

Smart Balancing of E-scooter Sharing Systems via Deep Reinforcement Learning

Gianvito Losapio¹, Federico Minutoli¹, Viviana Mascardi¹ and Angelo Ferrando¹

¹DIBRIS, University of Genova, Italy

Abstract

Nowadays, micro-mobility sharing systems have become extremely popular. Such systems consist in fleets of electric vehicles which are deployed in cities, and used by citizens to move in a more ecological and flexible way. Unfortunately, one of the issues related to such technologies is its intrinsic load imbalance; since the users can pick up and drop off the electric vehicles where they prefer. We present ESB-DQN, a multi-agent system based on Deep Reinforcement Learning that offers suggestions to pick or return e-scooters in order to make the fleet usage and sharing as balanced as possible.

Keywords

Micro-mobility, E-scooter Sharing Systems, Multi-agent Systems, Deep Reinforcement Learning

1. Introduction

In the last few years, micro-mobility sharing systems have become extremely popular. More and more companies are purchasing fleets of electric vehicles to be deployed in many cities around the world, allowing users to easily rent vehicles via a smartphone app. The last trend is to offer a so-called "free-floating" or "dockless" service related to e-scooters, e-bikes or e-moped: the vehicles can be picked-up or dropped-off anywhere within an operative area designed by the service provider to cover most of the busiest areas of cities [1, 2].


The great flexibility of such a service comes with the challenge of unpredictable usage patterns, with the result of an imbalanced distribution of the electric vehicles around the city. Moreover, battery capacity is limited and many vehicles can rapidly become out-of-charge during the course of the day, if overused in quick succession. In order to preserve a good quality of service despite of imbalance problems and battery limitations, companies need to devote a large operational effort for an efficient fleet management [3].


Typically, specialised workers are employed to accomplish two different, yet complementary tasks, namely *battery swap* and *relocation*. *Battery swap* refers to the process of inserting new batteries into out-of-charge vehicles, whereas *relocation* refers to the process of moving vehicles from one zone to another in order to rebalance the fleet distribution [4].


Quantity of workers, modality and frequency associated to *battery swap* and *relocation* operations represent crucial aspects in the definition of an efficient fleet management policy. A

WOA 2021: 22nd Workshop "From Objects to Agents", Bologna, Italy, 1-3 September 2021

✉ gvlosapio@gmail.com (G. Losapio); fede97.minutoli@gmail.com (F. Minutoli); viviana.mascardi@unige.it (V. Mascardi); angelo.ferrando@unige.it (A. Ferrando)

ORCID  0000-0002-1024-7512 (G. Losapio); 0000-0002-5472-7673 (F. Minutoli)

 © 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

critical trade-off is required to avoid high operational costs and, at the same time, maximize the usage of vehicles. Recently, users engagement has been proposed as a viable solution to alleviate the aforementioned problems. As a result, nowadays several companies engage users in various ways to solve the imbalance and the battery limitation problems [5, 6, 7].

In this work, we present ESB-DQN, a multi-agent system based on Deep Reinforcement Learning (Deep RL) capable of proposing convenient alternative locations for picking up or returning e-scooters. Every time a user is willing to rent an e-scooter, he/she is encouraged to accept alternative pick-up or drop-off points in exchange for monetary incentives.

Based on demand forecast models and artificial intelligence techniques, the ESB-DQN system is able to learn convenient recommendations for the users, in order to maximize the vehicle availability and, at the same time, minimize the number of *battery swap* and *relocation* operations. As a result, the system is able to improve service efficiency and to increase the service provider's long-term revenue. Provided with a smart monetary incentive mechanism, the system is also intended to improve customers' satisfaction and fidelity.

The code that supports the findings of this study is available upon request.

The main contributions of our paper are the following:

- an innovative customer-oriented rebalancing strategy has been defined through a multi-agent system based on deep reinforcement learning;
- an existing simulator of mobility sharing systems has been integrated with a state-of-the-art library for deep reinforcement learning;
- simulations based on real data have been carried out to preliminarily quantify the benefits of the proposed approach.

The paper is organized as follows: Section 2 contains an overview of related works. Section 3 describes the materials and the methods used throughout this research. Section 4 presents the experiments that have been carried out as well as the corresponding results. Section 5 concludes the paper with discussions and possible future works.

2. Related works

The recent work by Wen and colleagues [7] provides a comprehensive overview of the rebalancing strategies used to alleviate the imbalance problem in bike sharing systems. Such strategies have been classified according to two main categories: truck-based rebalancing and customer-oriented rebalancing. Truck-based rebalancing refers to the *relocation* operations mentioned above in Section 1. A specialized group of workers is in charge of moving vehicles from one zone to another by means of trucks. On the other hand, customer-oriented rebalancing is the process of encouraging users to adopt efficient behaviours by providing incentives. The latter category is the main topic of our investigation.

Most past works on rebalancing strategies are not targeted towards "free floating" systems. In particular, papers investigating truck-based rebalancing determine the optimal inventory for each station and design a dynamic optimal truck route with budget constraint [8, 9]. Analogously,

paper investigating customer-oriented rebalancing ponder the role of stations in the incentive proposals mechanism [5, 6, 7].

In our work, both rebalancing strategies have been taken into account: truck-based rebalancing is implemented through the simulator, whereas customer-oriented rebalancing is implemented through the reinforcement learning system. Few other works employ deep RL to investigate user incentives in bike sharing systems, including [10, 11]. However, their objective is to determine an optimal pricing mechanism, whereas the objective of our work is to determine convenient pick-up/drop-off zones for each booking request.

The motivation behind the use of deep Reinforcement Learning for such a task is mainly related to the possibility of combining many interesting aspects at once. The deep RL system can indeed incorporate demand forecasting models as a baseline to drive agents' behaviours and, at the same time, can learn efficient suggestions based on past experience and adapt to real-time demand and availability of the system. In this way, the decision process behind the offered suggestions can capture complex information about the dynamics of the mobility system. Furthermore, by formulating the problem as a game, several constraints may be introduced to enforce specific objectives in the mobility system (e.g., a target service availability).

Compared to previous works, the innovative contribution of our paper is thus twofold. On one side, the imbalance problem inside "free-floating" e-scooter mobility systems has been addressed for the first time. Both rebalancing strategies proposed so far in the literature have been adapted from station-based sharing systems. On the other side, a deep RL multi-agent system in charge of suggesting pick-up/drop-off zones constitutes an original solution which does not build on any existing work. The main influential work has been [4], in which the simulator has been introduced (Section 3.2) - an essential component of the ESB-DQN system.

3. Materials and methods

3.1. Data

To investigate the free-floating imbalance problem, we rely on actual e-scooter trips open data published by the Municipality of Louisville¹, Kentucky. The data comes fuzzed both in time and space for privacy reasons; in particular, each trip has any time-related information rounded to the closest quarter of an hour and any space-related information rounded at the 3rd decimal for both latitude and longitude. Hence, we follow the disaggregation procedure described in [4] such that each trip retains a unique Id, the duration, the distance, the start time, the end time, the start location and the end location. The main characteristics of the dataset are summarized in Table 1. They refer to a *training window* of observations registered over the whole year 2019. The number of trips in the simulation, denoted as N *trips sim* refers to a simulation window of observations over one single day, namely January 01, 2020.

Louisville's e-scooter ecosystem has rather limited complexity, reflecting heterogeneous temporal and spatial demands at the same time. Nonetheless, in order to further ease the formulation of the problem, the whole operative area in the city of Louisville has been quantized in a set \mathcal{Z} of $l \times l$ square zones as proposed in [4], with l the side of the squares, a key parameter

¹<https://data.louisvilleky.gov/dataset/dockless-vehicles>

Table 1

Main characteristics of Louisville dataset

City	N scooters	Avg trip dur. (s)	Avg trip dist. (m)	N zones	N trips train	N trips sim
Louisville	800	814	1601	279	199 789	154

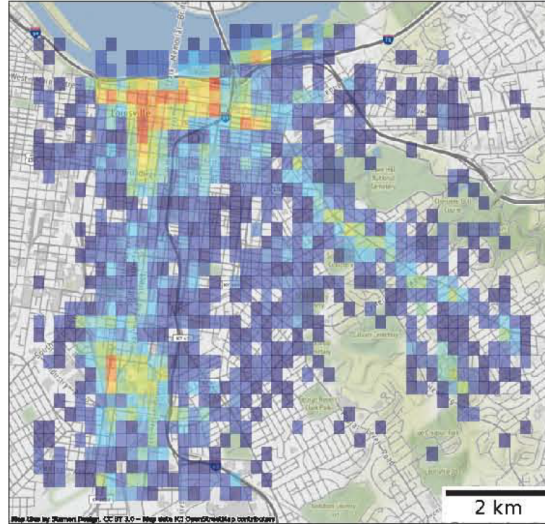


Figure 1: Characterisation of zones in the city of Louisville, following a heatmap colour scale on the number of average trip requests per zone in 2019. Credits to [4].

later clarified in Section 4. As a result, the start/end locations of each trip report the Id of the corresponding zone membership. Each zone $z_i \in \mathcal{Z}$ is associated with a set of valid 1-hop neighbours \mathcal{N}_{z_i} , i.e. the zones among the 8 adjacent zones registering at least one booking request within the *training window* of observations. As it can be seen in Figure 1, many of the zones do not have a full set of valid 1-hop neighbours, i.e. $|\mathcal{N}_{z_i}| \neq 8$. In fact, almost all of them do not, with a grand total space of valid neighbours, \mathcal{N}_{valid} , amounting to only the 60.6% of the whole space of possible neighbours \mathcal{N}^* , with $|\mathcal{N}^*| = 279 * 8 = 2232$.

Both the large subset of invalidity, $\mathcal{N}_{invalid}$, and the tiny simulation window of observations over January 01, 2020 only, are further discussed in Section 4 as they play a key role in the reasoning behind the training of the ESB-DQN multi-agent system.

3.2. Simulator

A modified version of the SimPy-based simulator presented in [4] has been used to simulate e-scooter sharing system dynamics in Louisville. A formal description of the simulator follows:

Fleet and zones. Let \mathcal{S} be the fleet of e-scooters. At any time t , each e-scooter $s \in \mathcal{S}$ is characterised by a unique plate Id, the state of availability, the state of charge of the battery

$b(s) \in [0, B]$, with B being the battery capacity, and the location $l(s)$ as a zone Id in \mathcal{Z} .

Trip requests. The simulator processes trip request events by directly reading them from the input trace over 2019. When the i -th trip request event fires at time t_i , the simulator checks whether there is any e-scooter s with enough residual energy, i.e., $b(s) \geq e_i$, being e_i the energy to complete such trip, either available in the same zone or in the 1-hop neighbouring zones (i.e., the 8 surrounding zones). This is equivalent to assume that customers will by default rent the nearest available e-scooter having enough battery charge.

Incentive proposals. In alternative, users are incentivized to pick-up and/or drop-off the vehicle from/to a different zone (in a limited nearby area). They randomly accept or decline the proposal according to a willingness factor $w \in [0, 1]$, and eventually get their incentive once the trip has been completed. If no alternative pick-up proposal is accepted and no scooter is available in the 1-hop neighbouring zones, the trip request is marked as *unsatisfied*.

Trip completion. Once the pick-up zone $p(i)$ and the drop-off zone $d(i)$ are defined, a trip-end event is scheduled at time $t_i + \delta t_i$, being δt_i the duration of the rental - drawn from a Gaussian distribution with mean μ equal to the duration of the trip reported in the trace, and standard deviation σ equal to 4 minutes, as a form of variability. When the trip-end event fires, the simulator makes the e-scooter s back available in position $d(i)$, and updates its battery charge $b(s) = b(s) - e(i)$. If $b(s) < \alpha B$, with α the operability threshold $\in [0, 1]$, the scooter s is marked as *dead* and is no longer available until a battery swap operation is performed.

Battery swap. Once every T_s time steps, a fleet of n_{swap} battery swap workers is triggered to perform battery swaps operations. Each worker is assigned a battery swap schedule, which consists of up to n_v vehicles to be re-charged inside several zones. Battery swap schedules are created and assigned with the following criteria: we compute the battery charge deficit for each zone z at time t , $\Delta_s(t, z)$, with $t = kT_s$, as the number of dead vehicles waiting for service in z . Then the zone z_o with the least deficit is identified, and a priority 0 queue is constructed for all the other $n - 1$ zones, with priority defined as:

$$p(t, z) = \frac{1}{\Delta_s(t, z)} + \frac{d(z, z_o)}{\max(d(z_j, z_o)_{j=1, \dots, n})}$$

with $d(z_i, z_k)$ being the Haversine distance between the i -th and the k -th zone. Each worker is then assigned a subset of the queue, potentially across multiple zones, following a lump sum costs policy whose goal is to construct a schedule that keeps the expected profit in the next T_s time steps, $P_{\text{swap}, t+T_s}$, higher than the expected battery swap costs, $C_{\text{swap}, t}$: an average cost of service has to be assumed for each vehicle, C_v . As soon as all the workers have an assigned schedule as a sequence of zone Ids, the shortest path to completion is computed for each of them by solving an equivalent TSP optimization problem. Once all the battery swap operations are completed, the workers wait as idle in their last zone on schedule.

Relocation. Once every T_r time steps in a limited working time interval T_{work} , a fleet of n_{rel} relocation workers is triggered to perform relocation operations. Each worker is assigned a

relocation schedule, which consists of up to n_v vehicles to be moved from some zones to others in order to balance the system. Relocation schedules are created following a similar criteria to what has been described above: a deficit $\Delta(t, z)$ is computed for each zone z , a priority 0 queue is computed off of that and a number of schedules are first generated following a lump sum costs policy and then optimized via TSP. In this case, $\Delta(t, z)$ is computed observing the availability of e-scooters with respect to the expected inward and outward flows for the zone z at time t computed over 2019 following the predictive model proposed in [4].

Initialization. At start time, e-scooters are randomly placed among the zones of the grid with uniform random charge $b(s) \in [B/2, B]$. Afterwards, both relocation and battery swap workers are similarly placed with uniform random among the 30 zones that have registered the highest demand in the training data over 2019. This is equivalent to assume the existence of landmarks within the city of Louisville that require a higher concentration of e-scooters.

Originally, battery swap operations were treated differently from relocation ones, as battery swap workers were modelled as a FIFO queue that would react on the fly to out of charge events. In this work, we have leaned towards the hourly scheduling approach already followed by relocation workers, as this would allow us to have a rough idea of the hourly workforce of battery swap workers that is necessary to do any sort of planning whose long-term objective is to reduce the overall maintenance costs of the system.

3.3. ESB-DQN multi-agent system

A multi-agent system has been designed, in charge of proposing alternative pick-up/drop-off zones to the users in change of incentives. In particular, two agents are defined: a *pick-up agent*, P , and a *drop-off agent*, D . At every generated trip request i with pick-up zone $p(i)$ and drop-off zone $d(i)$, the pick-up agent proposes an alternative pick-up zone $\hat{p}(i)$, whereas the drop-off agent proposes an alternative drop-off zone $\hat{d}(i)$. Both proposals share the same ultimate goal of improving the long-term balance of the system, while reducing the overall costs of service due to general maintenance, battery swap ops and relocation ops.

The next three paragraphs describe the fundamental components of the E-scooter Balancing DQN, or ESB-DQN for short, multi-agent system.

3.3.1. Environment

The environment wraps the modified simulator described in Section 3.2 to make it compliant with DeepMind’s DQN Zoo library for reinforcement learning [12]. The major change we have made to said simulator is conceptual: rather than simulating the whole cascade of trip requests between two time intervals of start and finish, t_0 and t_N , collecting a certain number of statistics about the run afterwards, as the original in [4] does, the simulator moves step by step across the states of the Louisville environment. The state, \mathbf{X}_t , is observed as soon as an environment-changing event fires, i.e., a trip request is scheduled; such observation is available to P and D , which will consequently pick an action, a_t . The simulator will then

move forward of one step into the state \mathbf{X}_{t+1} by applying such action. Formally, it is a fully observable environment which produces a $n \times 3$ state vector \mathbf{X}_t at every trip request i at time t :

$$\mathbf{X}_t = \begin{matrix} & A_{(n \times 1)} & B_{(n \times 1)} & C_{(n \times 1)} \\ \begin{matrix} n \times 3 \\ \end{matrix} & & & \end{matrix} = \begin{bmatrix} a_1 & b_1 & 0 \\ a_2 & b_2 & 0 \\ \vdots & \vdots & \vdots \\ a_p & b_p & 1 \\ \vdots & \vdots & \vdots \\ a_d & b_d & 1 \\ \vdots & \vdots & \vdots \\ a_n & b_n & 0 \end{bmatrix} \quad (1)$$

with $n = |\mathcal{Z}|$ the total number of zones, A the $n \times 1$ column vector with the number of available vehicles per zone z at time t , B the $n \times 1$ column vector with the deficit $\Delta(t, z)$ per zone z with respect to the expected optimal baseline at time t , introduced in Section 3.2, and C the two-hot encoded vector with 1s in correspondence of $p(i)$ and $d(i)$ only.

The vectors A and B are standardized via z-normalization to achieve a mean of 0 and a standard deviation of 1. C plays the role of a de-facto attention mechanism within the state \mathbf{X}_t itself. Indeed, it signals which zones of the operative area may be subject to alterations in the near future leading towards the state transition \mathbf{X}_t to \mathbf{X}_{t+1} , which may reflect in how knowledgeable the alternative proposals are.

Despite a detailed action space definition follows in the next paragraph, it is important to note that the ESB-DQN environment belongs to the family of constrained environments, i.e., the setting of our problem falls within constrained deep Reinforcement Learning. There are a number of ways to approach constraint-guided interactions to lead RL agents towards safe behaviour in their exploration. For example, an exploration pattern often persevered is to pretend those unsafe actions do not exist altogether, by strictly avoiding them from the range of actions the agents can pick. Or again, a terminal state may be invoked each time an invalid action is taken, and a new episode started over hoping for better fortune. Here instead, we focus on the third popular paradigm of constrained RL, that is, to let the invalid action pass through, but awarding the agent committing it a strongly penalized reward. Indeed, [13] show that this approach is actually the most beneficial under most constrained RL settings to augment the interaction capabilities of the agents with the surrounding environment, while not altering nor interrupting too abruptly their perception of it. The only limitation of this approach is that the harshly penalized reward should be ensured to be at least an order of magnitude smaller than the lowest possible reward achievable as a result of a valid action. Our approach is similar to [13], as we define a *fall-back* action, or NOP, that the agents can fall back to whenever they pick an invalid action, getting severely penalized as a result, to prompt the continuity of the simulation. In Section 3.4 we further explore this continuity while training the ESB-DQN system, by introducing the concept of *lives*, borrowed from Atari games [14].

3.3.2. Agents architecture

The agents are Deep-Q-Networks (DQN) implementing an ϵ -greedy policy with experience replay [14] belonging to the family of Q-learning. It is an off-policy approach towards deep RL wherein the agent estimates the expected reward for future actions from a given state without following an actual greedy policy, but instead relying on a behaviour policy enriched from direct experience with the environment to update the online policy, by satisfying Bellman's optimality equation. Such an approach is better suited for large state spaces, \mathcal{S} , against rather limited action spaces, \mathcal{A} , which we will see to be our case. In fact, they are Rainbow agents [15], a state-of-the-art DQN agents, which we have found beneficial for the three following main features: double Q-learning helps in preventing overestimation of the action values which may lead to very unpleasant proposals; distributional Q-learning helps in investigating the importance of the value distribution, which we find necessary to achieve long-term balance of the ESB-DQN system; prioritized experience replay helps in selecting the subset of previously experienced observations that are the most relevant, which we find necessary to characterize the complexity of the dynamics behind a free-floating sharing system .

Both the pick-up agent and the drop-off agent comprise a funnel-like three-layer fully connected network with ReLU activation functions, whose role is to flatten the input and extract a latent representation as a single vector of 256 units. The input of the network is the last observed environment state, \mathbf{X}_t , whereas the output feeds the standard Rainbow network that produces a distribution of logits, whose maximum value identifies the action picked by each agent, $a_{P,t}$ and $a_{D,t}$, respectively. The action space is limited to 9 different choices, corresponding to the 8 cardinal directions mapping the 8 adjacent zones (i.e., 1-hop neighbourhood) plus the calling zone, $p(i)$ or $d(i)$ respectively, which function as the NOP actions.

Following this formulation of the action space, and recalling Figure 1, it becomes clear why the ESB-DQN environment is constrained by a large set of invalid actions. In fact, in the early stages of the RL agents life-cycle, the expectation of picking an invalid action from any given zone z at any given time t far exceeds its complementary, which is further evidence of the need for outer aid for the RL agents to well characterize the dynamics of the system.

3.3.3. Reward

The following functions are defined to compute the reward:

$$\omega(z, t) = \Delta(z, t) \exp \left[- \left(\frac{1}{d(z, t)^+} N_A(z, t)^+ \right)^{\text{sign}(\Delta(z, t))} \right] \quad (2)$$

$$\psi(z, t) = N_D(z, t) \exp \left(- \frac{1}{d(z, t)^+} N_A(z, t) \right) \quad (3)$$

where:

- $\Delta(z, t)$: expected deficit of e-scooters at zone z at time t ;
- $d(z, t)$: future demand of e-scooters in z at time t (in a time interval $t + \Delta t$);

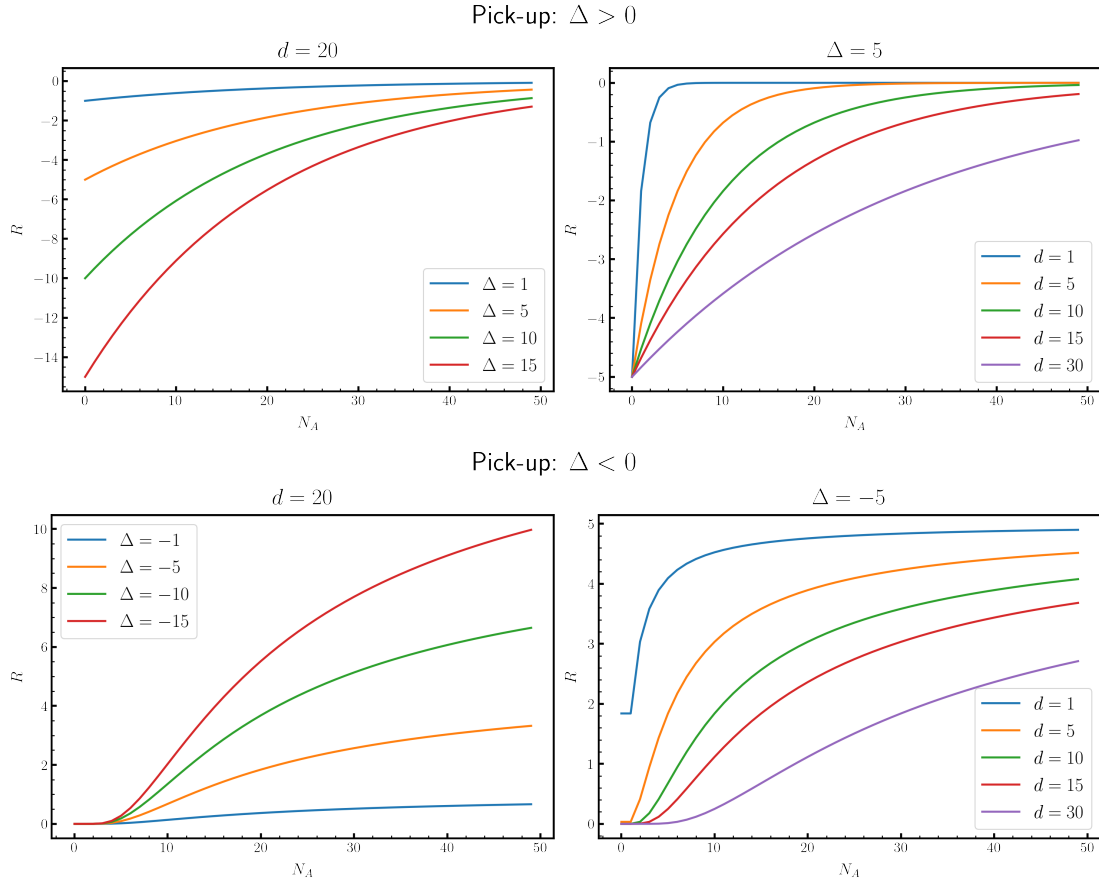


Figure 2: Pick-up action: some examples of the reward function R_P for different values of the parameters Δ, d . **(Upper row)** Negative reward: the agent suggests to pick-up a vehicle from a zone having an expected deficit of vehicles ($\Delta > 0$). As Δ and d increase, the reward is smaller because the expected deficit condition will be worsened. In both cases, the larger the number of available vehicles the higher the curve, as the deficit condition will be alleviated. **(Bottom row)** Positive reward: the agent suggests to pick-up a vehicle from a zone having an expected surplus of vehicles ($\Delta < 0$). The larger Δ the larger the reward, reflecting how problematic the surplus being improved. Similarly, when the number of available vehicles is high, the rebalancing effect is considered more valuable. As the demand d increases, the reward decreases because the pick-up may negatively affect the long-term balance of the zone.

- $N_A(z, t)$: number of available e-scooters in z at time t ;
- $N_D(z, t)$: number of dead e-scooters in z at time t ;
- $(\cdot)^+$ denotes the function $\max(\cdot, 1)$ and is used to prevent from division by zero.

Drop-off agent. Let $\hat{d}(i)$ be the chosen alternative drop-off zone for trip i at time t , $\mathcal{N}_{\hat{d}(i)}$ be the set of valid neighbours around $\hat{d}(i)$. If the state of charge of the vehicle s at the end of the trip is greater than the battery swap threshold, i.e. $b(s) - e_i \geq \alpha C$, then the reward

corresponding to each of the alternative zones $z \in \mathcal{N}_{\hat{d}(i)}$ is:

$$R_D(z, t) = \begin{cases} \omega(z, t) & \text{if } \hat{d}(i) = z \\ -\omega(z, t) & \text{otherwise} \end{cases}$$

Otherwise:

$$R_D(z, t) = \begin{cases} \psi(z, t) & \text{if } \hat{d}(i) = z \\ -\psi(z, t) & \text{otherwise} \end{cases}$$

The overall reward for the choice $\hat{d}(i)$ is:

$$\bar{R}_D(\hat{d}(i), t) = \begin{cases} \frac{1}{|\mathcal{N}_{\hat{d}(i)}|} \sum_{z \in \mathcal{N}_{\hat{d}(i)}} R_D(z, t) & \text{if drop-off action is valid} \\ -\gamma_D \max_{z \in \mathcal{N}_{\hat{d}(i)}} |R_D(z, t)| & \text{otherwise} \end{cases}$$

with γ_D being a constant which modulates the penalty of an invalid drop-off action.

Pick-up agent. Let $\hat{p}(i)$ be the chosen alternative pick-up zone for trip i at time t , $\mathcal{N}_{\hat{p}(i)}$ be the set of valid neighbours around $\hat{p}(i)$. The reward corresponding to each of the zones $z \in \mathcal{N}_{\hat{p}(i)}$ is:

$$R_P(z, t) = \begin{cases} -\omega(z, t) & \text{if } \hat{p}(i) = z \\ \omega(z, t) & \text{otherwise} \end{cases}$$

The overall reward for the choice $\hat{p}(i)$ is:

$$\bar{R}_P(\hat{p}(i), t) = \begin{cases} \frac{1}{|\mathcal{N}_{\hat{p}(i)}|} \sum_{z \in \mathcal{N}_{\hat{p}(i)}} R_P(z, t) & \text{if pick-up action is valid} \\ -\gamma_P \max_{z \in \mathcal{N}_{\hat{p}(i)}} |R_P(z, t)| & \text{otherwise} \end{cases}$$

with γ_P being a constant which modulates the penalty of an invalid pick-up action. Figure 2 shows some examples of the reward function R_P for different values of the parameters Δ , d .

3.4. Lives mechanism

As we have anticipated in Section 3.3, a major role during the training of the ESB-DQN system has been played by the parameter regarding the number of lives, k . The continuity of the simulation is a key factor for the eventual learning of the RL agents, as interrupting the simulation to just start it over too often, as soon as an invalid action happens, would slow down the convergence by a considerable margin, given how full of potential invalid actions ESB-DQN environment is.

To overcome this limitation, we have borrowed the concept of lives from Atari: every time one of the two agents or both commit an invalid action, the whole environment loses a life. By doing so, an invalid action does not immediately lead to a terminal state, but takes it closer to the ESB-DQN state. On life loss, the discount for the timestep t is zeroed, cancelling any connection between the previous and later events, and the agents are set to perform a NOP.

Let $a_t = (a_{t,P}, a_{t,D})$ be the generic action for the simulator taken at time t , defined as the resulting combination of the action picked by the pick-up (P) agent, $a_{P,t}$, and the action picked by the drop-off (D) agent, $a_{D,t}$. The set of invalid actions has been set as follows:

- either zone corresponding to $a_{P,t}$ or $a_{D,t}$ is invalid: $z_{P,t} \notin Z \cup z_{D,t} \notin Z$, with Z the set of valid zones of the city of Louisville;
- the zones corresponding to $a_{P,t}$ and $a_{D,t}$ are equal: $z_{P,t} = z_{D,t}$;
- the zones corresponding to $a_{P,t}$ and $a_{D,t}$ are equal to the original zone of opposite type: $z_{P,t} = \hat{z}_{D,t} \cup z_{D,t} = \hat{z}_{P,t}$;
- the original zones $\hat{z}_{P,t}$ and $\hat{z}_{D,t}$ are equal: $\hat{z}_{P,t} = \hat{z}_{D,t}$;
- the suggested pick-up zone does not have a suitable vehicle ready: $V_{P,\text{avail}} = \emptyset$

As soon as k reaches 0, then the simulation is stopped. Indeed, we would not want our RL agents to learn the dynamics of the environment while committing thousands of errors. It is important to note that by implying the concept of lives, the training framework of RL agents has turned into a sort of collaborative RL framework, wherein both P and D cannot rely solely on their capabilities to reach the goal, but even on the other's to reach a common goal: if D was to lose a life, P would lose it as well, and viceversa.

4. Experiments and Results

The ESB-DQN system has been trained to learn the best alternative zone proposals throughout simulations with the Louisville dataset. The aim of the experiments has been to evaluate whether incentivizing users to pick-up/drop-off vehicles in alternative zones can preserve a good quality of service with a reduced number of relocation and battery swap workers.

The quality of service is measured through the satisfied demand D_{sat} , defined as follows:

$$D_{\text{sat}} = \frac{N_{\text{trips}} - N_{\text{unsat}}}{N_{\text{trips}}} \quad (4)$$

where N_{trips} is the total number of trips, N_{unsat} is the number of *unsatisfied* trips (no available vehicles in the pick-up zone and in the 1-hop neighbourhood), both measured over a given fixed time interval T_{sim} . Through all our experiments, T_{sim} is equal to 1 day.

The parameters of the simulator have been set as follows:

- the number of available e-scooters is $|\mathcal{S}| = 400$;
- the size of the zones is $l = 200m^2$;
- the battery capacity is $B = 425 \text{ Wh}$ with $\alpha = 0.3$, whereas the energy required to complete a trip is proportional to the driving distance by a factor of 11 Wh/km (as suggested in [4]);
- the user willingness is $w = 1$;
- the battery swap operations are scheduled every $T_s = 1\text{h}$;
- the relocation operations are scheduled every $T_r = 1\text{h}$ in a working time interval $T_{\text{work}} = [9\text{AM}-6\text{PM}]$.

The fleet size $|\mathcal{S}|$ and the user willingness w immediately stand out from the lot of parameters. The former has been set to half the nominal fleet size granted by the city of Louisville. Indeed, as further experiments on cities with more complex dynamics have not been conducted for the time being, we have decided to restrict Louisville to a worst case scenario, as the quality of service would remain strong nonetheless (88%). The latter has been set to 1, as in the training phase we wanted to let both RL agents experience as much of the environment as possible, regardless of whether they would be actually asked to do so.

The parameters of the reinforcement learning system have been set as follows:

- the optimizer is Adam with a learning rate of 6.25×10^{-5} ;
- the learning period is 16;
- the batch size is 32;
- the timesteps are aggregated to look back to the last 3 timesteps before any decision process takes place;
- the global gradient norm clipping is 10;
- the importance sampling exponent ranges in $[0.4, 1]$;
- the experience replay buffer has size 5.2×10^3 , amounting to almost 30 full repetitions of the same day over and over, with priority exponent of 0.5;
- the target network update period is 1.6×10^2 ;
- the number of iterations is 48;
- the number of trips per episode is 1.3×10^3 , amounting to almost 10 full repetitions of the same day over and over;
- the number of validation trips is 2.6×10^3 ;
- the number of training trips is 5.2×10^3 ;
- the number of total lives k has been set to 100.

Moreover, concerning the reward function, the future demand is computed in a time interval $\Delta t = 1\text{h}$, whereas the constants modulating the penalties are $\gamma_D = \gamma_P = 2$. Also, every 3 iterations a checkpoint has been stored locally for evaluation purposes.

In the first experiment, the model has been trained from scratch with the number of relocation workers being $n_{\text{swap}} = 12$ and the number of battery swap workers being $n_{\text{rel}} = 6$. Other two experiments have been performed, by drastically reducing the number of workers and applying transfer learning from the pre-trained P and D agents. In particular, in the second experiment we have fixed $n_{\text{swap}} = 6$, $n_{\text{rel}} = 3$ and in the third experiment $n_{\text{swap}} = n_{\text{rel}} = 1$.

The final results in validation are shown in Table 2. The evolution of the satisfied demand during the learning procedure is represented in Figure 3.

The number of validation/training trips follows DeepMind’s suggested ratio of 1 : 2 between the online and the offline ϵ -greedy networks. For example, if a training episode would experience 1000 trips, a validation episode would experience only half of those. A single iteration took over 1 hour on a PC equipped with a GeForce GTX 1650 Ti GPU with 4GB of memory along with an Intel i7-10750H CPU with 32GB of RAM. Both CPU and GPU specs are crucial, as the SimPy processes undergoing the simulation run solely on CPU, whereas the forward and backward

pass of the RL agents' networks happen on GPU.

As shown in Figure 3(a), the two agents trained from scratch cause a decrease in the satisfied demand during the first iterations, due to their random behaviour with no previous experience. After around 25 iterations their policies have been efficiently updated. The level of satisfied demand has improved with respect to the baseline - referred to a standard mobility service with no user incentives. More interestingly, by reducing the number of workers and applying transfer learning, it is possible to observe again a beneficial effect over the satisfied demand. In particular, Figure 3(c) shows that in the critical scenario with $n_{\text{swap}} = n_{\text{rel}} = 1$ the satisfied demand is constantly larger with respect to the baseline. This means that by following the proposal of alternative pick-up and drop-off zones, users are actively participating to the system rebalancing and contributing to a positive increase of the quality of service.

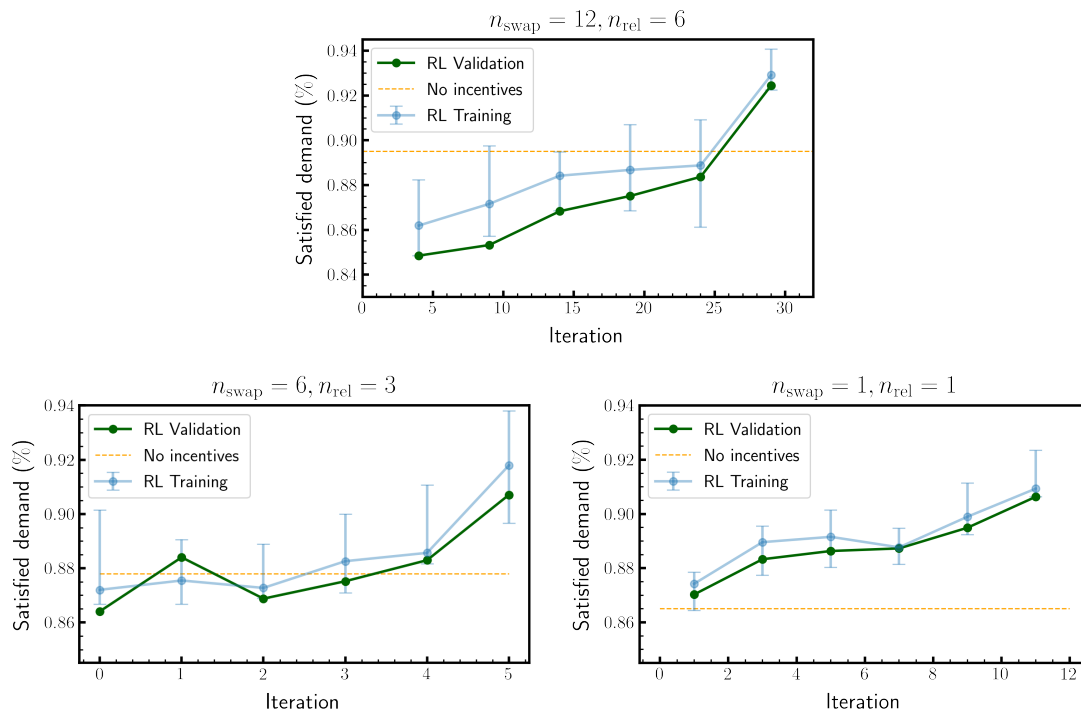


Figure 3: Evaluation of the satisfied demand during the learning procedure for training and validation agents versus a baseline model with no incentive policy (user willingness $w = 0$). The value of the parameters is shown in the titles. (a) Model trained from scratch, (b), (c) Transfer learning.

5. Discussion and Future Works

In this paper, we presented ESB-DQN, a multi-agent system based on deep reinforcement learning able to interact with a simulator in order to learn alternative pick-up and drop-off zones in e-scooter sharing services. The main objective is to combat the imbalance problem

Table 2
Results

Parameters	Training mode	N iterations	Satisfied demand D_{sat}	
			ESB-DQN	No incentives
$n_{\text{swap}} = 12, n_{\text{rel}} = 6$	from scratch	30	0.92	0.89
$n_{\text{swap}} = 6, n_{\text{rel}} = 3$	transfer learning	6	0.91	0.88
$n_{\text{swap}} = 1, n_{\text{rel}} = 1$	transfer learning	12	0.90	0.86

by providing user incentives in order to optimize vehicle availability as well as battery swap and relocation operations. At present, ESB-DQN expects to know the original pick-up and drop-off locations of each generic scheduled trip, $p(i)$ and $d(i)$, beforehand, in order to produce proper suggestions. Of course, such a constraint poses a strong limitation to the effectiveness of the system, as it is impractical to always expect users to know their future drop-off location before initiating the trip. Nevertheless, following the way the original simulator handles the notion of booking requests as pairs of pick-up and drop-off locations, forcing both pick-up and drop-off agents, P and D , to operate synchronously was a necessary starting point. The natural evolution of the ESB-DQN system requires the untying of this synchrony, to let P and D affect the state of the environment independently at different stages.

Preliminary experiments on real e-scooter data from Louisville (US) have shown encouraging results on the satisfied demand of the system, even with a strongly reduced number of workers.

Further experiments are required for a comprehensive evaluation of the ESB-DQN system. By varying different parameters of the simulator, e.g., the number of e-scooters $|S|$, the number of relocation workers n_{rel} or battery swap workers n_{swap} , it is possible to study how each of them, in turn, affects the user incentives policy. It is worth mentioning that more accurate demand forecasts for the computation of $\delta(z, t)$ in Eq. 2, 3 can be adopted with the aim of getting further improvements on the overall performance of the ESB-DQN system.

A fundamental effort should be devoted to scale-up experiments on a larger temporal scale and on larger datasets (e.g., Austin open data [16]). A larger number of iterations would indeed reflect in a better characterisation of the ϵ -greedy policy. Indeed, despite both RL agents have reached some sort of convergence with even a few iterations, there may be a few specific corner cases of states that leave them both unable to decide with high consistency. Concerning a possible speed-up, since SimPy processes run on the CPU, there is a lot of time left to gain by optimizing the underlying simulator to fasten the run time of a single day. On the other hand, the code related to the multi-agent system is already optimized for GPUs and TPUs.

Another interesting possibility is to apply the ESB-DQN system to other mobility sharing systems with different vehicles (e.g., e-bikes, e-moped). Provided with the right data and the proper scenario parameters (e.g., fuel type, fuel consumption, maintenance costs) both the simulator and the multi-agent system can be directly applied to such problems.

The proposed approach may be deployed in real mobility systems as a real-time service following the REST paradigm, integrated into existing app used by mobility service providers. A prototype of the API is under development along with a chatbot intended to provide a natural language interface the users could interact with as well.

References

- [1] A. Chang, L. Miranda-Moreno, R. Clewlow, L. Sun, Trend or Fad? Deciphering the Enablers of Micromobility in the US, Technical Report, 2019.
- [2] C. S. Smith, J. P. Schwieterman, E-scooter Scenarios: Evaluating the Potential Mobility Benefits of Shared Dockless Scooters in Chicago (2018).
- [3] R. Zhu, X. Zhang, D. Kondor, P. Santi, C. Ratti, Understanding Spatio-temporal Heterogeneity of Bike-sharing and Scooter-sharing Mobility, *Computers, Environment and Urban Systems* 81 (2020) 101483.
- [4] A. Ciociola, M. Cocca, D. Giordano, L. Vassio, M. Mellia, E-scooter Sharing: Leveraging Open Data for System Design, in: 2020 IEEE/ACM DS-RT, IEEE, 2020, pp. 1–8.
- [5] J. Pfrommer, J. Warrington, G. Schildbach, M. Morari, Dynamic Vehicle Redistribution and Online Price Incentives in Shared Mobility Systems, *IEEE Transactions on Intelligent Transportation Systems* 15 (2014) 1567–1578. doi:10.1109/TITS.2014.2303986.
- [6] C. Fricker, N. Gast, Incentives and Redistribution in Homogeneous Bike-sharing Systems with Stations of Finite Capacity, *EURO Journal on Transportation and Logistics* 5 (2016) 261–291. URL: <https://www.sciencedirect.com/science/article/pii/S2192437620300959>. doi:<https://doi.org/10.1007/s13676-014-0053-5>.
- [7] P. Yi, F. Huang, J. Peng, A Rebalancing Strategy for the Imbalance Problem in Bike-sharing Systems, *Energies* 12 (2019). URL: <https://www.mdpi.com/1996-1073/12/13/2578>. doi:10.3390/en12132578.
- [8] R. Regue, W. Recker, Proactive Vehicle Routing with Inferred Demand to Solve the Bike-sharing Rebalancing Problem, *Transportation Research Part E: Logistics and Transportation Review* 72 (2014) 192–209. URL: <https://www.sciencedirect.com/science/article/pii/S1366554514001720>. doi:<https://doi.org/10.1016/j.tre.2014.10.005>.
- [9] J. Liu, L. Sun, W. Chen, H. Xiong, Rebalancing Bike-sharing Systems: A Multi-source Data Smart Optimization, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 1005–1014. URL: <https://doi.org/10.1145/2939672.2939776>. doi:10.1145/2939672.2939776.
- [10] Y. Duan, J. Wu, Optimizing Rebalance Scheme for Dock-less Bike-sharing Systems with Adaptive User Incentive, in: 2019 IEEE MDM, IEEE, 2019, pp. 176–181.
- [11] L. Pan, Q. Cai, other authors, A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike-sharing Systems, in: Proc. of AAAI, volume 33, 2019, pp. 1393–1400.
- [12] DeepMind, DQN Zoo, 2020. URL: https://github.com/deepmind/dqn_zoo.
- [13] H. Spieker, Constraint-guided Reinforcement Learning: Augmenting the Agent-Environment Interaction, *CoRR abs/2104.11918* (2021). arXiv:2104.11918.
- [14] V. Mnih, K. Kavukcuoglu, other authors, Human-level Control through Deep Reinforcement Learning, *Nature* 518 (2015) 529–533.
- [15] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. G. Azar, D. Silver, Rainbow: Combining Improvements in Deep Reinforcement Learning, *CoRR abs/1710.02298* (2017). URL: <http://arxiv.org/abs/1710.02298>. arXiv:1710.02298.
- [16] Austin Shared Micro-mobility Open Data, 2019. URL: <https://dev.socrata.com/foundry/data.austintexas.gov/7d8e-dm7r>.