Towards Online Electric Vehicle Scheduling for Mobility-On-Demand Schemes

Ioannis Gkourtzounis[1], Emmanouil S. Rigas[2(✉)], and Nick Bassiliades[2]

[1] Department of Computing, The University of Northampton, Northampton NN15PH, UK
ioannisgk@live.com
[2] Department of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
{erigas,nbassili}@csd.auth.gr

Abstract. We study a setting where electric vehicles (EVs) can be hired to drive from pick-up to drop-off stations in a mobility-on-demand (MoD) scheme. Each point in the MoD scheme is equipped with battery charge facility to cope with the EVs' limited range. Customer-agents announce their trip requests over time, and the goal for the system is to maximize the number of them that are serviced. In this vein, we propose two scheduling algorithms for assigning EVs to agents. The first one is efficient for short term reservations, while the second for both short and long term ones. While evaluating our algorithms in a setting using real data on MoD locations, we observe that the long term algorithm achieves on average 2.08% higher customer satisfaction and 2.87% higher vehicle utilization compared to the short term one for 120 trip requests, but with 17.8% higher execution time. Moreover, we propose a software package that allows for efficient management of a MoD scheme from the side of a company, and easy trip requests for customers.

Keywords: Electric vehicles · Mobility on demand · Scheduling · Demand response · Software

1 Introduction

In a world where over 60% of the total population will be living in, or around cities, the current personal transportation model is not viable as it is based almost entirely on privately owned internal combustion engined vehicles. These vehicles cause high air and sound pollution, and suffer from low utilization rates [1]. One of the key elements of the vision of future Smart Cities is the development of Mobility-on-Demand (MoD) systems, especially ones using fleets of Electric Vehicles (EVs) [2]. Such vehicles emit no tailpipe pollutants and, once powered by electricity produced from renewable sources, they can play an important role towards the transition to a new and sustainable transportation era.

M. Slavkovik (Ed.): EUMAS 2018, LNAI 11450, pp. 94–108, 2019.
https://doi.org/10.1007/978-3-030-14174-5_7

Most of the deployed MoD schemes use normal cars. However, EVs present new challenges for MoD schemes. For example, EVs have a limited range that requires them to charge regularly their battery when they stop. Moreover, if such MoD schemes are to become popular, it will be important to ensure that charging capacity is managed and scheduled to allow for the maximum number of consumer requests to be serviced across a large geographical area. In this context, Pavone et al. have developed mathematical programming-based rebalancing mechanisms for deciding on the relocation of vehicles to restore imbalances across a MoD network, either using robotic autonomous driving vehicles [3], or human drivers [4], while Smith et al. [5] use mathematical programming to optimally route such rebalancing drivers. Moreover, Carpenter et al. [6] propose solutions for the optimal sizing of shared vehicle pools. However, in all these works internal combustion engine-based vehicles are assumed and hence do not account for the limited range of EVs and how to balance the number of pending requests at specific nodes across the network while serving the maximum number of users. In contrast, [7] consider on-demand car rental systems for public transportation. To address the unbalanced demand across stations and maximise the operator's revenue, they adjust the prices between origin and destination stations depending on their current occupancy, probabilistic information about the customers' valuations and estimated relocation costs. Using real data from an existing on-demand mobility system in a French city, they show that their mechanisms achieve an up to 64% increase in revenue for the operator and at the same time up to 36% fewer relocations. In addition, Rigas et al. [8] use mathematical programming techniques and heuristic algorithms to schedule EVs in a MoD scheme taking into consideration the limited range of EVs and the need to charge their batteries. They goal of the system is to maximize serviced customers.

In this paper, we step upon the work of Rigas et al. [8], and we solve the problem of assigning EVs to customers online. In so doing, we propose two scheduling algorithms for the EV-to-customer assignment problem aiming to maximize the number of serviced customers. The first one is shown to be efficient for short term bookings, while the second for both short and long term ones. Both algorithms are evaluated in a setting using real data regarding MoD locations in Bristol, UK. Moreover, we propose a software package which consists of a web platform that supports the efficient monitoring and management of a MoD scheme from the side of a company, and a mobile application for easy trip requests for customers.

The rest of the paper is organized as follows: Sect. 2 presents a mathematical formulation of the problem, while Sect. 3 describes the scheduling algorithms. Section 4 presents the software package for the management of the MoD scheme and Sect. 5 contains the evaluation. Finally, Sect. 6 concludes and presents ideas for future work.

2 Problem Formulation

We study a MoD setting where customer-agents $k \in K$, announce their intentions to drive between pairs of locations over time. Each time a new request is received

by the MoD company (we assume a single MoD company to exist), it applies an algorithm that schedules an available EV, if such an EV exists, to drive across the set of requested locations. In assigning EVs to trips, the MoD company aims to maximize the number of agents that will be serviced. We assume that EVs have a limited driving range which requires them to have their battery charged at the stops that form part of the MoD scheme.

In more detail, we denote a set of EVs $a \in A$ and a set of locations $l \in L$ which are pick-up and drop-off stations, where each $l \in L$ has a maximum capacity $c_l \in N$. We consider a set of discrete time points $T \subset \mathbb{R}, t \in T$, where time is global for the system and the same for all EVs. Moreover, we have a set of tasks $i \in \Delta$ where each task is denoted by a tuple $p_i = < l_i^{start}, l_i^{end}, t_i^{start}, \tau_i, b_i >$. l_i^{start} and l_i^{end} are the start and end locations of the task, t_i is the starting time point of the task, while τ_i is its travel time (each task has also an end time $t_i^{end} = t_i^{start} + \tau_i$), and b_i is the energy cost of the task. Each agent has a valuation $v_k(i) = 1$ for executing the requested task i and $v_k(i') = 0$ for any other task $i' \neq i$. Note that a task is a trip taking place at a particular point in time. Also, note that one-way rental is assumed, and therefore, start and end locations of a task are always different. Moreover, we assume that customers drive the cars between start and end locations without stopping or parking them during the trip. One-way rental introduces significant flexibility for users, but management complexities (e.g., complex decision making in choosing which customers to service, and high importance of the initial location of EVs) [9]. Henceforth, index a stands for EVs, l for locations, t for time points and i for tasks.

Each EV a has a current location at time point t, denoted as $l_{a,t}$, and this location changes only each time a executes one task. Here, we assume that at time point $t = 0$ all EVs are at their initial locations $l_{a,t=0}^{initial} \in L$, and that their operation starts at time point $t \geq 1$. Moreover, each a has a current battery level $b_{a,t} \in N$, a consumption rate con_a and therefore, a current driving range in terms of time $\tau_{a,t} = [b_{a,t}/con_a] \in N$. Now, for a task i to be accomplished, at least one EV a must be at location l_i^{start} at time point t_i, having enough energy to execute the task (i.e., $b_{a,t} > b_i$). We also define binary variable $prk_{t,a,l} \in \{0, 1\}$ to capture the location where each EV is parked at each time point, binary variable $\epsilon_{a,i,t} \in \{0, 1\}$ denoting whether EV a is executing task i at time t, and binary variable $\delta_i \in \{0, 1\}$ denoting whether task i is executed or not. At any t, each EV should either be parked at exactly one location, or travelling between exactly one pair of locations. In the next section we present the decision making algorithms.

3 Scheduling Algorithms

In this section, we study the scheduling of EVs to customers, based on their requests and EV availability across the set of stations. In this vein we develop two decision making algorithms, one for short-term bookings and another for long-term ones.

3.1 Short Mode Algorithm

The Short mode algorithm (see Algorithm 1) receives as input trip requests from agents to drive an EV between stations l_i^{start} and l_i^{end} starting at time point t_i^{start} (t^{cur} defines the current time point). Based on these data, the algorithm calculates the duration τ_i of the task and the energy demand b_i (line 1). In this way the tuple p_i that describes a task i is completed. Next, the vehicles that are currently at the start station or are travelling to it and will arrive in $t < t_i^{start}$ (i.e., $\forall a : prk_{t_i^{start}-1,a,l_i^{start}} = 1$) are added to the set $candidateEV \subseteq A$ (lines 2–4). If $candidateEV = \emptyset$, then task i cannot be executed. Otherwise, we check if the candidate vehicles have future routes and we add the vehicles without future routes, to set $betterCandidateEV \subseteq candidateEV$ (lines 5–10).

Algorithm 1. EVs Scheduling Algorithm - Short mode.

Require: i and A and L and T and t^{cur} and $\forall a, b_{a,t^{cur}}$ and $\forall a, t, l$ $\epsilon_{a,t,l}$ and $\forall a, t, l$ and $prk_{a,t,l}$

1: Calculate b_i and τ_i
2: **for** $\forall a \in A$ **do**
3: **if** $prk_{t_i^{start}-1,a,l_i^{start}} = 1$ **then**
4: Assign a to $candidateEV$
5: **if** $candidateEV = \emptyset$ **then**
6: $\delta_i = 0$
7: **else**
8: **for** $\forall a \in candidateEV$ **do**
9: **if** $\forall i, t > t_i^{start} + \tau_i$, $\epsilon_{a,i,t} = 0$ **then**
10: Assign a to $betterCandidateEV$
11: **if** $betterCandidateEV \neq \emptyset$ **then**
12: **for** $\forall a \in betterCandidateEV$ **do**
13: Calculate battery charge $b_{a,t_i^{start}-1}$
14: **if** $b_{a,t_i^{start}-1} - b_i > 0$ **then**
15: Assign a to $bestCandidateEV$
16: Sort $bestCandidateEV$ based on remaining energy after executing task i
17: Assign first vehicle a to task i and set $\delta_i = 1$
18: **else**
19: $\delta_i = 0$
return δ_i, ϵ, prk

If $betterCandidateEV = \emptyset$, then the task is not executed. Otherwise, we calculate and set the future charge (we use the term "charge" for "the current state of charge level") for each vehicle by subtracting the energy cost of the task b_i from the charge level of each vehicle, $b_{a,t_i^{start}-1}$. We sort the vehicles by future charge in descending order and if the future charge of a vehicle is greater than zero, we add it to set $bestCandidateEV \subseteq betterCandidateEV$. We assume that for $\forall a, t, l : prk_{a,t,l} = 1$ a is charging its battery, unless it is fully charged. Now, the $bestCandidateEV$ contains all the vehicles that are suitable to execute task i. The first vehicle in the list has the maximum charge level, so it is the

best candidate vehicle. Thus, we assign it to task i (i.e., $\forall t : t >= t_i^{start} \text{and} t < t_i^{start} + \tau_i, \epsilon_{a,i,t} = 1$ and $\delta_i = 1$). If $bestCandidateEV = \emptyset$, the task is not executed (lines 11–19).

In summary, the Short mode algorithm gets the vehicles that currently are, or finish in the start station before start time of the task and adds the vehicles without future routes to a list. We calculate the charge cost for the request trip, get the vehicles with enough charge and assign the best vehicle to the new route.[1]

The Short mode algorithm has a relatively simple implementation, but its biggest drawback is that if a vehicle is assigned for a route that starts for example, after 6 h, this vehicle can not be used for another route for this period of time. One way to tackle this problem is to restrict the minutes in the future, that users are allowed to request a trip.

3.2 Long Mode Algorithm

A restriction that does not allow users to request vehicles at any time during the day, may not be a viable solution. In order to overcome this problem, we developed and implemented the Long mode algorithm that applies a one-step look ahead technique to reschedule EVs to other tasks in order to increase vehicle utilization and customer satisfaction.

The Long mode algorithm (see Algorithm 2) receives trip requests and the *candidateEV* set is populated in the same way as in the short mode algorithm (lines 1–4). Next, we get the vehicles without future routes and the vehicles with one future route and add them to $betterCandidateEV \subseteq candidateEV$ and $betterCandidateEVWithLateRoutes \subseteq candidateEV$ respectively (lines 5–12). We calculate the future charge $b_{i,t_i^{start}}$ of the EVs in the same way as described in the previous section and sort them by future charge in descending order. We subtract the task's energy cost b_i from $b_{i,t_i^{start}}$ and if the value is greater than zero, we add it to the $bestCandidateEV$. Now, the $bestCandidateEV$ contains all the vehicles that are suitable for the requested trip. The first vehicle in the list has the maximum charge level, so it is the best candidate vehicle. Thus, we assign this vehicle to execute task i. If there are no vehicles in the $bestCandidateEV$, the first part of the algorithm which is actually the same to the short mode one, ends (lines 13–19).

At this point, there are no vehicles without future routes and with enough charge level to execute task i. Thus, we need to find a candidate vehicle a' with a future task i', assign this task to a substitute vehicle a and then assign the vehicle a' to task i (i.e., $a \rightarrow i'$, $a' \rightarrow i$). First, we check the task of the vehicles in the $betterCandidateEVWithLateRoutes$. If this task starts later than the start time plus the trip duration of the user request, we need to find a substitute vehicle. We add the vehicles that are currently in the start station and the vehicles that finish at the start station before the start time of task i', to the $substituteEV \subseteq betterCandidateEVWithLateRoutes$. Note that $t_{i'}^{start} > t_i^{start}$ and for this reason $candidateEV \subseteq substituteEV$ (lines 20–25).

[1] A flowchart for the Short mode is available at https://goo.gl/dxpfer.

Algorithm 2. EVs Scheduling Algorithm - Long mode.

Require: i and A and L and T and t^{cur} and $\forall a, b_{a,t^{cur}}$ and $\forall a,t,l\ \epsilon_{a,t,l}$ and $\forall a,t,l$ and $prk_{a,t,l}$

Calculate b_i and τ_i
for $\forall a \in A$ **do**
  if $prk_{t_i^{start}-1,a,l_i^{start}} = 1$ **then**
    Assign a to $candidateEV$
if $candidateEV = \emptyset$ **then**
  $\delta_i = 0$
else
  for $\forall a \in candidateEV$ **do**
    if $\forall i, t > t_i^{start} + \tau_i$, $\epsilon_{a,i,t} = 0$ **then**
      Assign a to $betterCandidateEV$
    else if $\forall i, \sum_{t:t>t_i^{start}+\tau_i}(|\epsilon_{a,i,t} - \epsilon_{a,i,t-1}| = 2)$ **then**
      Assign a to $betterCandidateEVWithLateRoutes$
if $betterCandidateEV \neq \emptyset$ **then**
  for $\forall a \in betterCandidateEV$ **do**
    Calculate battery charge $b_{a,t_i^{start}-1}$
    if $b_{a,t_i^{start}-1} - b_i > 0$ **then**
      Assign a to $bestCandidateEV$
  Sort $bestCandidateEV$ based on remaining energy after executing task i
  Assign first vehicle a to task i and set $\delta_i = 1$
else
  for $\forall a' \in betterCandidateEVWithLateRoutes$ **do**
    For the task i', vehicle a' is assigned to:
    for $\forall a \in A$ **do**
      if $prk_{t_{i'}^{start}-1,a,l_{i'}^{start}} = 1$ **then**
        Assign a to $substituteEV$
    for $\forall a \in substituteEV$ **do**
      if $\forall i, t > t_{i'}^{start} + \tau_{i'}$, $\epsilon_{a,i,t} = 0$ **then**
        Assign a to $substitutesWithoutRoute$
    if $substitutesWithoutRoute \neq \emptyset$ **then**
      for $\forall a \in substitutesWithoutRoute$ **do**
        Calculate battery charge $b_{a,t_{i'}^{start}-1}$
        if $b_{a,t_{i'}^{start}-1} - b_{i'} > 0$ **then**
          Assign a to $bestCandidatesubstituteEV$
      Sort $bestCandidatesubstituteEV$ based on remaining energy after executing task i'
      Assign a to i' and a' to i and set $\delta_i = 1$
      Break for loop
    else
      $\delta_i = 0$
return δ_i, ϵ, prk

We add the substitute EVs without future routes to the $substitutes\ WithoutRoute \subseteq substituteEV$ and we sort them by future charge in descending order. If the substitute vehicle of the best candidate vehicle

has enough charge for the future trip of the candidate vehicle, we assign the substitute vehicle to the candidate vehicle's future route and EV $a \in betterCandidateEVWithLateRoutes$ is assigned to task i and $\delta_i = 1$ (lines 26–38).

In summary the Long mode algorithm gets the vehicles in the start station without future routes and those with one future route and adds them to lists. If there are vehicles with enough charge in the first list, we assign the best vehicle to the new route, else we search for a substitute vehicle to replace the future route of a vehicle in the second list. We get substitute vehicles without future routes and with enough charge, and we assign the best substitute to the future route of the best vehicle in the second list. Now, the best vehicle in the second list is no longer "locked/assigned" and we assign it to the new route.[2] The differences between the two algorithms are summarized in Fig. 1.

In the next section, we present a software package that integrates the previously described algorithms and provides an efficient user interface for MoD companies to manage their fleet, and customers to request trips.

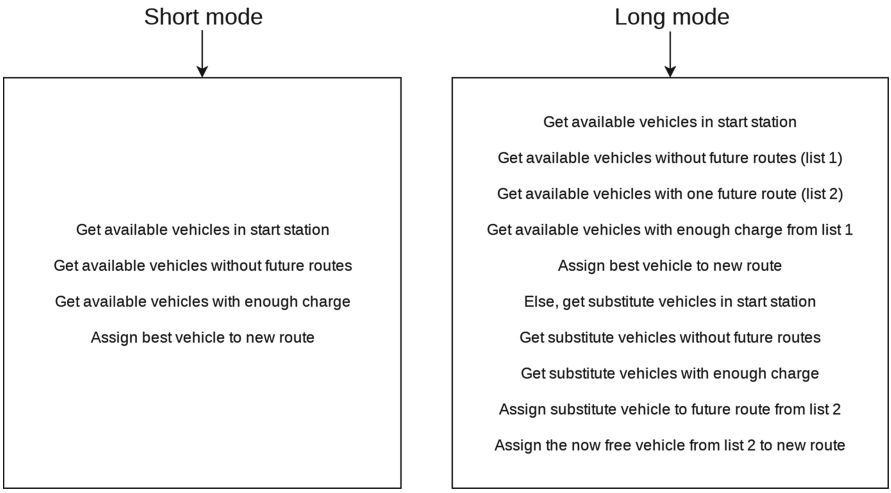

Fig. 1. Comparing Short mode and Long mode algorithms functionalities.

4 MOD Software Package

Apart from the scheduling algorithms, we designed and developed a fully functional software package (see Fig. 2), for a MoD company to monitor the state and locations of their EVs, and for the customers to make trip requests and bookings.[3]

[2] A flowchart for the Long mode is available at https://goo.gl/zFdWvh (part 1) and https://goo.gl/WZHrRc (part 2).

[3] A video demo is available at https://youtu.be/flyixErIE-A.

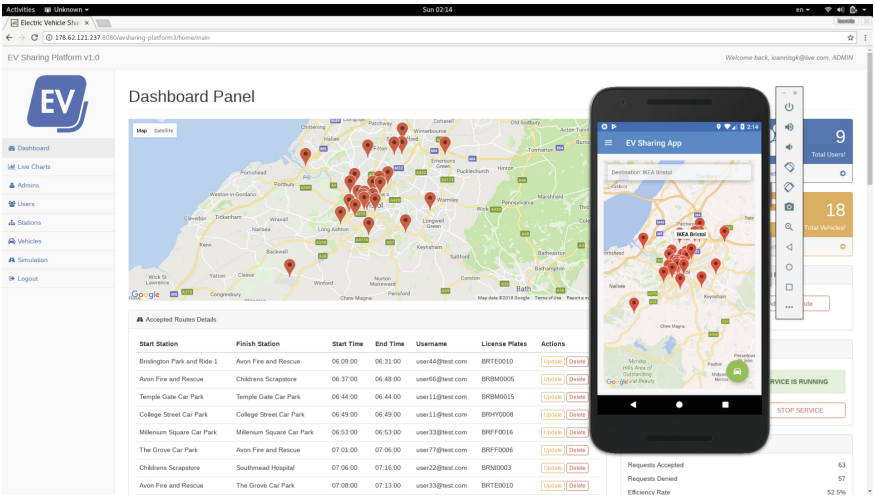

Fig. 2. Web platform and mobile application screenshot.

The web platform is installed on a server and it consists of the web pages presented to the administrators and a MySQL database to manage all the data. Administrators interact with the platform using web pages designed for specific functionalities. Thus, the interface allows them to login, upload a map with stations, upload vehicles' details and manage administrator and user accounts, stations, vehicles and routes. All operations submitted through the interface of the pages are implemented as transactions in the MySQL database. From the customer perspective, the mobile application lets users register, login, access their profile, their request history, select stations and request a vehicle for a trip. Creating an account and accessing account and station details, require access to the MySQL database on the server (Fig. 3).

Information exchange with the database is done via a RESTful web service that authenticates users and allows them to access, create and edit data on the database. Messages for vehicle requests use WebSockets, so the server listens to a specific port for incoming TCP messages. Those messages are encrypted and sent from the mobile device to the server through a TCP connection, making the communication safe and reliable. Then, the platform applies one of the algorithms presented in the previous section to accept or deny the user request. The status of the request (accepted or denied) will be sent back to the mobile application as an encrypted message through the TCP connection and the user can see if his request is accepted by the system.

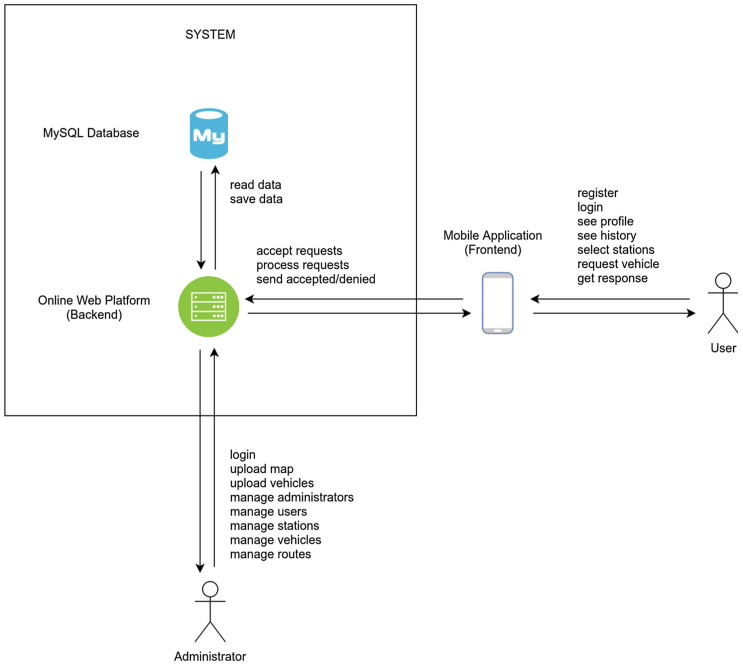

Fig. 3. System architecture with web platform, database and mobile app.

4.1 Web Platform

The pages of the web platform[4] let the administrators perform the functionalities related to managing user accounts, stations, vehicles and routes, as discussed earlier. The login page allows administrators to login with their username and password. On successful login, they are redirected to the Dashboard Panel. The Dashboard Panel shows route data and contains buttons to update, delete or add a new route. The stations are shown as pins on a Google map and buttons allow administrators to start or stop the service.

The Manage Admins and Manage Users pages show administrator and user data respectively, and contain buttons to update their existing details, delete them or add new administrators and users. The Manage Stations and Manage Vehicles pages offer the same functionalities, but they also allow administrators to upload stations and vehicle details from files in XML format. For each object there are two secondary pages, the Add New page and the Update page to provide secondary functionalities, such as adding and updating objects.

The Graphical User Interface (GUI) of the web platform consists of elements that are similar and shared between the pages. The navigation options link to the most important pages: Dashboard, Live Charts, Admins, Users, Stations, Vehicles, Simulation and Logout.

[4] https://github.com/ioannisgk/evsharing-platform3-release.

4.2 Mobile Application

The Android mobile application[5] consists of a set of screens: The Main Login screen allows users to login with their username and password. If they do not have an account, they can click the Register button to create a new account with their details. On successful login, they are redirected to My Profile screen. The Profile screen shows their details and contains buttons to update them and to move on to the next screens, such as the Open Map and User History.

The Open Map screen initiates the first step for requesting a vehicle and the map is shown with the stations as pins. Users can select the start and finish stations and on the second step, on the Request screen they can select a specific time and send the request to the platform. The platform will process all current data and decide whether to accept or deny the request. The mobile application gets the status of the request from the server and saves it to the device. The User History screen contains the history of requests and the Settings screen allows users to change the application settings.

The GUI of the mobile application consists of elements that are similar and shared between the screens. The navigation options link to the most important screens: My Profile, Open Map, Request Vehicle, User History, Settings and Logout. The navigation menu is hidden when the user touches the rest area of the application.

5 Testing and Evaluation

In this section, we present the evaluation of the scheduling algorithms. In so doing, we use real data on locations of pick-up and drop-off stations and realistic data regarding trip requests. We evaluate the algorithms based on a set of criteria: (1) the maximum number of route requests in a day, (2) the maximum number of minutes that a user is allowed to request a vehicle in the future, (3) the start times density factor, that essentially means how "close" or how "further away" in time, the start times of the route requests are, and (4) the number of available EVs. We generated test cases with different configurations and assigned them to the Short mode and Long mode algorithms.

Since we aim to maximize the number of customers being served in the MoD scheme, we apply all those different criteria in order to better evaluate the system under different conditions. The criteria 1 and 4 deal with the number of requests and the number of vehicles respectively, while criteria 3 and 4 focus on different requests times, as the problem is also directly associated with when the customers post their requests for routes.

Charging stations coordinates were collected for Bristol, UK as it is a strong candidate city to host a MoD scheme. We selected 27 stations that are also EV charging stations today[6] and categorized them in 5 traffic levels in order to resemble realistic travel conditions. We assume that the stations nearest to the

[5] https://github.com/ioannisgk/evsharing-app-release.

[6] Charging stations data were collected from https://goo.gl/pWXFm6.

center of the city have a higher number of requests, so the initial locations of the vehicles are at the stations closer to the city center, with a 100% charge level.

We developed a "route requests" generation tool that generates requests with random request times, start stations and finish stations. This tool helped us simulate a large number of vehicles requests from users with many different settings. We generated more than 400 test cases with different trip requests and used them as input to the MoD software. The platform now uses both the Short mode and Long mode algorithms to determine whether to accept or deny these requests. The algorithms also need to calculate the travel time and energy consumption of an EV during a trip, so we first calculate the average theoretical speed of the vehicle, depending on a base speed and the traffic level of the stations. Then, we compute the distance between the stations[7] and the trip duration in minutes. Energy consumption can be calculated by multiplying the duration with a base charge cost per minute.

With a total of 54 EVs, 60 and 120 maximum requests per day, we ran simulations with 60–600 min between the time the request is communicated to the MoD company and the start time of the trip, and different values of the requests allocation density within a day. The last factor shows how "concentrated" or close, the start times of the requests are, and we tested with 5 different density factors (i.e., the lower the value the more uniformly distributed the requests are).

First, we tested with 60 maximum requests per day. The Short mode produced an efficiency rate (the percentage of the requests that get accepted by the system and become routes) from 76.00%–86.67% and the Long mode from 77.00%–87.67% and it performed better with an average gain of 1.40%. When testing with 120 maximum requests per day, the Short mode produced efficiency rate from 57.17%–78.17% and the Long mode from 57.67%–80.83% and it performed better with an average gain of 2.08%. Since the more trip requests are accepted by the system the more customers are satisfied, we can directly connect the average gain of the efficiency rate of an algorithm, with the customer satisfaction levels. In other words, the Long mode algorithm achieves on average 2.08% higher agent satisfaction.

We also recorded the total minutes travelled by EVs and found that the gain of the Long mode algorithm translates to 8.88 more minutes of travelling time for every 60 requests, and 32.52 more minutes for every 120 requests for routes. In terms of vehicle utilization, the Long mode achieves a 2.87% higher average rate than the Short mode in 120 requests per day. However, the Short mode algorithm due to its simpler design achieves 17.8% lower execution times.

Looking closer at the charts in Figs. 4, 5 and 6, we can observe the following: (1) the Long mode algorithm performs better than the Short mode in all tests, (2) the Long mode offers more gain in the efficiency of the system when the number of requests increase, (3) when the time between the start times of the requests increases, the efficiency increases, (4) the overall efficiency seems not to be influenced by the requests allocation density within the day, (5) when the number of requests within the day increases, the efficiency of the system drops.

[7] Distance calculation using latitude and longitude https://goo.gl/3bDKuT.

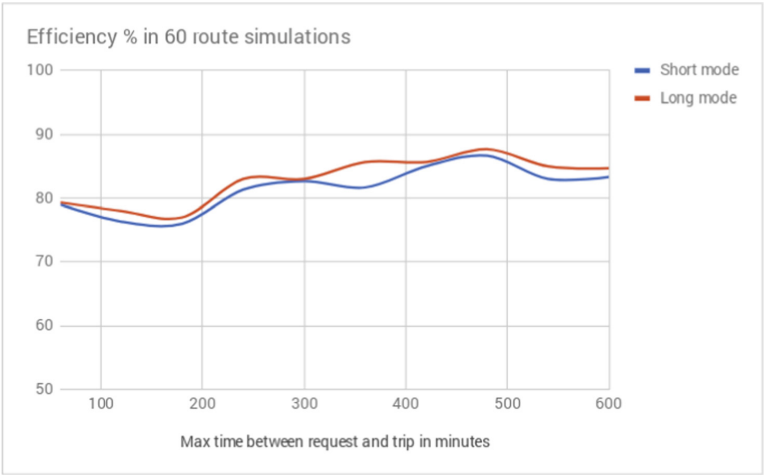

Fig. 4. Efficiency rate and max time between request and trip, when the system handles 60 requests and it is equipped with 54 EVs.

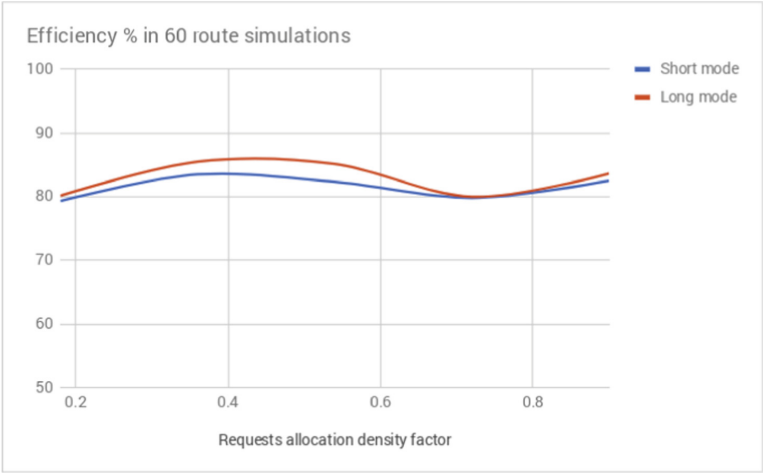

Fig. 5. Efficiency rate and requests allocation density, when the system handles 60 requests and it is equipped with 54 EVs.

Finally, we tested the system with different numbers of vehicles, as this is a very important factor for any MoD scheme company due to the high cost of EVs. The chart in Fig. 7 shows that the average gain of the Long mode algorithm is higher when using more vehicles and when the number of requests per day increase. When using 18, 27, 36, 45 and 54 EVs in 60 requests per day, the gain is 1.17%, 1.27%, 1.70%, 1.47% and 1.40% respectively. In the case of 120 requests per day, we have 1.83%, 1.93%, 2.27%, 2.05% and 2.08% gain. We conclude that

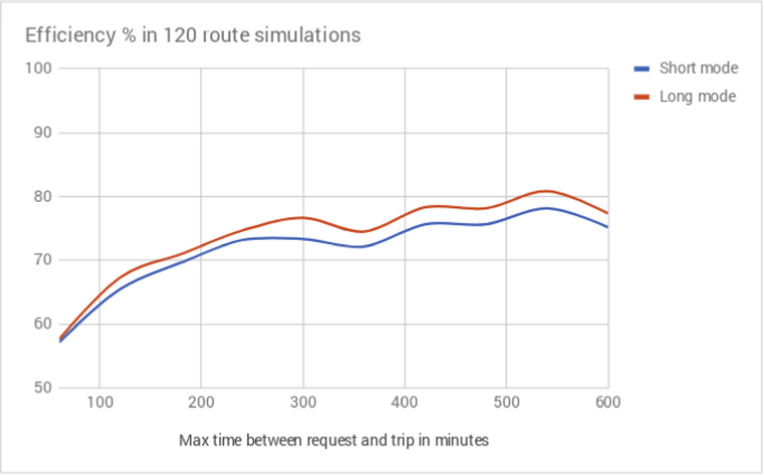

Fig. 6. Efficiency rate and max time between request and trip, when the system handles 120 requests and it is equipped with 54 EVs.

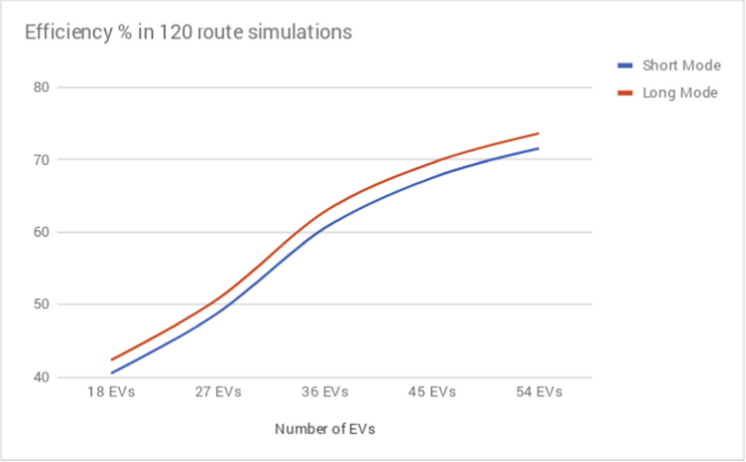

Fig. 7. Efficiency rate graph when the system handles 120 requests and the system is equipped with 18, 27, 36, 45 and 54 EVs.

the optimal number of EVs for our use case of 27 stations in Bristol, is 36 vehicles as it offers a slightly better increase in performance.

The Long mode algorithm has the advantage of looking ahead for vehicles that can substitute a current candidate vehicle with enough charge and a future route. That vehicle would be characterized as unavailable in Short mode, but now we can make one swap with the help of the substitute vehicle, and make the current candidate one, available for trips. Due to the fact that the Long mode incorporates only one such swap, the gains are relatively small in percentage

when compared with the results of the Short mode. We can see that this is confirmed by the observations in the charts as discussed earlier.

In Fig. 5 the overall efficiency shows some small fluctuations and it is not influenced significantly by the requests allocation density within the day, mainly because of the low number of requests. Moreover, if we look closer in the chart at Fig. 7, we will see that as the number of vehicles increases, the gain of the efficiency rate of the Long mode algorithm is almost constant. This is to be expected, because the current version of the Long mode incorporates only one swap functionality and the number of requests (120) is relatively too small to significantly affect the gain. A future version of the Long mode with more swaps would increase the gain when the vehicles increase.

For our test strategy on the software package, we used black-box test design techniques that included equivalence partitioning and boundary value analysis for proper input validation testing. The system was evaluated when the user scenarios were executed according to the test cases generated by our "route requests" generation tool, which simulates user requests for trips. Taking all tests into account, we can say that our software solution is thoroughly tested and achieves its aim and objectives.

6 Conclusions and Future Work

In this paper, we studied a setting where electric vehicles (EVs) can be hired to drive from pick-up to drop-off stations in a mobility-on-demand (MoD) scheme. In this vein, we proposed two scheduling algorithms for assigning EVs to trips. The first one is efficient for short term reservations, while the second for both short and long term ones. While evaluating our algorithms in a setting using real data on MoD locations, we observe that the long term algorithm achieves on average 2.08% higher customer satisfaction and 2.87% higher vehicle utilization compared to the short term one in 120 requests per day.

Moreover, we designed, implemented and tested a system that consists of a web platform and a mobile application. The web platform allows administrators to manage users, stations, vehicles, routes, and accepts or denies vehicle requests using our algorithms. The mobile application lets users register, login, see the available stations in their area and send vehicle requests for trips to the web platform. The platform executes our algorithms in order to decide whether to accept or deny them.

As future work we aim to apply machine learning techniques in order to efficiently predict future customers' demand. Moreover, we aim to use load balancing techniques to enhance the charging procedure of the EVs. In addition, given that EVs hold large batteries we consider using them as temporal energy storage devices in a vehicle-to-grid domain. Finally, we aim to study ways to manage the uncertainty in future trip execution caused by unexpected circumstances such as traffic congestion and accidents.

References

1. Tomic, J., Kempton, W.: Using fleets of electric-drive vehicles for grid support. J. Power Sources **168**, 459–468 (2007)
2. Mitchell, W., Borroni-Bird, C., Burns, L.: Reinventing the Automobile: Personal Urban Mobility for the 21st Century. The MIT Press, Cambridge (2010)
3. Pavone, M., Smith, S., Frazzoli, E., Rus, D.: Robotic load balancing for mobility-on-demand systems. Int. J. Robot. Res. **31**, 839–854 (2012)
4. Pavone, M., L. Smith, S., Frazzoli, E., Rus, D.: Load balancing for mobility-on-demand systems. In: Robotics: Science and Systems (2011)
5. Smith, S., Pavone, M., Schwager, M., Frazzoli, E., Rus, D.: Rebalancing the rebalancers: optimally routing vehicles and drivers in mobility-on-demand systems. In: 2013 American Control Conference, pp. 2362–2367 (2013)
6. Carpenter, T., Keshav, S., Wong, J.: Sizing finite-population vehicle pools. IEEE Trans. Intell. Transp. Syst. **15**, 1134–1144 (2014)
7. Drwal, M., Gerding, E., Stein, S., Hayakawa, K., Kitaoka, H.: Adaptive pricing mechanisms for on-demand mobility. In: Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems, pp. 1017–1025 . International Foundation for Autonomous Agents and Multiagent Systems (2017)
8. Rigas, E., Ramchurn, S., Bassiliades, N.: Algorithms for electric vehicle scheduling in large-scale mobility-on-demand schemes. Artif. Intell. **262**, 248–278 (2018)
9. Barth, M., Shaheen, S.: Shared-use vehicle systems: framework for classifying carsharing, station cars, and combined approaches. Transp. Res. Rec. J. Transp. Res. Board **1791**, 105–112 (2002)