Heuristics for Electric Taxi Fleet Management at Teo Taxi

Charly Robinson La Rocca and Jean-François Cordeau

HEC Montréal, Canada

ARTICLE HISTORY
Compiled November 2, 2017

ABSTRACT
The advent of new technologies in the area of transportation has led to the creation of new business models that benefit from their capabilities. Among them, Teo Taxi, a Taxelco project, uses a unique business model: taxi drivers are employees instead of being self-employed. In addition, the company uses only electric vehicles. This shift entails new challenges regarding fleet management and process optimization. This article tackles these issues and proposes solutions to three operational problems: the dispatch, relocation and charging of electric taxis. To evaluate the performance of our algorithms, we designed a discrete-event simulator. The application models a taxi fleet and computes relative performance according to average waiting time and hourly income per taxi. We show that dispatching taxis based on the solution to an assignment problem can provide a 20% increase in income relative to the simplistic approach where requests are assigned to the nearest taxi. Furthermore, we adapt to the case of taxi fleet management a relocation strategy designed for ambulances. Finally, in order to efficiently manage battery charges, we use a strategy that involves terminal calls. This strategy stabilizes the average state of charge and eases operations overall.

KEYWORDS
Optimization, heuristics, simulation, taxi service, dispatch, location problem, charging problem, electric vehicles, routing problem.

1. Introduction

Taxi is one of the most widely used means of transportation in urban areas. Several attributes make it an indispensable service. Unlike public transportation, pickup and drop-off points are not fixed and the service is not constrained by a predefined schedule. This distinction is considerable; it is why the taxi routing problem is quite unique. As opposed to many routing problems, demand is not known with any certainty. Hence, assignments cannot be planned ahead. Heuristics are often necessary to ensure that solutions can be found in a short amount of time. Another important distinction is that taxi drivers are traditionally self-employed. This can impede efficiency because control and information are not centralized. In fact, this way of managing taxi services is criticized by many (Salanova et al. 2011). It reduces the capacity to optimize because some decisions are taken by actors who do not have information about the whole system. Decisions such as when to take breaks or where to position taxis have a significant impact on productivity.

These weaknesses prompted the development of new approaches in order to improve efficiency. This paper studies the case of Teo Taxi, a start-up located in Montreal, which proposes to consolidate property: taxi licences and vehicles are owned by the dispatcher. Taxi drivers are no longer the owners, instead they are employees and work fixed shifts. The company can thus ensure a consistent customer experience and have a greater control over its fleet. The model is uncommon in another way: taxis are electric vehicles (EV). While this reduces variable costs, autonomy management becomes a vital issue. Charging infrastructures are still scarce in most cities and the autonomy of EV is notably lower than that of gasoline vehicles (GV). Both changes carried by this model imply new operational challenges. According to our research, fleet management strategies for a centralized taxi service with EV have not yet been studied. The main goal of this paper is to present solutions for this specific kind of problem.

Our study covers three problems for which two key performance indicators (KPIs) will be tracked: hourly income per taxi and average estimated time of arrival (ETA). These problems are:

(1) Dispatching problem: "how to assign clients to taxis?" or "which taxi should we choose to serve a specific request?". When demand cannot be fully satisfied, it includes selecting which requests should be served.
(2) Autonomy management problem: "when and where should a vehicle go to charge?" Here, charging stations need to be modelled with their specific charging speed and availability criteria.
(3) Location problem: "how to relocate available taxis across a specific territory?" To cut down waiting times, available taxis should be relocated dynamically toward zones where demand is high.

As one would expect, this paper does not cover all three problems extensively. Instead, it aims to provide solutions that can adequately be built upon. Indeed, the solutions proposed in this paper are subjet to considerable constraints. First, they must be applicable in a real-world environment where access to data is limited. Because this study was done in collaboration with Teo Taxi, we based our hypotheses on data that was readily available to the company. Second, these strategies must be easy to implement. This implies that solutions must be found in real time, even with a fleet comprising hundreds of taxis. Third, they must improve fleet efficiency. Income is the basic metric used to evaluate efficiency and ETA is tracked to make sure that the quality of service is acceptable.

The remainder of this paper is organized as follows. Section 2 reviews the literature covering taxi routing problems with a focus on EV. Then, Section 3 describes the simulator that was built to model Teo Taxi's fleet and to test the algorithms. The algorithms for each problem are presented in Section 4. This is followed by computational results in Setion 5 and by the conclusion.

2. Literature review

This section explains how existing research addresses taxi fleet management problems. In particular, it contains a description of solutions to the dispatching problem, which is related to the vehicle routing problem. As explained before, this paper is concerned with a specific variant of dispatching problems where the fleet consists of a collection

of EV. Accordingly, the second part of the literature review reveals how dispatching strategies can be extended to include autonomy constraints. Lastly, the vast literature on location problems is analyzed to identify strategies applicable to taxi services.

2.1. *Dispatching problem*

The taxi dispatching problem can be seen as a special case of the more general vehicle routing problem (VRP), which consists in finding routes that must be taken by vehicles in order to serve demand effectively. The literature on the VRP is quite vast, as it covers many different cases: problems with constraints on vehicle capacity, time windows, multiple depots, multiple product types, etc. When pickup points are associated to specific drop-off points, the VRP becomes a pickup and delivery problem (Savelsbergh and Sol 1995). This is similar to the taxi dispatching problem but most of the literature considers the static variant of the problem. Pureza and Laporte (2008) propose strategies for the dynamic pickup and delivery problem with time windows (DPDPTW), where data for some of the requests are not known before planning. They reschedule planned routes to accommodate new requests. Unfortunately, their approach cannot be applied to our problem as some demand must be known in advance, which is not the case with taxis.

Most taxi dispatching strategies found in the literature assume that drivers are self-employed. In this case, the main goal is to find a trade-off between the interests of the drivers and those of clients (Kümmel et al. 2016). To achieve that, Alshamsi et al. (2009) start by defining zones. Taxis that are candidates for a ride are within zones near the pickup location. This ensures reasonable waiting times for clients. Their algorithm selects among the candidates the taxi that has waited the longest. Hasheminezhad and Bahreininejad (2010) use the concept of "energy" to establish the probability for a taxi to be assigned to a ride. It increases when a taxi is available and decreases when a ride is assigned to it. This logic allows a fair distribution of income among drivers.

The Teo Taxi model differs from the traditional one. As employees, Teo Taxi drivers have guaranteed income. Therefore, the main goal shifts. The salaries of drivers are now fixed costs and the objective is to maximize the total income generated by the fleet. Maciejewski and Nagel (2013) and Maciejewski et al. (2016) propose many strategies adapted to this case:

(1) Requests are assigned to the nearest available taxi. Researchers use different definitions for "the nearest": some use Euclidean distance, others use the Dijkstra algorithm to find the shortest route. This can be improved if traffic is taken into account (Lee et al. 2004).
(2) Requests are assigned to the nearest taxi (available or not). Queuing of requests is possible if drop-off points are known. Fleet capacity to serve demand is expected to increase.
(3) Same as before but reassignments are allowed. Because delays are possible, if at any point in time the nearest taxi changes, assignments will be adapted.
(4) Each time new information is available (new request, delays, etc.) assignments between pending requests and taxis are reconsidered. The assignment problem is similar to a minimum-cost flow problem where weights on edges are waiting times. This strategy was validated for the DPDPTW (Pureza and Laporte 2008).

2.2. *Autonomy management problem*

Models for taxi services usually neglect the autonomy management problem. This is reasonable if taxis are GV because the amount of time spent to refuel is quite small during a day of work. Traditional taxi services have self-employed drivers who take care of the issue on their own. With a fleet of EV driven by employees, autonomy management systems are critical. The number of charging stations is generally limited, which means that their capacity needs to be managed in an active way. As one would expect, the literature on the subject is scarce as the problem of managing a fleet of EV is quite new. Lu et al. (2012) propose a dispatching heuristic based on demand, charging station availability and the state of charge (SOC) of the vehicles. Whenever a request is created, only taxis that pass the reachability test are taken into account. These have enough autonomy to fulfill the request i.e., the sum of the pickup segment, drop-off segment and time to return to a charging station is less than the remaining autonomy. After that, demand and charging station availability are estimated for region L at time t where the destination of the request is located. The taxi with the lowest SOC will be chosen if demand is low and station availability is high at L. If demand is high, the taxi with the highest SOC will be chosen instead in order to allow it to complete another ride immediately following the current one. The heuristic logic changes according to the types of charging services available at L (battery charging or battery switch). To assign taxis to stations, a basic rule is used: any taxi with less than 10% remaining power is returned to a nearby charging station. The researchers compared their heuristic to a dispatching strategy that ignores charging station availability and demand. They showed a reduction of 31% for the time spent at charging stations and the number of tasks completed increased by 8%.

Zhou et al. (2015) suggest a Stackelberg model based on game theory to find solutions to the assignment problem between taxis and charging stations. The model has two steps. First, taxi drivers send requests to stations when charging seems necessary. Following this, stations respond with the price that they ask for their service. Price is modulated by three parameters: time to reach the station, cost of electricity and waiting time at the station. Researchers showed that busy stations would return high prices, which helps control the distribution of taxis in the city.

A substantial challenge is predicting the state of charge of EV. Models found in the literature study correlations with specific parameters like speed (Vaz et al. 2015) or temperature (Panday and Bansal 2015). In practice, these models are too involved to be included in a taxi fleet simulator. That is why most articles assume a constant discharge rate as the taxis moves.

2.3. *Location problem*

To the best of our knowledge, the dynamic relocation of taxis has not been addressed before. Fortunately, this problem shares a lot of similarities with the ambulance relocation problem. Gendreau et al. (2001) use a double constraints model. The first one enforces that all demands must be satisfied within a radius of r_2 minutes. The second one is applied on a proportion α of the demand that must be served within a radius of r_1 minutes ($r_2 > r_1$). The objective is to maximize the backup coverage demand, which is the fraction of the demand that can be reached by at least two vehicles within r_1. In addition, a relocation cost is considered to stabilize solutions.

The main contribution of the paper is the master-slave heuristic that can solve the model in a few seconds so it can be used in real time.

Andersson and Värbrand (2007) propose a model based on preparedness. They start by dividing the territory into zones and, for each zone j, a preparedness p_j is calculated. It is inversely proportional to the demand c_j: as demand increases, more ambulances would be necessary to cover the zone adequately. Naturally, p_j also depends on how close ambulances are to zone j. Hence, for each ambulance l that can reach zone j ($l \in L_j$), the travel time t_j^l is computed. Preparedness can then be estimated with the following equation:

$$p_j = \frac{1}{c_j} \sum_{l \in L_j} \frac{\gamma^l}{t_j^l}, \tag{1}$$

where γ^l is the contribution factor of ambulance l. A linear model is proposed to ensure that no zone is below the preparedness threshold P_{min} (6) while minimizing the maximum of relocation times t_j^k (3). In model (2)-(7), x_j^k is a binary variable taking value 1 if ambulance k is relocated to zone j and 0 otherwise.

$$\min z \tag{2}$$

$$z \geq \sum_{j \in N^k} t_j^k x_j^k \qquad \forall k \tag{3}$$

$$\sum_{j \in N^k} x_j^k \leq 1 \qquad \forall k \tag{4}$$

$$\sum_{k} \sum_{j \in N^k} x_j^k \leq M \tag{5}$$

$$\frac{1}{c_j} \sum_{l \in L_j} \frac{\gamma^l}{t_j^l(x)} \geq P_{min} \qquad \forall j \tag{6}$$

$$x_j^k \in \{0, 1\}. \tag{7}$$

Constraints (4) state that ambulances can only relocate to a subset of areas N^k that are close enough. Constraint (5) limits the total number of relocations to at most M. Simulation showed that maintaining a high level of preparedness is helpful in reducing waiting times.

Most articles mentioned in this section have adopted simulation as their main tool to benchmark the proposed models. This is not surprising because simulation offers the modularity necessary to characterize fleet management strategies. Unlike analytic models, it allows for a temporal and spatial resolution of the state variables. Simulation allows for fleet management at a microscopic scale; each taxi is modelled individually. The next section shows how we went about building our own simulation tool.

3. A taxi fleet simulator

Many options are available to us for the simulator platform. The most frequently used in the literature is MATSim (Horni and Axhausen 2016). This open source software allows for the implementation of agent-based simulation in the area of transportation. Maciejewski et al. (2016) have used it in order to characterize different dispatch rules. Nonetheless, we decided to develop our own solution, like Kümmel et al. (2016). A custom solution has the advantage of being more flexible and adapted to the problem at hand. We have more control over the basic assumptions. For example, we can assess the trade-offs that need to be made between precision and computation time.

According to terms defined by Bratley et al. (1987), our simulator is a synchronous discrete event simulator. It is synchronous, because the size of time steps remains constant in time. A non-synchronous simulator updates its state variables only when an event begins or ends. This approach is to be avoided in our case, as it does not allow for the easy identification of "wait until" events (Bratley et al. 1987), which constitute the majority of our events. In addition, programming synchronous simulators is simpler. They do, however, have the drawback of requiring more computation time. In order to better conceptualize our simulator, we chose to divide it in three blocks that are associated with distinct operations: importing and processing collected data, calculating state variables for each time step (the heart of the simulator), and processing the results of the simulation.

Simulated demand comes from Teo Taxi's historical data. Attributes available on rides are:

- Booking time: When the client ordered a taxi;
- Price: How much the client paid for the ride;
- Pickup location: Where the client asked to be picked-up;
- Pickup time: When the client was picked-up;
- Drop-off location: Where the client asked to be dropped-off;
- Drop-off time: When the client was dropped-off;
- Status: Current status of the ride ("cancelled" or "completed").

Raw data were cleaned up to avoid aberrant results. A ride was rejected if any of these criteria was met:

- Any attribute from the previous list was empty;
- Pickup or drop-off location is not in Teo Taxi zone of service;
- A segment duration is not within 0.5 to 30 minutes. For example, it is very unlikely for a client to wait more than 30 minutes to get a taxi;
- Price is not between 3.5$ and 39$. Because of the way pricing works in Montreal, prices outside this range are not possible or are very unlikely.

Before simulation starts, ride objects are loaded using cleaned up data. These objects are Armadillo matrices (Sanderson and Curtin 2016) and each column represents an attribute. This library offers some basic functions for data processing (statistical functions, sorting algorithms, etc.) which are needed during the simulation. Once demand is loaded, taxis are placed randomly into Teo Taxi's zone of service. Since our simulator is stationary, the impact of these initial positions is negligible if the simulated time horizon is long enough.

At its core, the simulation is a "for" loop that iterates over the time horizon. With

each iteration, conditions are verified and trigger fleet management algorithms. For example, if there is a customer request, the dispatch algorithm is called and a taxi is assigned to it. Following this, a second loop sweeps over each taxi to update their state variables. Algorithm 3.1 below summarizes these steps.

Algorithm 3.1 The simulator's core process

```
for all Time steps do
    if Customer request then
        Dispatch
    end if
    Assign to charging stations
    Relocate
    for all Taxis do
        Update state variables
    end for
end for
```

Because Teo Taxi uses a mobile app as an entry point to order a taxi, their dispatch flow is distinct from standard services. To illustrate this, a flow chart was built (see Figure 1). When a customer opens the app, an ETA is estimated using their current location. If waiting time is not acceptable ($ETA > ETA^*$), the user exits and creates a cancelled request (balking). If not, an order is placed and the system checks if it can be satisfied. If it can, a taxi is dispatched, but the client can choose to cancel during that time (reneging). These concepts are based on queuing theory (Pazgal and Radas 2008). By hypothesis, balking and reneging probabilities only depend on waiting time. Cancel rates were deduced from historical data but their exact values cannot be revealed for confidentiality reasons.

Routing engine

We now seek to answer the following question: what is the best way to estimate the time needed to travel from one point in the city to another? Online solutions cannot be used because of latency issues and costs. Free offline solutions exist, such as OSRM, which is an open source C++ routing engine. The latter allows us to calculate a route based on OpenStreetMap data. Computation is done locally; it does not take into account recent changes in roads or traffic. Based on our experience with this tool, OSRM systematically underestimates travel time. We compared times estimated by OSRM with historical data and we obtained an average error of 350 seconds on a route for which the average duration is 755 seconds. The relative error is considerable and is attributable to human factors. Ultimately, we had to reject this solution due to the computation time needed for each call (several hundred milliseconds). Experience shows that this performance level is not acceptable for the aim of running simulations, since hundreds of simulations are often needed to obtain statistically significant results. This is what brought us to develop a simpler solution. Travel time is estimated by a linear function whose only argument is the Euclidean distance between the point of origin and the destination. The curve was built using the gsl library (GNU Scientific Library). The linear regression was calculated on the full set of historical data.

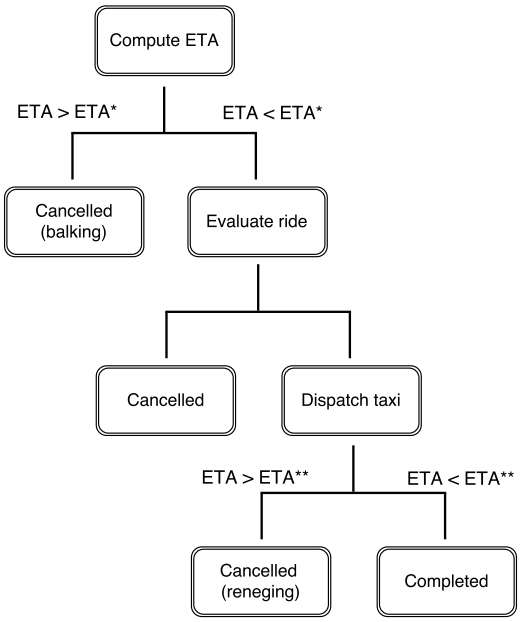


Figure 1. Flow of ordering a taxi. ETA thresholds are the maximum delay a client is willing to wait before (ETA^*) or after (ETA^{**}) ordering.

4. Heuristics

Heuristics for the dispatching problem, the autonomy management problem and the location problem are presented in this section. We describe their general approach and how they go about solving the problem at hand. When necessary, we expose the mathematical model and the parameters that support the heuristic.

4.1. *Dispatch algorithms*

This section presents three algorithms for the dispatch problem.

(1) Radius-based dispatch algorithm: the customer is assigned to the nearest available taxi at the time of the request if the latter is within a circle around the customer.
(2) Queue-based dispatch algorithm: the customer is assigned to the nearest available taxi at the time of the request (available or occupied). A customer who is assigned an occupied taxi enters into a queue.
(3) Assignment-based dispatch algorithm: customers are assigned taxis in order to minimize total waiting time.

4.1.1. Radius-based algorithm

The radius-based heuristic is fairly simple. A circle with a given radius is associated with each customer. Only the taxis inside this circle can serve them. The size of the circle depends on the hourly revenue of the ride to be dispatched: the ratio between the price displayed on the taximeter and the time necessary to complete it. This includes travel time for customer pick-up and drop-off. The value of a ride displayed on the taximeter is, by law, a function of the distance travelled by the taxi and the time that the customer is on board. To deduce it, we need to know the route followed by the taxi

and the speed on each of the segments. However, that level of detail is rarely available. The trajectory between two points is instead assumed to be a straight line. Distance and travel time are directly proportional. This implies that we need to conceive a function which evaluates the revenue of a ride as a function of the distance between two points. To do so, we use a simple linear interpolation adjusted using historical data.

The next step is to define a relationship that gives us the radius that we need to use as a function of revenue. The objective of this function is to favour the most profitable rides by associating a larger radius to them. For reasons of simplicity, we chose a linear relationship. At 27$ per hour, the radius is nil, which implies that no ride generating a revenue below this threshold will be served. The position of the threshold and the value for the slope of the curve (650m/($ per hour per taxi)) were determined by a process of trial-and-error. According to our experiments, these parameters are the ones that offered the most interesting results.

4.1.2. Queue-based algorithm

The queue-based dispatch algorithm takes into account the future availability of the occupied taxis. It is possible to do so, as the destinations of the customers are known in advance. Whether or not it is occupied, the taxi assigned to the customer is the one that can serve them within the shortest time frame. The implementation of this rule is simple: only two adjustments must be made. First, during the calculation of the ETA, one must add the time necessary to complete the current ride to the time necessary to reach the client. Second, the notion of waiting line is added for each taxi. Only taxis with an empty waiting line are considered during the dispatch process. If a taxi's waiting line is not empty, it is automatically assigned to the next task in its line when it becomes available.

4.1.3. Assignment-based algorithm

The assignment-based algorithms break the "first come, first serve" rule. Instead of assigning a taxi at the time of the call, this is done by solving an assignment problem. Each time new information becomes known, the simulator calls the COIN-OR CBC tool, which solves the assignment problem. Waiting lines and reassignments are authorized. Our objective is to maximize hourly revenue. This algorithm differs significantly from the elementary algorithm. Our model for the abandonment rate due to balking no longer applies as we do not know the ETA in advance (because reassignments are authorized). We make the assumption that with this system, no ETA is displayed to the user. Customer cancellations no longer exist. The system is now the one that decides to cancel user requests based on their profitability.

If customers can no longer cancel, a mechanism must be included in the model in order to preserve a certain level of service. In order to do so, we propose the addition of a weighting factor F to the hourly revenue as a function of waiting time: if the ETA is greater than 20 minutes, the hourly revenue for this ride is divided by 10. This mechanism naturally leads to a solution where all the customers are served within this time frame if possible. The back up solution is to add a constraint which forces the ETA to be smaller than 20 minutes. However, a fixed constraint comes

with its share of disadvantages. On the one hand, adding a constraint means that it is no longer possible to find a solution to the problem. We must therefore modify the problem (decrease the number of customers, for example) in an iterative manner until a solution is found. On the other hand, the 20-minute constraint is not always appropriate. When demand is low, the time needed for the taxi to reach the customer is of little importance. It is preferable to serve a customer at 21 minutes than to do nothing.

To obtain a classic assignment problem, the number of taxis must be equal to the number of clients in the waiting line. This is not necessarily the case. We use the strategy of Maciejewski et al. (2016) in order to address this issue. A certain number of ghost taxis or customers is added to artificially reach a balance. The revenue per assignment with a ghost entity is 0. Ghost taxis will be assigned to the least profitable customers and vice versa. If a customer is assigned a ghost taxi, the system cancels their request. A taxi assigned to a ghost customer stays immobile.

We use the following notation to formalize the problem.
Sets:

- J: set of taxis (available or with a client on board)
- I: set of clients

Parameters:

- ETA_{ij}: waiting time for client i if served by taxi j
- R_{ij}: hourly income if taxi j serves client i
- $F_{ij}(ETA_{ij})$: weight for the assignment of client i to taxi j
  - $F(ETA \leq 20 \text{ minutes}) = 1$
  - $F(ETA > 20 \text{ minutes}) = 0.1$

Decision variables :

- A_{ij}: binary variable taking value 1 if client i is assigned to taxi j, and 0 otherwise.

The problem can then be stated as follows:

$$\max \sum_{i,j} F_{ij} R_{ij} A_{ij} \tag{8}$$

subject to

$$\sum_{j} A_{ij} = 1 \qquad \forall i \tag{9}$$

$$\sum_{i} A_{ij} = 1 \qquad \forall j. \tag{10}$$

Eventually, we would like to take into account the vehicles' autonomy. In order to do so, the A_{ij} must be defined for the possible assignments; that is to say those for whom $C_{ij} + e \leq Co_j$ is satisfied, where C_{ij} is the charge level necessary to serve customer i with taxi j, Co_j is the charge level of taxi j at the time of assignment and e is a security factor.

4.2. *Autonomy management algorithms*

Modelling charge state is a complex problem. The behaviour of the batteries depends on a great number of factors such as speed, acceleration, type of vehicle, and temperature. A significant effort must be made to develop a rigorous model. Such an endeavour goes beyond the scope of our project. We opt for a modest model that uses a constant rate of discharge. The following summarizes our assumptions:

(1) Discharge rate remains constant in time. We therefore ignore the effect of movement on the discharge rate. We set its value at 15% per hour. Vehicles have an autonomy of 210 km per hour and travel at a speed of 30 km per hour on average: $100\%/(210 \text{ km}/30 \text{ km}) \approx 15\%$.
(2) Time lost at charging stations is always 15 minutes. Teo Taxi switches vehicles at charging stations, so the delay does not depend on the charge level. Fifteen minutes represents the time necessary to conduct operational activities such as the inspection of the vehicle. We assume that there is always a vehicle at the station that is available for the permutation.
(3) The charging rate at the station is linear and is proportional to its capacity. Teo Taxi owns three stations and each of them has different capacities.

The simplest heuristic to manage autonomy is the use of a charge threshold noted $T1$. When a taxi's charge level is beneath this threshold, it is deployed to the nearest charging station. This approach is reasonable, as it allows the fleet's charge level to be maintained. However, it is far from optimal. It is problematic if a significant portion of the fleet comes into service at the same time. Many taxis will reach the charge threshold at approximately the same moment. This generates a significant amount of traffic at the station. Furthermore, when a taxi reaches the threshold, it may be far from the station and will waste precious time travelling to it. Finally, the heuristic does not take into account the state of the fleet. For example, during off times (low utilization rates), it may be advantageous to deploy available taxis to available stations in order to maximize their utilization without affecting service level or profitability.

Our heuristic is an extension of the heuristic with $T1$ threshold. It comprises the notion of being called to the terminal, which happens when a station is available to accommodate a taxi. The call to the terminal is limited to taxis (available or occupied) with a charge level beneath a certain threshold $T2$ and which are located inside a circle around the station. The radius of the circle is fixed at 10 minutes. The threshold for the call to the booth is variable, it depends of the utilization rate of the fleet.

This heuristic has a weakness. It does not allow for efforts to be uniformly distributed across the stations. If many of the taxis downtown reach the $T1$ threshold at the same time, they will all be assigned the same station: the one closest to downtown. Certainly this is a plausible scenario, as a large part of the demand is concentrated in this area. We add a constraint to the heuristic which limits the number of taxis waiting at each station. The constraint uses the notion of debt, which is the time necessary to recharge all the taxis that are assigned to the booth. This debt must not be greater than 30 minutes. Let us take a concrete example: a booth is free and calls a taxi. The debt of the station therefore increases by 12 minutes (if SOC = 50%). One minute later, a second taxi reaches the T1 threshold near this station, the debt increases by

21 minutes (if SOC = 12%). The total debt is now 32 minutes (12-1+21). Hence, no taxi can be assigned to this booth for the next two minutes.

4.3. *Location algorithms*

A location algorithm seeks to dynamically re-position the fleet of taxis so that they can serve customers within a smaller time frame. The objective is to position taxis where demand is greater. As a consequence, the algorithm depends on a demand prediction model to make its decisions. Although demand prediction is an important issue for taxi fleet management, it is outside the scope of this study. Teo Taxi's demand is particularly hard to predict. First, because demand for taxis is very volatile in general: it depends heavily on the weather. Second, Teo Taxi has been in operation for less than a year. Thus, limited historical data are available. It is not yet possible to distinguish the effect of seasonality and of growth on the demand. Hence, a simplified demand prediction model was selected. It is based on a mesh which divides the territory into zones to be served. To determine the relative importance of the zones, we count the number of rides associated to each zone that we then divide by the total number of rides. It is this ratio that we use in our model to weight each one of the zones.

Our relocation heuristic is closely related to the one designed by Andersson and Värbrand (2007). Their method was simply adapted to the specific case of taxis. The first step is to find the zones that are not adequately covered. We pose that the priority P of a zone depends on two parameters: level of service LS, and demand D. A difficult-to-serve zone (low service level) will have a high priority and its value is weighted by the demand predicted for that zone. We thus use the following formula to compute the priority:

$$P = \frac{D}{LS} = D \cdot \sum_{j=1}^{nt} ETA_j. \tag{11}$$

We suppose that the service level is the inverse of the sum of travel times for the nt taxis. In other words, we want to prioritize zones that are far from all taxis. Occupied taxis are included in this sum. The ETA for an occupied taxi is the time to finish the current ride, plus the travel time toward the zone. The value of D is equivalent to the proportion of rides whose origins are in the zone. When the value of P is known for each zone, we select the n zones with the highest priority, where n is the number of free taxis. After this, all that is left to do is to assign these taxis to the priority zones in order to minimize total travel time. This whole process is done every 10 minutes: frequently enough so that solutions are always up to date.

5. Experimental results

Results for each algorithm were obtained using our simulator written in C++ on a desktop computer using a 2.6 GHz dual-core Intel Core i5 processor and 8 GB of 1600 MHz LPDDR3 memory. Simulated demand scenarios were generated randomly using Teo Taxi's historical data collected from November 2015 to September 2016. For example, to simulate a demand density of 200 requests per hour for 12 hours, the

simulator draws 2400 random requests from the database and sets booking times in order for them to appear uniformly within the time horizon. Requests were located mainly in downtown Montreal, at the center of Teo Taxi's zone of service. When a request is made, the simulator executes the dispatching algorithm to select the best vehicle and displays the corresponding ETA to the client. Using our balking heuristic, the cancellation probability can be estimated. It is then used to evaluate whether the client cancels their request or not (see Figure 1). If the request is accepted, the vehicle assigned to it starts to move on a straight line up to the pickup location and then to the destination location. For each completed request, the income generated and the waiting time is stored locally to be inspected later on to compute KPIs.

In order to structure our analysis, we must first elaborate a framework so that all the algorithms are characterized according to a standard. The elementary algorithm will be our basis for comparison. It states that a customer is assigned to the nearest available taxi at the time of the request. We seek to understand how the solutions proposed in this paper differ from it. We develop an analysis framework which defines the simulation parameters. The choice of parameters will have consequences on the scope of the results. It is important to be able to justify them, which is what we do below.

(1) Simulation time: Bratley et al. (1987) ask the questions: "How do we know when we have reached steady-state? When can we act as if we had reached steady-state?". They explain that for many, the answer is "never". The adequate analysis method for stationary simulators remains to be defined. The intuitive rule is: the simulation must be sufficiently long for the effects of the initial conditions to be negligible. We traced our performance indicators according to simulation time and saw that a 12.5 hour-long simulation allows for the steady-state to be reached.
(2) Time-step (temporal discretization): this parameter depends on the level of precision that is desired. It is set to one second. This discretization seems reasonable, as most of our events last several minutes.
(3) Number of taxis: fleet size is 50 taxis, which is the number of taxis owned by Teo Taxi in the summer of 2016.
(4) Demand: requests are chosen at random from historical data.
(5) Demand density: this is the independent variable of our study. We measure performance indicators for values between 20 and 400 requests per hour.
(6) Zone: we use Teo Taxi's service zone as of summer 2016.

The characterization method used in this paper is inspired from Bratley et al. (1987) in section 3.2 of the book. We first iterate on the demand vector and for each demand density we perform N simulations with a different random seed. The N simulations allow us to deduce the performance indicators' confidence intervals:

$$\overline{X} \pm k\sqrt{s^2/N}, \tag{12}$$

where s is the standard deviation, k is the ordinate for a probability of $(1-\alpha)/2$ in the Student distribution with $N-1$ degrees of freedom. Parameters were set so that $\alpha = 0.8$ and $N = 10$.

Two main key performance indicators (KPIs) were used to track the algorithms' performance:

- Average waiting time: it describes how long a client waits on average to receive his taxi. Only completed requests are taken into account.
- Hourly income per taxi: this is the total income generated per hour divided by the number of taxis in service.

These KPIs are measured by our simulator for different demand scenarios (between 20 and 400 requests/hour). This is done for both the tested algorithm and the reference algorithm. This allows us to measure relative performance instead of absolute performance, which is less suitable. Figures in this section display relative performance for both KPIs.

5.1. *Dispatching algorithms*

Three dispatching heuristics were tested: radius-based, queue-based and assignment-based. Figure 2 compares them. Each graph contains both KPIs (average waiting time and hourly income per taxi) and how they vary according to demand with their corresponding 80% confidence interval. We can see that the choice of heuristic does not impact any KPI when demand is very low. For every KPI and heuristic, relative performance is close to 0% when demand is around 20 requests per hour. The reason for that is simple: the closest taxi is always the best choice when there is overcapacity. All heuristics perform the same way. This is an important observation because it suggests that performance cannot be improved when there is too much overcapacity and it should be avoided. When demand increases to 200 requests per hour, the radius-based heuristic shows a reduction of waiting times because it singles out requests that are far away. As a side effect, the fleet does not complete as many rides and total income decreases by 6.5%. This happens because some requests are rejected even if they could be completed. The heuristic becomes interesting when demand is very high (5% gain in income and 16% reduction in waiting time) since capacity is the bottleneck in this situation. Hence, one can afford to dismiss the less valuable requests. In the end, the radius-based strategy should be improved to reduce the number of requests it rejects when demand is average.

The queue-based heuristic allows requests to be queued to the same vehicle. Thus, when the request is dispatched, both busy and available taxis are considered. The benefit of this is an increase in capacity, because more taxis are dispatchable. With regard to KPIs, the impact of this heuristic is marginal when demand is below the 200 request per hour mark because there is already overcapacity. On the other hand, if demand is high enough, relative income increases quite significantly, up to 12%. Cancellations that occur when no taxi is available are less likely. Thus, more tasks are completed with the same number of vehicles. Requests that would have been cancelled take longer on average to complete because they are queued. The vehicle assigned to the request has to complete its current ride before travelling to the pickup point. For clients, this means more waiting time on average, with an increase of up to 46% according to our results. This is not an unreasonable increase, because we did take into account clients' tolerance to waiting time in our model. Some clients are willing to wait to get a taxi and the queue-based heuristic allows us to serve them.

The results of the assignment-based heuristic differs from the first one: relative income increases along with demand because our algorithm favours profitable requests.

When demand is high, there are more requests to choose from which increases our capacity to select the profitable ones. In fact, from an income standpoint, this strategy is always more interesting than the elementary one (more than 20% when demand is at its peak). This is expected, as the main goal of the assignment problem is to maximize hourly income. The trade-off is the increased average waiting time. This is caused by reassignments that try to maximize income while setting aside the quality of service as long as ETA is below 20 minutes. Quality of service could be improved by adding tighter constraints on ETAs and changing these dynamically as the demand evolves. One would expect a decrease in profitability with these constraints.

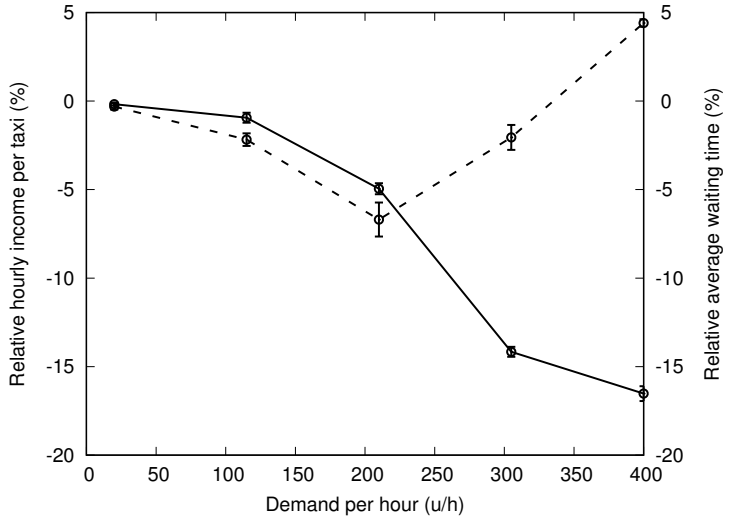
Interestingly, the assignment algorithm is comparable performance-wise to an exact algorithm that assumes deterministic demand. Knowing all the requests for a day, it is possible to enumerate all solutions for the dispatching problem and select the most profitable one. In total, there is $(nt+1)^{nc}$ possibilities (where nt is the number of taxis and nc the number of clients), because each client can be ignored or assigned to any taxi. Also, every solution that did not respect the 20-minute waiting time constraint was rejected. With $nt = 1$ and $nc = 12$, we found an hourly income of 36$ per hour per taxi with an average waiting time of 10.4 minutes. In the same conditions, the assignement-based heuristic generated 37.2$ per hour per taxi with an average waiting time of 21.3 minutes. The latter generated more income because its waiting time constraint is less strict.

5.2. *Autonomy management algorithms*

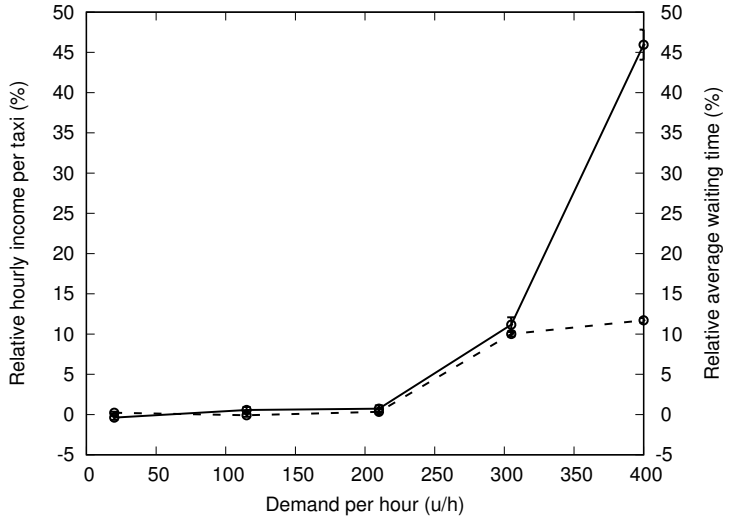
Analyzing the autonomy management problem required us to model how vehicles' autonomy evolves over time. To do so, a constant discharge rate of 15% per hour was applied on every vehicle in service. This is a good approximation for an EV moving constantly at an average speed of 30 km per hour at an ambient temperature of 20°C. In our case, these assumptions are reasonable because Teo Taxi's fleet is fairly uniform (most of them are Kia Soul EVs) and operate around the downtown area where average speed stays around 30 km per hour.

Teo Taxi uses the car-switching strategy to charge its fleet. Intead of waiting for the vehicle to charge at a station, the driver switches to a charged vehicle so he can return on the road promptly. This considerably reduces the time spent charging but it requires a surplus of vehicles standing by stations. Our model assumes that charging is linear and proportional to the station's capacity. For example, a station with a total capacity of 100 kW would be able to charge four vehicles in one hour (Kia Soul EV's capacity is around 25 kWh). At this time, Teo Taxi owns three charging stations which amount to a total charging capacity of 500 kW.

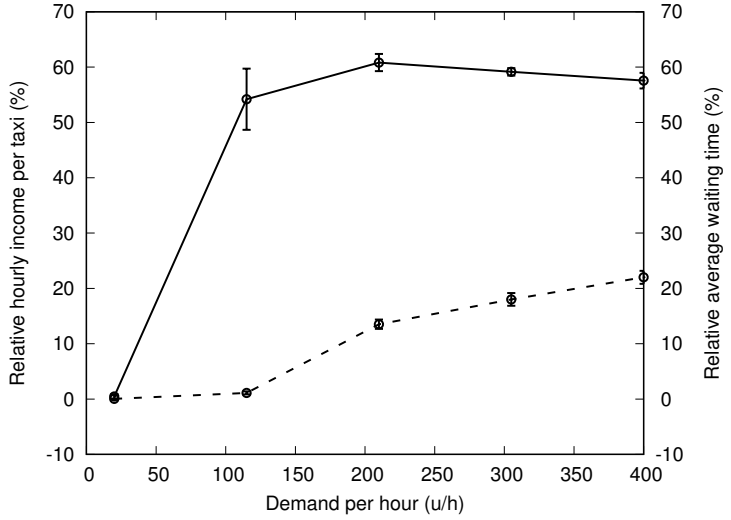
The basic heuristic for autonomy management uses one threshold $T1$ after which a taxi has to return to a station to be charged. This threshold was set to 20% in the following experiments. With 20% left, an EV still has 80 minutes of autonomy, which is enough to complete a ride and return to a station. As previously explained, this strategy has flaws that can be solved by adding a second threshold $T2$ that prescribes when a taxi should return to an available station near it. Unlike dispatching algorithms, the main focus here is not income or waiting time. Managing autonomy means dealing with the risks of outage. The measure



(a) Radius-based dispatching heuristic



(b) Queue-based dispatching heuristic



(c) Assignment-based dispatching heuristic

Figure 2. Comparison of dispatching heuristics. Solid line: relative average waiting time; dashed line: relative hourly income

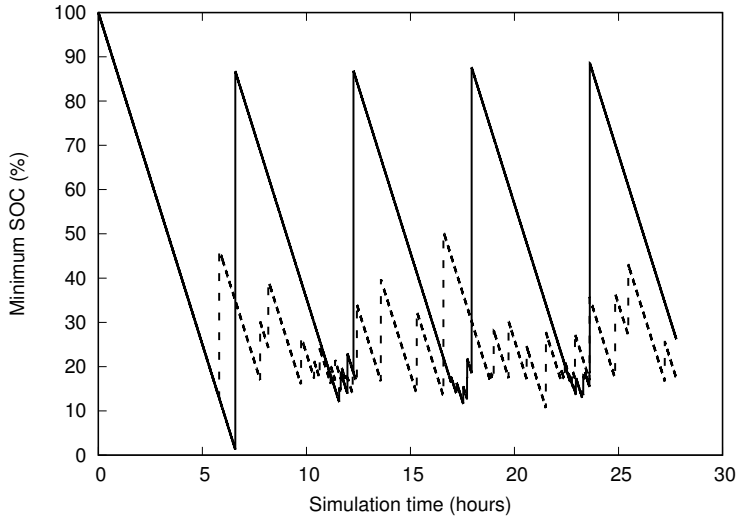


Figure 3. Comparison of autonomy management heuristics. Solid line: T1 heuristic, dashed line: T2 heuristic

used to track risk is the minimum SOC among all taxis. Figure 3 shows how this KPI evolves with time. The conclusion that stands out is how our heuristics allow for a more stable minimum SOC. Indeed, with $T1$ strategy, minimum SOC fluctuates abruptly between 12% and 87%, which is problematic. Because all taxis usually start the day with a fully charged battery, they also reach the 20% mark at the same time. When this happens, charging capacity is saturated and time is wasted waiting for charging stations to free up. Using the $T2$ strategy ensures that vehicles are charged consistently throughout the day. Minimum SOC stays around 25% so that fleet behavior is more predictable and easier to manage for fleet operators.

Furthermore, our results show that Teo Taxi's charging capacity of 500 kW is sufficient with the current configuration (a fleet size of 50 taxis and a $T1$ threshold at 20%). The minimum SOC never reached 0% during our simulations, but this might not always be the case when Teo Taxi decides to expand its operations to a wider territory with an increased fleet size. The simulator would certainly be a great tool to study the impact of this. Indeed, the usage of every station was tracked and we found out that the one closest to the heart of downtown Montreal is the most exploited one, with an average waiting time of 30 minutes. This is three times more than the second most used station. This is in line with the trends observed in Teo Taxi's operations. It is an important insight because it suggests that our methodology can be used to predict where to invest in order to reduce waiting time at charging stations.

5.3. *Location algorithms*

Relocations happen on a regular clock. On every few minutes (we tested 5 and 10 minutes), the location heuristic is applied to the fleet. First, it calculates the preparedness for every zone, which takes into account demand and the proximity of taxis. Second, available taxis are assigned to critical zones where preparedness is low. Taxis are only relocated to zones that can be reached within the relocation period.

As expected, the main benefit of relocating taxis is the reduction of waiting times,

since on average they are closer to pickup points. With a 10-minute relocation period, there is at best a decrease of 7% for the average waiting time, which corresponds to an increase of 1.6% in revenue (see Figure 4 (a)). Relocating taxis does not directly impact profitability because hourly income depends primarily on the total number of requests completed. Therefore, if there are enough taxis to satisfy every request, the total number of requests completed will be the same whether or not taxis were relocated. This is true in a world where there is no balking. The small increase that we observed in hourly income is associated with a cutback on balking. As waiting time decreases, clients are less likely to cancel their request. The gain of the heuristic is noticeable around the 200 requests per hour mark, when half of the fleet is available. There is no apparent gain when demand is high because all taxis are busy and cannot be relocated.

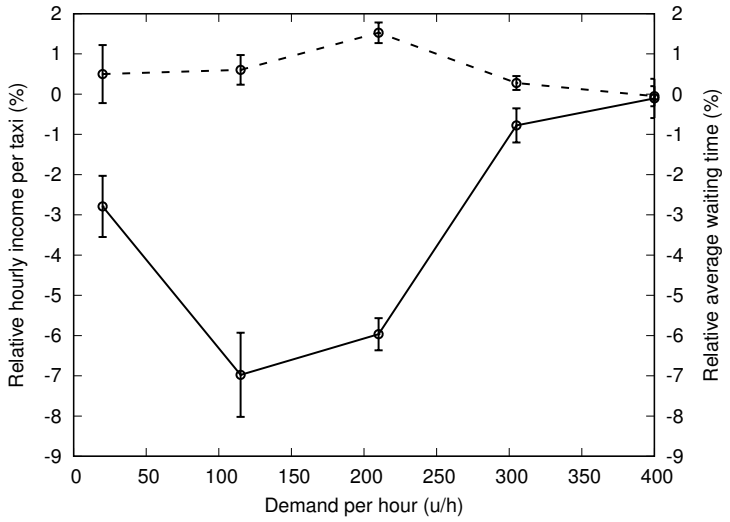
When the relocation period is reduced to 5 minutes, performance is marginally improved (about 10%) compared to the 10-minute one. Benefits are still visible at 400 requests per hour, which was not the case with the previous relocation period. Results show that one should opt for a shorter relocation period as long as it stays reasonable for drivers.

6. Conclusion

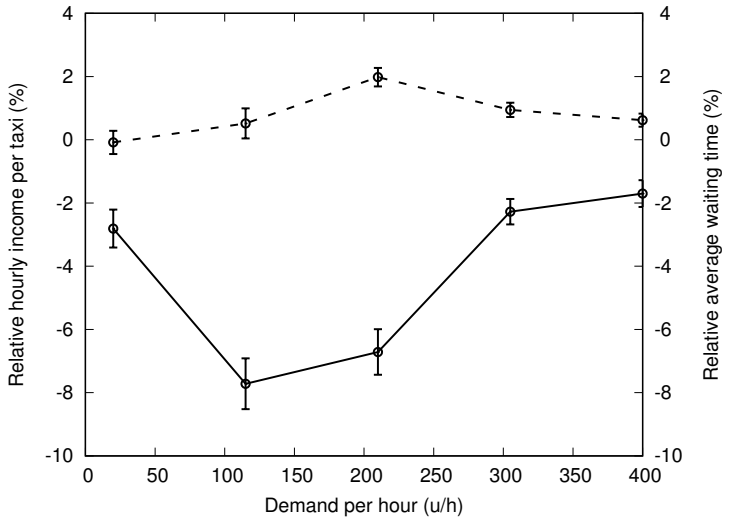
The main objective of this paper is the design of heuristics to help manage a fleet composed of electric taxis. We placed our attention on three challenging problems: the dispatching problem, the autonomy management problem and the location problem. A methodology based on simulations was used to acquire a better understanding of how different strategies can solve these problems while measuring their impact on average waiting time and hourly income per taxi. Results were promising enough to motivate Teo Taxi to implement, with some minor alterations, the radius-based and the queued-based heuristics into their dispatching system. They are actively implementing algorithms based on this research.

The fleet management problem also involves selecting the right number of taxis to operate simultaneously at any given point in time. This matter was not addressed in this research, as fleet size was fixed to 50, but our results showed that finding the right value for this parameter is vital to the efficiency of the fleet. Too many taxis and the hourly income per taxi drops, too few and there is an upsurge in the average waiting time. Furthermore, as shown, benefits of dispatching strategies are apparent when demand density is high. This suggests some interesting interactions between dispatch strategies and fleet size that could be the subject of further research. Fleet size management can be a challenge as it involves dealing with drivers' schedules, which act as limiting constraints. Generally, adapting capacity to demand variations can be quite expensive. If so, yield management strategies should be favoured as they are easier to implement in modern ordering channels and they can stabilize demand effectively.

Our research represents the first few steps in understanding the new business model of Teo Taxi and what can be done to improve its profitability. Even though Teo Taxi's model is quite unique, it shares a lot of characteristics with new sustainable modes of transportation. This research can be readily applied to transportation services that



(a) Relocation period of 10 minutes



(b) Relocation period of 5 minutes

Figure 4. Performance of location heuristic. Solid line: relative average waiting time; dashed line: relative hourly income

have to deal with on-demand requests and autonomy-constrained vehicles. There is still much to learn about these issues and we believe that this paper can support further research in this direction.

Acknowledgments

This research was supported by MITACS and by the Natural Sciences and Engineering Research Council of Canada. This support is gratefully acknowledged.

References

Alshamsi A, Abdallah S, Rahwan I. 2009. Multiagent self-organization for a taxi dispatch system. *8th International Conference on Autonomous Agents and Multiagent Systems*, Budapest, May 10-15, p. 21-28.

Andersson T, Värbrand P. 2007. Decision support tools for ambulance dispatch and relocation. Journal of the Operational Research Society. 58(2):195–201.

Bratley P, Fox BL, Schrage LE. 1987. *A guide to simulation.* 2nd ed., New York, Springer-Verlag, 397 p.

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

Hasheminezhad M, Bahreininejad A. 2010. A multi-agent taxi dispatching system. International Journal of Agent Technologies and Systems. 2(2):1–10.

Horni NK A, Axhausen K. 2016. The Multi-Agent Transport Simulation MATSim. London: Ubiquity Press. License: CC-BY 4.0.

Kümmel M, Busch F, Wang DZ. 2016. Taxi dispatching and stable marriage. Procedia Computer Science. 83:163–170.

Lee DH, Wang H, Cheu R, Teo S. 2004. Taxi dispatch system based on current demands and real-time traffic conditions. Transportation Research Record. 1882:193–200.

Lu JL, Yeh MY, Hsu YC, Yang SN, Gan CH, Chen MS. 2012. Operating electric taxi fleets: A new dispatching strategy with charging plans. *2012 IEEE International Electric Vehicle Conference,* Greenville, 4-8 mars, p. 1-8.

Maciejewski M, Bischoff J, Nagel K. 2016. An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intelligent Systems. 31(1):68–77.

Maciejewski M, Nagel K. 2013. Simulation and dynamic optimization of taxi services in MATSim. VSP Working Paper 13-05, TU Berlin, Transport Systems Planning and Transport Telematics.

Panday A, Bansal HO. 2015. Hybrid electric vehicle performance analysis under various temperature conditions. Energy Procedia. 75:1962–1967.

Pazgal AI, Radas S. 2008. Comparison of customer balking and reneging behavior to queueing theory predictions: An experimental study. Comput Oper Res. 35(8):2537–2548. Available from: http://dx.doi.org/10.1016/j.cor.2006.12.027.

Pureza V, Laporte G. 2008. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. INFOR. 46(3):165–176.

Salanova JM, Estrada M, Aifadopoulou G, Mitsakis E. 2011. A review of the modeling of taxi services. Procedia-Social and Behavioral Sciences. 20:150–161.

Sanderson C, Curtin R. 2016. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software. 1:1–26.

Savelsbergh MW, Sol M. 1995. The general pickup and delivery problem. Transportation Science. 29(1):17–29.

Vaz W, Nandi AK, Landers RG, Koylu UO. 2015. Electric vehicle range prediction for constant speed trip using multi-objective optimization. Journal of Power Sources. 275:435–446.

Zhou H, Liu C, Yang B, Guan X. 2015. Optimal dispatch of electric taxis and price making of charging stations using Stackelberg game. *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, Yokohama, November 9-12, p. 4929-4934.