0$). Each request $r$ also has a pickup point $r_p$ and a drop-off point $r_d$. These points are located within a region $D$.
- $L$: The number of taxis (taxis are numbered from 1 to $L$). Each taxi $k$ starts from location $p^k_s$ and terminates at location $p^k_t$ within the area $D$ ($\forall k = 1, L$).
- All pickup and drop-off points associated with requests are stored in the set $P_R = \{r_p, r_d\}_{r \in R}$, named as the set of request-related points.

**Variables used in the model**

- $s(p)$: a point immediately following point $p$ \forall $p \in P \setminus P_T$.
- $id(p)$: the identifier of a taxi’s route passing through the point $p$ \forall $p \in P$.
- $od(p)$: position of a point $p$ on route $id(p)$ \forall $p \in P$.
- $ad(p)$: the total travel distance from the starting point of route $id(p)$ to point $p$, \forall $p \in P$.
- Arrival time $at(p)$ and departure time $dt(p)$ of taxi $id(p)$ at point $p$ \forall $p \in P$.
- $b(p, t) = 1$ if a taxi is parked at point $p$ at time $t$ and 0 otherwise \forall $p \in P_Q, t \in [t_s, t_e]$.

**Constraints of the problem**

- (a) $s(p) \neq p$ \forall $p \in P \setminus P_T$.
- (b) $od(s(p)) = od(p) + 1$ \forall $p \in P \setminus P_T$.
- (c) $id(p) = id(s(p))$ \forall $p \in P \setminus P_T$.
- (d) $ad(s(p)) = ad(p) + d(p, s(p))$ \forall $p \in P \setminus P_T$.
- (e) $id(r_p) = id(r_d)$ \forall $r \in R$.
- (f) $od(r_p) < od(r_d)$ \forall $r \in R$.
- (g) $od(r_p) = od(r_d) - 1$ \forall $r \in R, tp(r) = 1$.
- (h) $at(s(p)) = dt(p) + t(p, s(p))$ \forall $p \in P$.
- (i) $dt(p) = at(p) + dr(p)$ \forall $p \in P_R$.
- (j) $dt(p) > at(p)$ \forall $p \in P_Q$.
- (k) $e_p \leq at(p) \leq l_p$.
- (l) $b(p, t) = at(p) \leq t \wedge t \leq dt(p)$ \forall $p \in P_Q, t \in [t_s, t_e]$.
(m) $\sum_{p \in P} g[p, q]b(p, t) \leq c(q) \ \forall q \in Q, \ t \in [t_s, t_e]$.

Constraints (a)–(c) guarantee that a route is a sequence of points. Constraint (d) indicates the accumulated distance whenever a taxi moves to a new point and also eliminates the emergence of sub-tours. Constraint (e) specifies that the pickup and drop-off points of a request must be on the same route. Constraint (f) states that the pickup point must be visited before the drop-off point. Constraint (g) guarantees that passengers travel from the origin to the destination without interruption. Constraints (h)–(j) compute the arrival time and departure time between two consecutive points and Constraint (k) specifies the time window at each point. Constraints (l) and (m) ensure that the number of taxis parking at a time point does not exceed the capacity of the parking place.

Objective

The problem objective is to minimize the total travel distance:

$$F = \sum_{k=1}^{L} d(P^T_k).$$

3 | TAXI DEMAND PREDICTION

This section introduces our algorithm for predicting the number of new requests issued from some location at some future time. We analyze the request occurrence frequency based on the historical requests collected over time and space. Taxi requests can be viewed as spatiotemporal data points or events. It is assumed that (i) The occurrence frequency of an event can take a non-negative integer value in a time interval. (ii) Events are independent of other events. (iii) The arrival of an event is independent of the event prior to it. (iv) The average rate of occurrence of an event is independent of any occurrences. Under these conditions, the count variable of taxi requests can be appropriately modeled with the non-homogeneous Poisson process, which is widely used in counting processes[15]. Within a time interval $T = [t_s, t_e] \subset T$ (accurate to 1 second) and a finite time-dependent spatial domain $D(t) \subset \mathbb{R}^2$, this process depends on the arrival rate function $\lambda(t)$, where $t$ represents a location in $D(t)$ and $t \in T$ is a time point. The Poisson process is used first for learning the distribution of taxi requests and then for generating the predicted pickup points at time point $t$ at location $a$. For convenience in online applications, when all decisions must be made quickly, a spatiotemporal Poisson process can be viewed as a purely temporal Poisson process (TPP) over an area [16]. Specifically, we partition region $D$ (say, a city), into a set of $n$ non-overlapping areas $D = \{D_1, D_2, \ldots, D_n\}$. A TPP on an area $D_i$ depends on a single mathematical parameter $\lambda_{D_i}(t)$. We divide $T$ into $a$ equal time periods $\{T_j = [t_j, t_j + (t_e - t_s)/a]\}_{j=1}^{a}$ with $t_1 = t_s$. The random variable $Z_{D_i \times T_j}^{(d)}$ represents the number of requests in area $D_i \in D$ during period $T_j (j \in [1, a])$ on date $d$ of a month $(d = 1, ..., C)$. As the training data for learning the distribution of taxi requests, we take the union of all data collections $Z_{D_i \times T_j}^{(d)}$ obtained by observing each area $D_i$ during each time period $T_j$. The probability of observing $x$ requests is computed as

$$P(Z_{D_i \times T_j}^{(d)} = x; \lambda_{D_i \times T_j}) = \frac{e^{-\lambda_{D_i \times T_j}^*} (\lambda_{D_i \times T_j}^*)^x}{x!},$$

where $\lambda_{D_i \times T_j}^* = \int_{t_j}^{t_j + (t_e - t_s)/a} \lambda_{D_i}(u)du$ is the arrival rate learned through the maximum likelihood principle. That is, we try to optimize the approximate arrival rate based on the observed arrival data. Given a sequence of observations $z_1, ..., z_n$ of the Poisson variable $Z$, the rate is computed by the following maximum likelihood estimation:

$$\lambda^* = \frac{\sum_{i=1}^{n} z_i}{n}.$$

In this work, we assume that the arrival rate of the Poisson process depends on the location and time period but is independent of date. The optimal arrival rate is simply computed as

$$\lambda_{D_i \times T_j}^* = \sum_{d=1}^{C} Z_{D_i \times T_j}^{(d)}.$$

To capture the variation of taxi demands on different days, we include an extra dimension representing the seven days of a week. Let $\delta = \omega(d)$ be a day of the week corresponding to date $d$. The rate in the above model is adjusted to $\lambda_{D_i \times T_j \times \delta}$, which has three parameters: the area $D_i$, period $T_j$, and day of week $\delta$. The optimal rate is then computed as

$$\lambda_{D_i \times T_j \times \delta}^* = \frac{\sum_{d \in \delta} Z_{D_i \times T_j}^{(d)}}{|C_{\delta}|},$$

where $\delta$ is the set of days.
4 | ONLINE TAXI-SHARE ROUTING PROBLEM

In the online scenario, we propose an algorithm named OTSF-DP that solves the taxi-routing problem based on the demand prediction (described in pseudo-code in Algorithm 1). The working time of the routing system begins at \( t_\text{cur} \). Line 1 determines the union of the sets \( PP = \{ P_f \}_{f=1}^n \), where \( P_f \) is a set of predicted pickup points in the \( f \)th time period (\( \forall f = 1, a \)). Line 2 acquires a set of new requests \( R_{\text{cur}} \) sent during \([ t_{\text{cur}}, t_{\text{cur}} + \Delta T \] ). Line 5 updates the information on a taxi, such as its status and location. For each new request \( r \in R_{\text{cur}} \), Line 7 finds an appropriate taxi \( k \) and the best position \( j \) on the route. The pickup and drop-off locations associated with the executing request and the schedule and routes of the taxi are then adapted to optimize the task completion (Lines 9 and 10). This process applies an exchange operator and an Or-opt operator [17]. Finally, Line 11 determines the direction from the last drop-off point to the best parking point. The algorithm is repeated until the end of the working time.

![Algorithm 1 Proposed algorithm for online routing of taxis (the OTSF-DP algorithm)](https://example.com/algorithm1.png)

**Algorithm 1 Proposed algorithm for online routing of taxis (the OTSF-DP algorithm)**

1. \( PP \leftarrow \{ P_f \}_{f=1}^n \)
2. \( R_{\text{cur}} \leftarrow \text{UnscheduledRequests}(t_{\text{cur}}) \)
3. \( t_{\text{cur}} \leftarrow t_j + \Delta T \)
4. **while** \( t_{\text{cur}} < t_j \) **do**
5. **for** request \( r \in R_{\text{cur}} \) **do**
6. \( (k, j) \leftarrow \text{FindAppropriateTaxi}(r, t_{\text{cur}}) \)  
7. \( \text{Section 4.2} \)
8. **if** \( (k, j) \neq \perp \) **then**
9. \( \text{RequestInsertion}(r, k, j, PP) \)  
10. \( \text{Section 4.5} \)
11. \( \text{ImprovementOperator()} \)  
12. **DirectionToParkingPoint}(k) \)  
13. **Section 4.6**
14. **end if**
15. **end for**
16. \( R_{\text{cur}} \leftarrow \text{UnscheduledRequests}(t_{\text{cur}}) \)
17. \( t_{\text{cur}} \leftarrow t_{\text{cur}} + \Delta T \)
18. **end while**

4.1 | Route representation

The schedule of taxi \( k \) is represented by a sequence \( s_k = (o^k, p_1^k, ..., p_{n_k}^k, p_e^k) \), in which \( o^k \) is the departure location, \( p_c^k \) is the parking location, and \( p_1^k, ..., p_{n_k}^k \) are the pickup or drop-off points associated with the request. The length of a taxi’s plan \( l_k \) is defined as the number of request points in \( s_k \) sequence. To construct the route, we insert the road points into the sequence \( s_k \), meaning that the road points lie between the pickup and drop-off points. Let \( S_k \) be the sequences of request-related points and sub routes that taxi \( k \) must pass through. Formally, \( S_k \) is defined as \( \langle o^k, p_1^k, u_1^{k,1}, ..., p_{n_k}^k, u_{n_k}^{k,n_k}, p_e^k, p_c^k \rangle \), where \( \{ u_i^{k,i} \}_{i=1}^{n_k} \) denote the sub routes between two request-related points of taxi \( k \). Along its final sub-route \( u_{n_k}^{k,n_k} \), taxi \( k \) is completely vacant. On large-size realistic road networks, computing the shortest paths between two points is time consuming for a conventional algorithm such as Dijkstra. To overcome this disadvantage, our algorithm first decides the sequence of request-related points \( (s_k) \) using an approximate distance measure (the Manhattan distance) over the spatial grid layout of most streets in the city. The detailed route \( (S_k) \) is then determined using the actual distances on the road network before applying the shortest path algorithm. We define the following variables:

- \( \text{len}(S_k) \): length of the detailed route \( S_k \).
- \( p(S_k, i) \): the \( i \)th point of route \( S_k \).
- \( \text{first}(S_k, i) \): the first points of \( S_k \).
- \( \text{last}(S_k, i) \): the last len(\( S_k \)) - 1 + 1 points of \( S_k \).
- \( d(u, v) \): the shortest travel distance from \( u \) to \( v \) in the road network.
- \( i(u, v) \): the minimum travel time along the shortest path from \( u \) to \( v \).
- \( \text{od}(S_k, p) \): the order of point \( p \) in route \( S_k \).
- \( \text{at}(S_k, i), \text{dt}(S_k, i) \): the arrival and departure times, respectively, of \( p \) at point \( p(S_k, i) \).

4.2 | Possible positions for insertion

When the system receives a new request, our algorithm immediately updates the statuses and locations of the taxis and finds the nearest available taxi. We then insert the pickup and drop-off locations associated with the request into the best position that optimizes the obtained route without violating the constraints. Assuming that the routing system requires \( \Delta T \) time to select taxi \( k \) for a request, the algorithm updates the taxi’s route \( S_k \). The statuses and locations of taxi \( k \) are changed over the interval \([ t_{\text{cur}}, t_{\text{cur}} + \Delta T \] ). Note that the sequence of request-related locations whose arrival times preceded \( t_{\text{cur}} + \Delta T \) cannot be changed. When a taxi is serving a passenger, its route must not be changed until the passenger is dropped off. Therefore, a number of locations in the current schedule must remain fixed. These
unchanged locations are described by first($S^k, j$) in which $j$ is the maximal index in $S^k$ before the locations become fixed.

### 4.3 Route re-optimization

To optimize taxi $k$’s total travel distance, we need to reorder the points in last($S^k, j$) (where $j$ is the above-mentioned maximal index). The function $\text{opt}(o_j, (o_{j+1}, ..., o_k))$ inputs the parameters $o_{j+1}, ..., o_k \in s^k$ (the request-associated points) and $o_j$ (the road point or request point) and returns the best order of points, denoted as $(o_j, (o_{j+1}, ..., o_k^*))$, which minimizes $d(o_j, o_{j+1}^*) + \sum_{i=j+1}^k d(o_i^*, o_{i+1})$ under the problem constraints. The function returns null if all constraints are violated.

### 4.4 Route establishment

The full information of the detailed route $s^k$ (sequence of points, arrival and departure times at each point) is computed from $s^k$ by the function route($s^k$). Note that the sequence of road points between two consecutive points of $s^k$ is based on the shortest path between those points on the road network.

### 4.5 Request insertion

This subsection presents Algorithm 2, which inserts a request into the schedule of a taxi. When the system receives a new request, our algorithm immediately finds an appropriate taxi and then updates its schedule and route by inserting the pickup and drop-off locations associated with the request into the best positions. The best insertion position in this paper differs from that in [14]. In [14], a new point (pickup or drop-off point) is inserted immediately after the nearest point in last($S^k, j$). Our algorithm finds the best order of points in last($S^k, j$) (including the new points) by the route re-optimization function, which accounts for the information change during the processing time $\Delta T$. Our route re-optimization function thus intensifies the search process.

In Algorithm 2, the taxi’s new route is the combination of sub-route $S^k_1$ of the first unchanged points, sub-route $S^k_2$ of the reorganized points (including the new request-related points), and sub-route $S^k_3$ from the last drop-off point to a new parking location. Line 1 finds a fixed point in $S^k$ at index $j$. Line 2 stores the sequence of fixed points in $S^k_1$. Line 3 calls the function $\text{opt}(\cdot)$ that returns the best sequence $\text{seq}$ of the last points after index $j+1$ in $S^k$. If no sequence satisfies the time-window constraints, the new request is rejected (Lines 4 and 5). Line 9 finds the best parking space $p^k$ for taxi $k$. A parking point $p_e \in P_0$ is said to be the best parking point in time period $f \in [1, a]$ if its score, calculated as $\text{score}(f, p_e) = G(f, p_e)/H(f + 1, p_e)$, is highest during that period. Here, $G(f, p_e)$ denotes the total number of taxi requests issued at parking location $p_e$ over time span $[1, f]$ and $H(f + 1, p_e)$ is the total number of requests issued at all locations predicted by our models within radius $\theta_2$ of point $p_e$ over time span $[1, f + 1]$. The function $\text{FindBestParking}(p^k, dt(p^k), PP)$ computes the parking scores and returns the parking location as $p^k = \text{argMax}_{p \in P_0} \{\text{score}(f, p) | d(p^k, p) \leq \theta_1 \ast d^*\}$, where $d^*$ is the shortest travel distance from the last point $p^k$ to the nearest parking point, and $\theta_1$ is a rate parameter that defines the rate of actual traveling distance exceeding the direct distance from point $p^k$ to the nearest parking location.

#### Algorithm 2 RequestInsertion($r, k, j, PP$)

1: $o_k \leftarrow p(S^k, j)$
2: $S^k_1 \leftarrow \text{first}(S^k, j)$
3: $\text{seq} \leftarrow \text{opt}(o_k, (p^k_{j+1}, ..., p^k_{l}, r, r_d))$ \(\triangleright\) See Section 4.3
4: if $\text{seq} = \bot$ then
5: return $S^k$
6: else
7: $S^k_2 \leftarrow \text{route}(\text{seq})$
8: $p^k_e \leftarrow p(S^k_2, \text{len}(S^k_2))$
9: $p^k_e \leftarrow \text{FindBestParking}(p^k_e, dt(p^k_e), PP)$
10: $S^k_3 \leftarrow \text{route}(p^k_{l+1}, p^k_e)$ \(\triangleright\) See Section 4.4
11: return $S^k_1 :: S^k_2 :: S^k_3$
12: end if

### 4.6 Improvement operator

After establishing the taxis’ routes, our algorithm re-organizes the remaining requests of all taxis using an exchange improvement operator, which seeks the best exchange of the segments between two routes that most significantly reduces the total distance. The exchange operator is illustrated in Figure 2. The points in last($S^k, j$) are exchangeable with those of last($S^k, j$). As this operator does not change the feasibility of a solution, it is appropriate for exploring rich neighborhood structures in online algorithms.
4.7 Prediction-based idle taxi direction

We now propose an algorithm that directs a taxi from the last drop-off point to a pre-specified parking place under the predicted taxi demands. More precisely, we find the best sub-route for each taxi \( u_k \) without a load \( u_{lk} \)\((k = 1, L)\), as shown for \( u_{lk} \) candidates in Figure 1. In this demonstration, the driver of each candidate \( u_{lk} \) follows a sequence of road points passing through the shaded areas (i.e., a path). The best sub-route \( u_{lk} \) has the highest probability of receiving a new request while driving along that sub-route.

We assume that \( u_{lk} \) consists of a series of road points \( o_1, o_2, ..., o_m \) in the city \( D \). The areas covering these road points are compiled into a list \( W = \{D_i \} \), where \( D_i \) is an area in \( D \) such that \( u_{lk} \cap D_i \neq \emptyset \), \( \forall i = 1, y \) and the road points in area \( D_i \) are visited before those in area \( D_{i+1} \), \( \forall i = 1, y-1 \).

Now, we need to determine the probability that at least one taxi request is issued in region \( W \) during the time interval \( T = (dt(p_k^l), dt(p_k^l) + \hat{t}(p_k^l, p_k^e) + \beta) \), where \( \beta \) is the allowed additional travel time. This task is implemented as follows:

\[
P(Z_{W \times T} > 1) = (1 - P(Z_{D_i \times T_i} = 0)) + \sum_{i=2}^{y} \left( \prod_{j=1}^{i-1} P(Z_{D_j \times T_j} = 0) \right) (1 - P(Z_{D_i \times T_i} = 0)),
\]

where \( T_i \) is the travel time of the taxi to subarea \( D_i \). The details are given in Algorithm 3. Based on Equation 1, this algorithm computes the best \( u_{lk} \) for each taxi \( k \) at a given time point. Lines 1 and 2 initialize the best \( u_{lk} \) with zero probability. Line 3 collects the candidate sub-routes from the last drop-off point to the parking locations. Lines 4–13 explore the best \( u_{lk} \) with the highest probability of intercepting a request. Lines 5–7 ignore any \( u_{lk} \) that violate the travel-time condition, where \( \beta \) is the allowed time that can exceed the direct travel time from the last drop-off point to the indicated parking location.

### Algorithm 3 DirectionToParkingPoint\( (k) \)

1. \( F_{\text{best}} \leftarrow \emptyset \)
2. \( P_{\text{best}} \leftarrow 0 \)
3. \( L \leftarrow \text{List of possible sub routes from point } p_k^l \text{ to point } p_k^e \)
4. for sub route \( (o_1, ..., o_m) \) in list \( L \) do
5. \( \text{if } \hat{t}(p_k^l, o_1) + \sum_{i=2}^{m} \hat{t}(o_{i-1}, o_i) + \hat{t}(o_m, p_k^e) > \hat{t}(p_k^l, p_k^e) + \beta \) then
6. \( \text{Continue} \)
7. \( P \leftarrow \text{probability of request occurrence computed by (1)} \)
8. if \( P_{\text{best}} < P \) then
9. \( P_{\text{best}} \leftarrow P \)
10. \( F_{\text{best}} \leftarrow (o_1, ..., o_m) \)
11. end if
12. end for
13. Insert \( F_{\text{best}} \) into route \( S^k \) as the \( u_{lk} \)

## EXPERIMENTS

### 5.1 Data description

Instances in the present study were collected from the Cabspotting database (http://cabspotting.org), which records the taxi trails in the Bay Area of San Francisco. The prediction model was learned on the traces extracted from 07-2005 to 07-2006 and the algorithms were evaluated on the traces of the first nine days of 03-2010. Half of the collected requests were randomly chosen and converted into parcel requests with relaxed time windows. The call time of each taxi request was assumed as 10 minutes before the pickup time of the request. The latest pickup time was 15 min after calling. The latest drop-off time was obtained by adding the shortest travel time from the origin to the destination plus 30 min to the time call.

### 5.2 Simulation design

To evaluate the efficiencies of different routing algorithms, we constructed a simulator for monitoring the results, which can be extended to other dynamic vehicle routing problems. Implementing the OTSF algorithm, we first directed an unoccupied taxi to the nearest...
parking location. We then directed the taxi to the best parking location using the score measure in the OTSF-DP algorithm. Next, we compared the performances of the proposed algorithm, the existing DSARP algorithm proposed by [12], and its extended version DSARP-DP, which predicts information on new requests and parking location. The parameter settings are listed in Table 1. To compare our solutions with those in [12], we adopted some of the taxi fare parameters introduced in [12] (see Table 2) and calculated the profits of the solutions.

Note that people requests are served directly in our model (i.e., passengers do not share a ride). Hence, there are no discounts for passengers.

5.3 | Experimental results

The efficiency of our developed solution methods was evaluated in numerical experiments on a PC (Intel Core i7-4790 CPU @ 3.660 GHz, CPU 16 GB RAM). The OTSF, OTSF-DP, DSARP and DSARP-DP algorithms were implemented in JAVA. Two experimental scenarios were constructed: tight restrictions on parcel requests and a wide time window for parcel requests. The numbers of requests for service in both scenarios are given in Table 3.

#### 5.3.1 | Scenario 1: Tight restrictions on parcel requests

The first scenario included all taxi requests for people and parcel delivery received over nine days. The time window of a parcel request complied with the temporal constraint described in Section 5.1. The results of the four routing algorithms are compared in Table 4. The best-achieved solution in each instance is shown in boldface. In most cases, the algorithms using the predicted information for routing achieved higher profits than the algorithms not using the predicted information. Compared with OTSF, the OTSF-DP algorithm achieved total profit savings of 2520–5730 USD per day. Although the numbers of requests did not greatly differ among the algorithms, the profit increased significantly after utilizing the predicted information.

In [12], the parcel requests were assumed to be known in advance and the numbers of requests were usually lower in DSARP and DSARP-DP than in OTSF and OTSF-DP. The authors of [12] always scheduled the next six parcel requests for each taxi. Thus, the taxis' movements were non-flexible and routing detours were inevitable. If the time window of these parcel requests is strictly constrained, such requests are assigned high priority. Therefore, instead of a taxi that has completed its trip within the locality of the pickup, a taxi might arrive from a distant location. For this reason, the total profit was lower in DSARP and DSARP-DP than in our OTSF and OTSF-DP algorithms.

#### 5.3.2 | Scenario 2: Wide time window of parcel requests

For the second scenario, we selected 50% of the total number of people requests and 30% of the total number of parcel requests with randomly selected pickup times between 8:00 and 16:00. For a fair comparison, the time window of the selected parcel requests was adjusted to that of [12]. More precisely, the lower bound was uniformly and randomly drawn from 8:00 to 16:00 (accurate

<table>
<thead>
<tr>
<th>Param</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>34 parking places in the city, selected based on GoogleMaps and $c(q) = 39 \forall q \in Q$.</td>
</tr>
<tr>
<td>$L$</td>
<td>1000 taxis in total.</td>
</tr>
<tr>
<td>$P_{\text{road}}$</td>
<td>Road points of San Francisco City, collected from OpenStreetMap with 131 245 nodes and 259 792 arcs.</td>
</tr>
<tr>
<td>$\theta_1$</td>
<td>1.5 (Coefficient of maximal travel distance).</td>
</tr>
<tr>
<td>$\theta_2$</td>
<td>5 km (Radius from a parking location).</td>
</tr>
<tr>
<td>$\alpha, \beta, \Delta T$</td>
<td>96 600 (s), 30 (s).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Avg. Cost (USD/km)</th>
<th>Passenger delivery</th>
<th>Parcel delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Init Fare (USD)</td>
<td>Travel Fare (USD/km)</td>
</tr>
<tr>
<td>0.8</td>
<td>3.5</td>
<td>2.7</td>
</tr>
</tbody>
</table>

Note that people requests are served directly in our model (i.e., passengers do not share a ride). Hence, there are no discounts for passengers.
### Table 3  Number of requests for taxi services in two scenarios

<table>
<thead>
<tr>
<th>Ins</th>
<th>Scenario 1</th>
<th></th>
<th>Scenario 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#r((tp(r) = 1))</td>
<td>#r((tp(r) = 0))</td>
<td>#r((tp(r) = 1))</td>
<td>#r((tp(r) = 0))</td>
</tr>
<tr>
<td>D1</td>
<td>6167</td>
<td>6167</td>
<td>2352</td>
<td>1366</td>
</tr>
<tr>
<td>D2</td>
<td>5768</td>
<td>5768</td>
<td>2692</td>
<td>1535</td>
</tr>
<tr>
<td>D3</td>
<td>6646</td>
<td>6646</td>
<td>2784</td>
<td>1601</td>
</tr>
<tr>
<td>D4</td>
<td>6673</td>
<td>6673</td>
<td>2600</td>
<td>1517</td>
</tr>
<tr>
<td>D5</td>
<td>7210</td>
<td>7210</td>
<td>2718</td>
<td>1633</td>
</tr>
<tr>
<td>D6</td>
<td>8229</td>
<td>8229</td>
<td>2730</td>
<td>1876</td>
</tr>
<tr>
<td>D7</td>
<td>8238</td>
<td>8238</td>
<td>2760</td>
<td>2037</td>
</tr>
<tr>
<td>D8</td>
<td>5166</td>
<td>5166</td>
<td>2317</td>
<td>1334</td>
</tr>
<tr>
<td>D9</td>
<td>4147</td>
<td>4147</td>
<td>895</td>
<td>299</td>
</tr>
</tbody>
</table>

### Table 4  Routing results of the four algorithms in the first scenario

<table>
<thead>
<tr>
<th>Ins</th>
<th>#Reqs a</th>
<th>OTSF</th>
<th>OTSF-DP</th>
<th>DSARP</th>
<th>DSARP-DP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PCT b</td>
<td>Profit c</td>
<td>PCT b</td>
<td>Profit c</td>
</tr>
<tr>
<td>D1</td>
<td>12 334</td>
<td>99.71</td>
<td>71.22</td>
<td>99.91</td>
<td>74.97</td>
</tr>
<tr>
<td>D2</td>
<td>11 536</td>
<td>99.74</td>
<td>63.24</td>
<td>\textbf{100.00}</td>
<td>\textbf{67.09}</td>
</tr>
<tr>
<td>D3</td>
<td>13 292</td>
<td>99.14</td>
<td>72.88</td>
<td>\textbf{99.98}</td>
<td>\textbf{78.61}</td>
</tr>
<tr>
<td>D4</td>
<td>13 346</td>
<td>99.65</td>
<td>74.03</td>
<td>\textbf{99.94}</td>
<td>\textbf{78.89}</td>
</tr>
<tr>
<td>D5</td>
<td>14 420</td>
<td>99.62</td>
<td>80.16</td>
<td>\textbf{99.91}</td>
<td>\textbf{85.00}</td>
</tr>
<tr>
<td>D6</td>
<td>16 458</td>
<td>99.73</td>
<td>90.54</td>
<td>\textbf{99.81}</td>
<td>\textbf{94.36}</td>
</tr>
<tr>
<td>D7</td>
<td>16 476</td>
<td>\textbf{99.92}</td>
<td>89.65</td>
<td>99.90</td>
<td>\textbf{92.47}</td>
</tr>
<tr>
<td>D8</td>
<td>10 332</td>
<td>99.97</td>
<td>60.50</td>
<td>\textbf{99.98}</td>
<td>\textbf{63.50}</td>
</tr>
<tr>
<td>D9</td>
<td>8294</td>
<td>99.92</td>
<td>46.79</td>
<td>99.93</td>
<td>\textbf{49.31}</td>
</tr>
</tbody>
</table>

*Total number of requests to be executed.
Percentage of requests served.
Obtained profit (Profit × 1000 USD).

### Table 5  Routing results of the four algorithms in the second scenario

<table>
<thead>
<tr>
<th>Ins</th>
<th>#Reqs</th>
<th>OTSF</th>
<th>OTSF-DP</th>
<th>DSARP</th>
<th>DSARP-DP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PCT</td>
<td>Profit</td>
<td>PCT</td>
<td>Profit</td>
</tr>
<tr>
<td>D1</td>
<td>3718</td>
<td>99.62</td>
<td>26.31</td>
<td>\textbf{99.68}</td>
<td>\textbf{27.92}</td>
</tr>
<tr>
<td>D2</td>
<td>4227</td>
<td>99.67</td>
<td>29.41</td>
<td>\textbf{100.00}</td>
<td>\textbf{30.89}</td>
</tr>
<tr>
<td>D3</td>
<td>4385</td>
<td>97.86</td>
<td>30.15</td>
<td>\textbf{94.67}</td>
<td>\textbf{18.31}</td>
</tr>
<tr>
<td>D4</td>
<td>4117</td>
<td>99.85</td>
<td>27.95</td>
<td>\textbf{100.00}</td>
<td>\textbf{29.62}</td>
</tr>
<tr>
<td>D5</td>
<td>4351</td>
<td>99.95</td>
<td>28.00</td>
<td>\textbf{99.83}</td>
<td>\textbf{28.56}</td>
</tr>
<tr>
<td>D6</td>
<td>4606</td>
<td>99.78</td>
<td>26.91</td>
<td>\textbf{99.81}</td>
<td>\textbf{28.56}</td>
</tr>
<tr>
<td>D7</td>
<td>4797</td>
<td>99.79</td>
<td>28.00</td>
<td>\textbf{99.81}</td>
<td>\textbf{28.56}</td>
</tr>
<tr>
<td>D8</td>
<td>3651</td>
<td>\textbf{99.97}</td>
<td>7.01</td>
<td>\textbf{99.81}</td>
<td>\textbf{28.56}</td>
</tr>
<tr>
<td>D9</td>
<td>1194</td>
<td>\textbf{99.97}</td>
<td>7.01</td>
<td>\textbf{99.81}</td>
<td>\textbf{28.56}</td>
</tr>
</tbody>
</table>
to 1 min) while the upper bound was fixed at 23:59:59. Table 5 lists the percentages of requests served and the profits gained by each algorithm. The OTSF-DP algorithm continued to outperform the other algorithms. Although the wide time window advantaged the DSARP and DSARP-DP algorithms, the profits of these algorithms were exceeded by those of the OTSF and OTSF-DP algorithms. The profit differences between the OTSF-DP and OTSF algorithms ranged from 1480 to 3160 USD. Comparing the results of the first and second scenarios, we found that the algorithms using the predicted information were more efficient when the taxi demand was high, because increasing the number of taxi requests enhances the opportunity for an unloaded taxi to receive a qualified request.

5.3.3 Analysis of profit growth

Figure 3 plots the accumulated profits of the four routing algorithms during the first three of the nine days. In general, the OTSF and OTSF-DP algorithms obtained higher growth rates than the DSARP and DSARP-DP algorithms. DSARP and OTSF (without prediction capability) often assigned new requests located far from the taxi’s current location, which reduced the profit. Specially, the accumulated profits of DSARP and DSARP-DP dropped sharply over some time period of the day. However, as taxis must serve some pre-scheduled requests, the profit was stabilized at a low level after falling. This result proves that the taxi-demand prediction improves the routing direction and allows flexible movement.

Table 6 depicts the efficiencies of the algorithms using the predicted information as routing support. The total idle distance per day was significantly reduced after guiding the direction with the predicted information. The improvement rate remained stable from 9.64% to 12.76%. The average idle travel time was lower in OTSF-DP than in OTSF. As the total travel cost constitutes most of the total operation cost of a taxi company, even a relatively modest improvement in the idle travel cost can significantly benefit the company. Therefore, using the predicted information in routing can improve the operational efficiency of drivers and enhance the user experience.

5.3.4 Efficiency of traveling direction based on taxi demand prediction

Taxi requests are random events. We stipulated that a new request made in a given period was not predicted (i.e., was a failed request) if the pickup point of this request was outside the set of predicted pickup points. Figure 4 shows the average percentages of failed requests in each period over nine days. The failed requests comprised less than 6% of all requests and tended to occur from 2 a.m. to 6 a.m. The demands for both taxi rides and parcel delivery were quite low at midnight. Basing the taxi’s routing on the predicted information seems ineffective over this time period. The same inference can be reached from Figure 3, which shows non-significant differences in the cumulative profits of the algorithms around midnight.

Table 6 Efficiency of the algorithm using the predicted information

<table>
<thead>
<tr>
<th>Ins</th>
<th>OTSF</th>
<th>OTSF-DP</th>
<th>GAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg t</td>
<td>d_1</td>
<td>Avg t</td>
<td>d_2</td>
</tr>
<tr>
<td>D1</td>
<td>512</td>
<td>2.49</td>
<td>448</td>
</tr>
<tr>
<td>D2</td>
<td>565</td>
<td>2.74</td>
<td>519</td>
</tr>
<tr>
<td>D3</td>
<td>497</td>
<td>2.82</td>
<td>423</td>
</tr>
</tbody>
</table>

*aAverage idle travel time of picking up a new request.

*bTotal idle travel distance.

*cImprovement rate GAP(%) = (d_1 - d_2)/d_1.
CONCLUSION

In this work, we solved the problem of routing taxis through a road network in an offline shared people-and-parcel ride situation. First, we improved the share-a-ride model in [12] to predict parcel delivery demands based on historical requests. To this end, we derived a model that predicts the most likely parts of the road network that will receive taxi requests at some period of the day. This model utilizes a spatiotemporal Poisson process. Next, we proposed the OTSF-DP algorithm for the online routing problem. Conventional navigation systems typically provide the driver with the shortest path to the nearest parking location after completing all requests. In contrast, we suggest that drivers follow the route with highest probability of receiving a new request while traveling to the specified parking location. The OTSF-DP uses the predicted information to guide the routing. However, if such an opportunity is lacking, a new parking location is recommended to the driver. At this location, it is hoped that the taxi will soon receive a new request. The OTSF-DP algorithm aims to minimize the taxi’s idle time. Finally, we compared the performances of the proposed and previous algorithms on adapted real datasets. The experimental results showed that our algorithms provided more profits and overall benefits than the existing methods, and also reduced the idle travel distance in most of the considered instances. Currently, we are developing an integrated people-and-goods delivery system based on an intelligent rideshare concept, by which people traveling in the same direction can share a ride to maximize the usage of vacant seats.

FUNDING INFORMATION

This work has been supported by Vietnam National University, Hanoi (VNU), under Project No. QG.20.56

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

REFERENCES

AUTHOR BIOGRAPHIES

Son Nguyen Van received his MSc and BSc degrees from Vietnam National University in 2014 and 2011. He is currently pursuing a PhD degree at the Hanoi University of Science and Technology, Vietnam. He has worked as a lecturer at the Academy of Cryptography Techniques, Vietnam, since 2015. His research interests include combinatorial optimization and statistical machine learning.

Nhan Vu Thi Hong received her MSc and PhD degrees in computer science from Chungbuk National University, Cheongju, Rep. of Korea, in 2004 and 2007, respectively. She worked as a postdoctoral researcher at the Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea in 2007, Chungbuk National University in 2008, and Ohio University in Athens, USA, from 2009 to 2010. Since 2011, she has worked as a lecturer at the Faculty of Information Technology, University of Engineering and Technology, Vietnam National University. Her current research interests are data mining applications to location-based services and mobility-aware ridesharing services as well as artificial intelligence and software engineering.

Dung Pham Quang received his PhD degree in 2011 from Universite Catholique de Louvain, Belgium. He is now a lecturer at Hanoi University of Science and Technology, Vietnam. His domain expertise is combinatorial optimization, especially in transportation and logistics.

Hoai Nguyen Xuan received his PhD degree in computer science from the University of New South Wales, Australia, in 2005. He worked as a university lecturer at Le Quy Don Technical University, Vietnam, Seoul National University, Rep. of Korea, and Hanoi University, Vietnam. Currently, he is an adjunct professor of computer science at HUTECH. His research interests include genetic programming, grammar guided evolutionary learning, evolutionary computation, and statistical machine learning.

Behrouz Babaki received his master’s and PhD degrees from KU Leuven, Belgium, in 2012 and 2017, respectively. He is currently a postdoctoral researcher at Polytechnique Montreal, Canada. His research interests include the intersection of machine learning, constraint solving, and probabilistic inference.

Anton Dries received his master’s and PhD degrees from KU Leuven, Belgium, in 2006 and 2010, respectively. He was a Research Expert in the Artificial Intelligence Lab (DTAI) at the Department of Computer Science of the KU Leuven in Belgium, where his research focused on probabilistic logic programming and other declarative programming languages, machine learning, and data mining. Since 2018, he has worked at the Real Time Information Processing research group of Bell Labs.

How to cite this article: S. Nguyen Van, N. Vu Thi Hong, D. Pham Quang, H. Nguyen Xuan, B. Babaki, and A. Dries, Novel online routing algorithms for smart people-parcel taxi sharing services, ETRI Journal 44 (2022), 220–231. https://doi.org/10.4218/etrij.2021-0406