Undefined 1 (2014) 1–5
IOS Press

Smart Balancing of E-scooter Sharing Systems via Deep Reinforcement Learning: A Preliminary Study

Gianvito Losapio [a], Federico Minutoli [a], Viviana Mascardi [a], and Angelo Ferrando [a]
[a] *DIBRIS, University of Genova, Italy*
E-mail: gvlosapio@gmail.com, fede97.minutoli@gmail.com, viviana.mascardi@unige.it, angelo.ferrando@unige.it

Abstract. Nowadays, micro-mobility sharing systems have become extremely popular. Such systems consist in fleets of dockless 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 users can pick up and drop off the electric vehicles where they prefer.

In this paper we present ESB-DQN, a multi-agent system for E-Scooter Balancing (ESB) based on Deep Reinforcement Learning where agents are implemented as Deep Q-Networks (DQN). ESB-DQN offers suggestions to pick or return e-scooters in order to make the fleet usage and sharing as balanced as possible, still ensuring that the original plans of the user undergo only minor changes.

The main contributions of this paper include a careful analysis of the state of the art, an innovative customer-oriented rebalancing strategy, the integration of state-of-the-art libraries for deep Reinforcement Learning into the existing ODySSEUS simulator of mobility sharing systems, and preliminary but promising experiments that suggest that our approach is worth further exploration.

Keywords: Micro-mobility, Dockless E-scooter Sharing Systems, Smart Balancing, Multi-agent Systems, Deep Reinforcement Learning

1. Introduction

The adoption of Electric Vehicles (EV) has been constantly growing in the last few years and this trend is expected to accelerate exponentially. From the analysis of the electric vehicle market growth across U.S. cities published in September 2021, it turns out that

> *The electric vehicle market in the United States has grown from a few thousand vehicles in 2010 to more than 315 thousand vehicles sold annually from 2018 to 2020. In 2020, the electric share of new vehicle sales was approximately 2.4%, an increase from about 2% in 2019* [4].

Similar figures, at a global scale, are reported in Global EV Outlook issued in April 2021 by the International Energy Agency, IEA[1].

While these reports deal with any kind of electric vehicles including cars and public transportation means, lightweight two-wheels vehicles play a very important role in boosting the green trend by changing the way we conceive mobility in our cities.

As reported in the SLOCAT Transport and Climate Change Global Status Report 2nd Edition published in June 2021[2],

[1] `https://www.iea.org/reports/global-ev-outlook-2021`, accessed on January 10th, 2022.
[2] `https://tcc-gsr.com/`, accessed on January 10th, 2022.

> *Electric-assisted bicycles (e-bikes) are by far the most popular electrified road transport mode in Europe and North America; e-bike sales in Europe surpassed 4.8 million units in 2020, three times the number of electric passenger cars sold in the European Union (EU) that year.*

Instead of owning personal electric vehicles, many citizens prefer to take advantage of sharing systems. Quoting the SLOCAT Report again,

> *The use of e-bikes in shared systems has also grown strongly since 2017 and a study found that 35% of shared electric bike trips substituted car travel, while 30% substituted walking. As of August 2020, some 2,015 bike-sharing systems were in operation around the world.*
> *In 2019, nearly 140 million trips were taken on shared bicycles and scooters in the USA, up 60% from 2018. Shared scooter use grew more than 100%, while shared bike use increased around 10%.*

It is a matter of fact that in the last few years micromobility 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. From 2017, the growing trend is to offer a so-called "free-floating" or "dockless" service related to e-scooters, e-bikes or e-moped[3]: 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 [6,24].

Dockless shared vehicles have a huge potential, but their great flexibility comes with the challenge of unpredictable usage patterns, with the risk of an imbalanced distribution of the electric vehicles around the city. Moreover, battery capacity is limited and many vehicles can rapidly run 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 [27].

[3]While in this paper we will mainly refer to e-scooters, the problems raised by their adoption in a dockless context and the solution that we propose apply to any small electric vehicle suitable to be shared and used in a limited operational area.

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 [7].

Quantity of workers, modality and frequency associated with *battery swap* and *relocation* operations represent crucial aspects in the definition of an efficient fleet management policy. A critical trade-off is required to avoid high operational costs and, at the same time, maximize the usage of vehicles. Recently, user's 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 [11,22,26].

In this work, that extends the "WOA: From Objects to Agents" 2021 workshop paper [19], we present ESB-DQN (for "E-Scooter Balancing based on Deep Q-Networks"), a multi-agent system based on Deep Reinforcement Learning, Deep RL for short, capable of proposing convenient alternative locations for picking up or returning e-scooters. Every time users are willing to rent an e-scooter, they are encouraged to accept alternative pick-up or drop-off points in exchange for incentives. The alternative points that ESB-DQN considers as valid ones are located in adjacent zones w.r.t. those originally planned by the user. This means for example that a user will never be invited to drop the e-scooter off in a point that is more distant than $2l\sqrt{2}$ meters from the planned one, being l the length of the square zone side. Indeed, users are not meant to become the relocators of vehicles, but just to improve the relocation process without significant changes to their original plans.

Based on demand forecast models and artificial intelligence techniques, the ESB-DQN system learns 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. The system is intended 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 meant to improve customers' satisfaction and fidelity.

Albeit our study is still in its early stages, and is only related with the suggestion of alternative pick-up and drop-off locations, our long-term vision of ESB-DQN

is ambitious, as summarized in Figure 1: a user interacts in natural language with the system whose front-end is an app; the system proposes alternative adjacent pick-up and drop-off locations with incentives, based on the Deep Reinforcement Learning algorithm implemented in the back-end; the user either agrees or declines, always interacting in natural language; the system is also available to answer general questions on the terms and conditions of use of the e-scooter and may provide touristic suggestions on points of interest and on services available in the area.

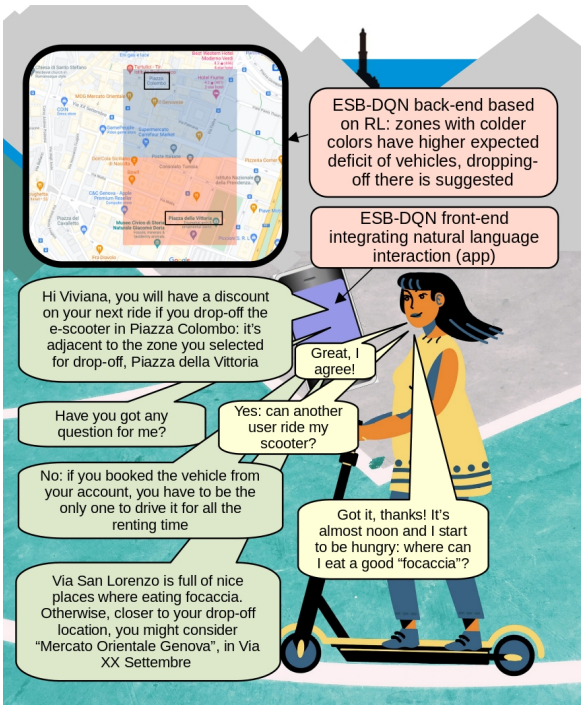

Fig. 1. The ESB-DQN vision, with the front-end implemented as an app interacting via voice with the user, and the back-end integrating the approach presented in this paper. The zone of Piazza Colombo (top rectangle in the representation of the ESB-DQN back-end) is highlighted with a cold color, meaning higher expected deficit of vehicles, while the zone of Piazza della Vittoria (bottom rectangle in the representation of the back-end) is highlighted with a warm one.

The code that supports the findings of this study is available to the research community on GitHub[4] and extends the ODySSEUS simulator developed by the SmartData lab at the Polytechnic University of Torino, Italy[5]. The choice of extending ODySSEUS was a natural one given that — as discussed in Section 2 — it has been designed as an origin-destination simulator of shared e-mobility in urban scenarios, being hence conceived for the same application domain that we address. The kind support provided by the ODySSEUS' developers, who reacted to all our requests in a timely and helpful fashion, represented a further positive element of our choice.

The main contributions of this paper are the following:

- a careful analysis of the state of the art has been carried out;
- 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, ODySSEUS, 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 data used throughout this research. Section 4 describes the simulator we implemented, extending ODySSEUS, to simulate the e-scooter sharing dynamics while Section 5 describes the ESB-DQN multi-agent system. Section 6 presents the experiments that have been carried out as well as the corresponding results. Finally, Section 7 concludes the paper with a discussion of potential and limitations of the proposed approach, and of its future extensions. An Appendix reports details om the simulator parameter and how to use it to reproduce the experiments presented in this paper.

With respect to the WOA 2021 paper, this version adds almost eighty percent new contents including a more thorough analysis of the state of the art, further technical details on the system, and a deeper and more critical discussion of its advantages and disadvantages.

2. Related works

This section deals with three kinds of related works: those that further motivate the interest in the domain

[4] https://github.com/DiTo97/odysseus-escooter-dqn, accessed on January 10th, 2022.

[5] https://smartdata.polito.it/odysseus-an-origin-destination-simulator-of-shared-e-mobility-in-urban-scenarios/, accessed on January 10th, 2022.

addressed by this study, those specifically oriented towards rebalancing strategies, and those that address the e-scooter domain as we do, but from different perspectives.

2.1. Works motivating our study

The raising interest in the adoption of vehicle sharing systems is witnessed, besides by documents and reports by national and international agencies like those mentioned in the Introduction, by many recent scientific papers.

In 2020 Aguilera-García et al. [1] collected data from Spanish cities through an online survey in order to identify the drivers for adoption and frequency of use of moped scooter-sharing services in urban areas. Among their conclusions, we mention that age in the range 26-34 and university level education play a key role in determining a positive attitude towards this mobility approach.

The analysis by Eccarius and Lu [10] is instead targeted towards university students: survey responses from 471 university students in Taiwan were collected, and their analysis showed that lack of perceived compatibility with personal values, mobility needs and lifestyle particularly drives students with low usage intention, while awareness-knowledge about the sharing system and environmental values influence the formation of usage intention in indirect ways.

Comparisons of the performance of bike and scooter sharing have been carried out on data from the Tri-City metropolitan area in northern Poland [3] and from Singapore [27]. The first study concluded that e-scooters are more often used for leisure rides, while e-bicycles are mainly used as first and last mile transport and to commute directly to various places of interest; respondents that adopted shared micromobility are generally young, and e-scooter users are on average younger than e-bike users. The second study confirms the high maintenance cost for rebalancing and charging e-scooter fleets, which is the motivation for our work.

2.2. Works on rebalancing strategies

The recent work by Wen et al. [26] 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 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. Regue and Recker [23] addressed the dynamic bike sharing repositioning problem with demand prediction; although their proposal is consistent with an agent-oriented view, being proactive as demands are anticipated, and being self-adaptive as more precise predictions can be made as new data are available, repositioning takes place in existing stations, and hence the problem they address is different from ours. Junming et al. [18] developed a Meteorology Similarity Weighted K-Nearest-Neighbor regressor to predict the station pick-up demand based on large-scale historic trip records and proposed an inter-station bike transition model to predict the station drop-off demand. Incentive proposals mechanisms are also taken under consideration in papers investigating the role of stations in customer-oriented rebalancing [11,22,26].

In our work, both truck-based and customer-oriented 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 those by Duan and Wu [9] and by Pan et al. [21]. 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.

In particular, Duan and Wu [9] consider rebalancing the dockless bike sharing system by providing users with monetary incentives with the long-term goal of maximizing the number of satisfied users who successfully complete their rides over a period of time. The locations of sources and destinations are extracted from a selected dataset named *Mobike* containing more than 100k trip records of Shanghai, and the experiment results are promising. W.r.t. our work, Duan and Wu's simulator is driven by less parameters and the model of the domain is less precise: neither battery swap costs (in terms of needed workers and employed time), neither relocation cost (again, in terms of workers and time needed to relocate vehicles) seem to be taken

Table 1
Summary of the main features of the works on rebalancing; details are provided for those dealing with dockless fleets only, characterized by "yes" in the **Dockless** column. They include the information if **Battery swap** and **Relocation** are modeled, the used **Dataset**, and whether the proposed tool **Employs Deep RL** or not.

Reference	Year	Dockless	Battery swap	Relocation	Dataset	Employs Deep RL
Pfrommer et al. [22]	2014	no	/	/	/	/
Regue and Recker [23]	2014	no	/	/	/	/
Junming et al. [18]	2016	no	/	/	/	/
Fricker and Gast [11]	2016	no	/	/	/	/
Yi et al. [26]	2019	no	/	/	/	/
Duan and Wu [9]	2019	yes	no	no	Mobike	yes
Pan et al. [21]	2019	yes	no	no	Mobike	yes
Ciociola et al. [7]	2020	yes	yes	yes	Louisville and Minneapolis	no
ESB-DQN	2021	yes	yes	yes	Louisville	yes

into account. The same *Mobike* dataset was used by Pan et al. [21] for developing their Hierarchical Reinforcement Pricing, building upon the Deep Deterministic Policy Gradient algorithm. Their pricing algorithm captures both spatial and temporal dependencies using a divide-and-conquer and outperforms state-of-the-art methods in both service level and bike distribution.

The motivation behind the use of deep Reinforcement Learning for our 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, few works address the imbalance problem inside "free-floating" e-scooter mobility systems. Most 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 work that mainly influenced our proposal is the one by Ciociola et al. [7], in which the ODySSEUS simulator has been introduced (Section 4) — an essential component of the ESB-DQN system.

Ciociola et al. created a flexible, data-driven demand model by using modulated Poisson processes for temporal estimation, and Kernel Density Estimation for spatial estimation. The demand model was used in conjunction with a configurable e-scooter sharing simulator to compare performance of different electric scooter sharing design options, such as the impact of the number of scooters and the cost of managing their charging. Experiments were carried out on open data about e-scooter sharing rides in Minneapolis and Louisville.

Table 1 summarizes the main features of both the systems discussed in this subsection and ESB-DQN.

2.3. Related works on urban decorum, dockless e-scooter flows, self-repositioning

Other works that are related with ours albeit focusing on different issues, include the study by Carrese et al. [5] who observe that leaving electric vehicles in the wrong position not only interferes with pedestrians and other vehicles, but also reduces urban decorum. They suggest introducing "beautificators", namely employees that fix inappropriate and disordered parking made by users, by moving vehicles close to the place where they had been left, but in safer, and more acceptable from the urban decorum point of view, place. The repositioning that they propose is different by traditional relocation made to rebalance fleets; their approach could be merged with ours, by recognizing more incentives

to users who park their vehicles in the right zone, and also in the right way.

In a paper published in 2021, He and Shin [13] analyze e-scooter distribution features and flow dynamics for the data-driven designs, and propose a novel spatio-temporal graph capsule neural network to predict future dockless e-scooter flows given the reconfigured regions. They the data from the Louisville database of trips that we also employed (see Section 3), plus data from other cities in the US. Although by predicting e-scooter distribution one might also devise strategies for rebalancing, He and Shin do not address rebalancing as their main goal. Also, by exploiting neural networks, they use a different learning approach.

A visionary development of fleet rebalancing involves self-repositioning shared personal mobility devices [17], namely small electric vehicles that can move autonomously at slow speed to reposition themselves, but require to be driven by their user during trips. While the adoption of such self-repositioning devices would boost the efficiency of fleet operations, there are many obstacles to their application ranging from engineering challenges such as self-stabilization of two-wheels vehicles [12] to infrastructural requirements, to regulatory issues involving self-driving electric vehicles.

3. Data used in this study

To investigate the free-floating imbalance problem, we rely on actual e-scooter trips open data published by the Municipality of Louisville[6], 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 [7] 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 2. They refer to a *training set* of observations registered over the whole year 2019. The number of trips we used in the simulation, denoted as *N trips sim*, refers to observations used in the simulation, namely those that took place on January 01, 2020.

[6] `https://data.louisvilleky.gov/dataset/dockless-vehicles`, accessed on January 10th, 2022.

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 [7], with l the side of the squares, a parameter later clarified in Section 6. 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 set* of observations. These zones are valid pick-up/drop-off alternatives because — being adjacent to them — they are not too distant from the pick-up/drop-off points that the user had planned to start from/end to, respectively. As it can be seen in Figure 2, 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 $\mathcal{N}^\star$, with $|\mathcal{N}^\star| = 279 * 8 = 2232$.

Both the large subset of invalidity, $\mathcal{N}_{invalid}$, and the small number of observations used in the simulation (over January 01, 2020 only), are further discussed in Section 6 as they play a key role in the reasoning behind the training of the ESB-DQN multi-agent system.

4. Simulator of the e-scooter sharing dynamics

A modified version of ODySSEUS, the SimPy-based simulator presented in [7], has been used to simulate e-scooter sharing system dynamics in Louisville. A formal description of the simulator follows; the most relevant symbols used in this section are summarized in Table 4, along with the name of the corresponding parameters in the implemented simulator, when available.

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 $b(s) \in [0, \beta]$, with β 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

Table 2
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

Table 3
Summary of most relevant symbols used in this section and corresponding parameters in the simulation (if they exist).

Parameter	Symbol in formulae	Parameter's name in the simulator
Pick-up agent	P	(none)
Pick-up agent action at time t	$a_{P,t}$	(none)
Drop-off agent	D	(none)
Drop-off action at time t	$a_{D,t}$	(none)
Pick-up zone of the ith generated trip request	$p(i)$	(none)
Drop-off zone of the ith generated trip request	$d(i)$	(none)
Alternative pick-up zone of the ith generated trip request	$\hat{p}(i)$	(none)
Alternative drop-off zone of the ith generated trip request	$\hat{d}(i)$	(none)
Side length of the square zones	l	`bin_side_length`
Number of vehicles in the environment	$\|\mathcal{S}\|$	`n_vehicles`
Number of battery swap workers	n_{swap}	`n_workers`
Number of relocation workers	n_{rel}	`n_relocation_workers`
Acceptance probability for each incentive proposal	w	`incentive_willingness`
Battery capacity	β	`beta`
Battery level to mark vehicles as *dead* (percentage of β)	α	`alpha`

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 identified by adjacent zones ("1-hop neighbouring zones"). 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 $b(s) < \alpha\beta$, 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:

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

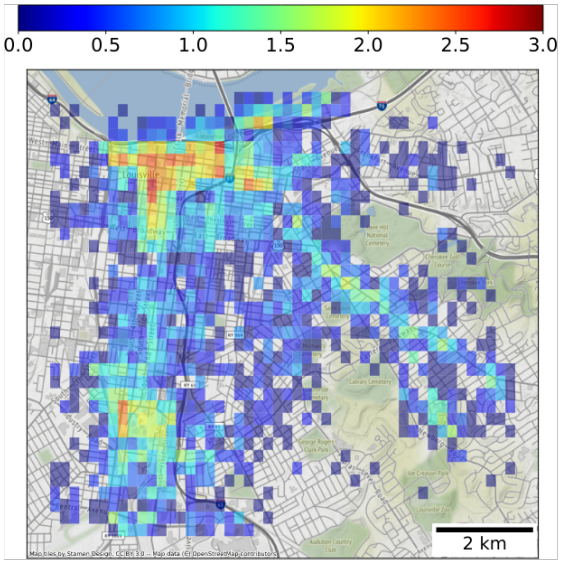

Fig. 2. Characterisation of zones in the city of Louisville, following a heatmap colour scale on the number of trips starting form each zone in 2019 in a decimal logarithmic scale (exponent reported in the legend). The warmer the color, the higher the number of trips starting from the zone. Credits to [7].

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, $P_{swap,t+T_s}$, higher than the expected battery swap costs, $C_{swap,t}$: 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 Traveling Salesperson Problem optimization. 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 [7]. Relocation time is modeled in the system following the original approach of the ODySSEUS simulator, with minor adjustments: the ESB-DQN parameters include the average time employed by any relocator to put the e-scooter onto the van used for the relocation task, and the average time employed by any relocator to reach the destination point. Values for these parameters were set as in the ODySSEUS simulator.

Initialization. At start time, e-scooters are randomly placed among the zones of the grid with uniform random charge $b(s) \in [\beta/2, \beta]$. 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.

In the ODySSEUS simulator, 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 our 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.

5. The 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.

Intuitively, if the system recognizes that there is even a slight imbalance between $p(i)$ (or $d(i)$) and some nearby zones, the respective agent, P (or D), is encouraged to offer the users a proposal in change of an incentive based on the added Euclidean distance. Or again, if there are a certain number of out of charge vehicles gathered in some other areas, and the clock t is close to the next service schedule, $(k+1) * T_s$, then both agents are likely to offer the users a proposal.

The next three paragraphs describe the fundamental components of the ESB-DQN multi-agent system.

5.1. Environment

The environment wraps the modified simulator described in Section 4 to make it compliant with DeepMind's DQN Zoo library for Reinforcement Learning [8]. The major change we have made to the ODySSEUS 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 [7] does, our 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:

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

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 4, and C is a binary $n \times 1$ column vector with 1s in correspondence of $p(i)$ and $d(i)$ only, indicating if a certain zone is the start or the end zone of the trip (the position corresponding to some zone z contains 1 in vector C if $z == p(i)$ or $z == d(i)$).

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.

In our initial design of the system, we planned the ESB-DQN environment to emit a fourth column vector, D, with the number of dead vehicles per zone z at time t. Indeed, we believed that D could enhance the agents' understanding of the need of re-balancing of z, whereas A and B alone would already take care of the trend of usage of z. Eventually, since this information is highly negatively correlated with A, we opted to leave it out.

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 [25], namely the integration of constraint models and RL techniques, such that the RL agent, also the learner, is guided by an encapsulating constraint model that describes safety constraints for the agent's task. 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, Spieker [25] shows 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 limi-

tation 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 the one by Spieker, 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 our setting, invalid actions for the agents include proposing zones that are not valid in the operational area, proposing a drop-off zone that is identical to the pick-up one or vice-versa, suggesting a pick-up zone where no suitable vehicles are available. In Section 5.4 we further explore this continuity while training the ESB-DQN system, by introducing the concept of *lives*, borrowed from Atari games [20].

5.2. Agents architecture

The agents are Deep-Q-Networks (DQN) implementing an ϵ-greedy policy with experience replay [20] 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 [14], 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 P and the drop-off agent D 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 2, 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 (for example, many zones that the agent may select are not inside the e-scooter operational boundaries in the Louisville city, a pick-up zone may be identical to the drop-off one), which is further evidence of the need for outer aid for the RL agents to well characterize the dynamics of the system.

5.3. Reward

The rationale behind our reward function is the following, with Δ being the expected deficit of e-scooters in the zone and time under consideration:

- **Pick-up case with $\Delta < 0$.** If the agent confirms the pick-up when there is an expected surplus of vehicles, then the reward is positive. As delta decreases the reward increases because the surplus condition of the zone is improved. Similarly, if the number of available vehicles is high, then the rebalancing for that area is encouraged. As the demand increases, the reward decreases because this may negatively affect the long-term balance of the zone.
- **Pick-up case with $\Delta > 0$.** If the agent confirms the pick-up when there is an expected deficit of vehicles, then the reward is negative. As delta and the demand increase, the negative reward is larger because the deficit condition of the zone is worsened. In both cases if the number of available vehicles is higher, then the curves rise more slowly because this can have a less negative impact on the long-term deficit of vehicles in the zone.
- **Drop-off case with $\Delta > 0$.** If the agent suggests the drop-off when there is an expected deficit of vehicles, then the reward is positive. As delta or the demand increases the reward increases and the curve flattens out more smoothly, i.e., with larger

delta or with larger demand the deficit condition will be more difficult to solve, and as a consequence the reward is higher than the number of vehicles currently available.

– **Drop-off case with $\Delta < 0$.** If the agent suggests the drop-off when there is an expected surplus of vehicles, then the reward is negative. As delta decreases the reward is smaller, and the curve falls down faster, i.e. if delta is larger, the surplus condition of the zone will be more difficult to solve, thus the reward falls down based on the number of vehicles currently available. As the demand increases there is a less negative reward because the surplus condition could be useful even if an additional vehicle is added to the zone. For this reason the curve falls down less quickly.
– **Dead-vehicle case at the end of the trip.**
  * **Drop-off case:** the larger the number of dead vehicles the larger the positive reward because the efficiency of battery swap operations may be improved. The reward decreases with the number of available vehicles because this increases the imbalance of the zone (caused by large numbers of available and dead vehicles in the same zone).
  * **No Drop-off case:** the larger the number of dead vehicles the larger the negative reward because the efficiency of battery swap operations may be decreased. The reward increases with the number of available vehicles because this can avoid the imbalance of the zone (caused by large numbers of available and dead vehicles in the same zone).

The following functions are defined to compute the reward:

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

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

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 $t+\Delta t$);
– $N_A(z,t)$: 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:

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

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:

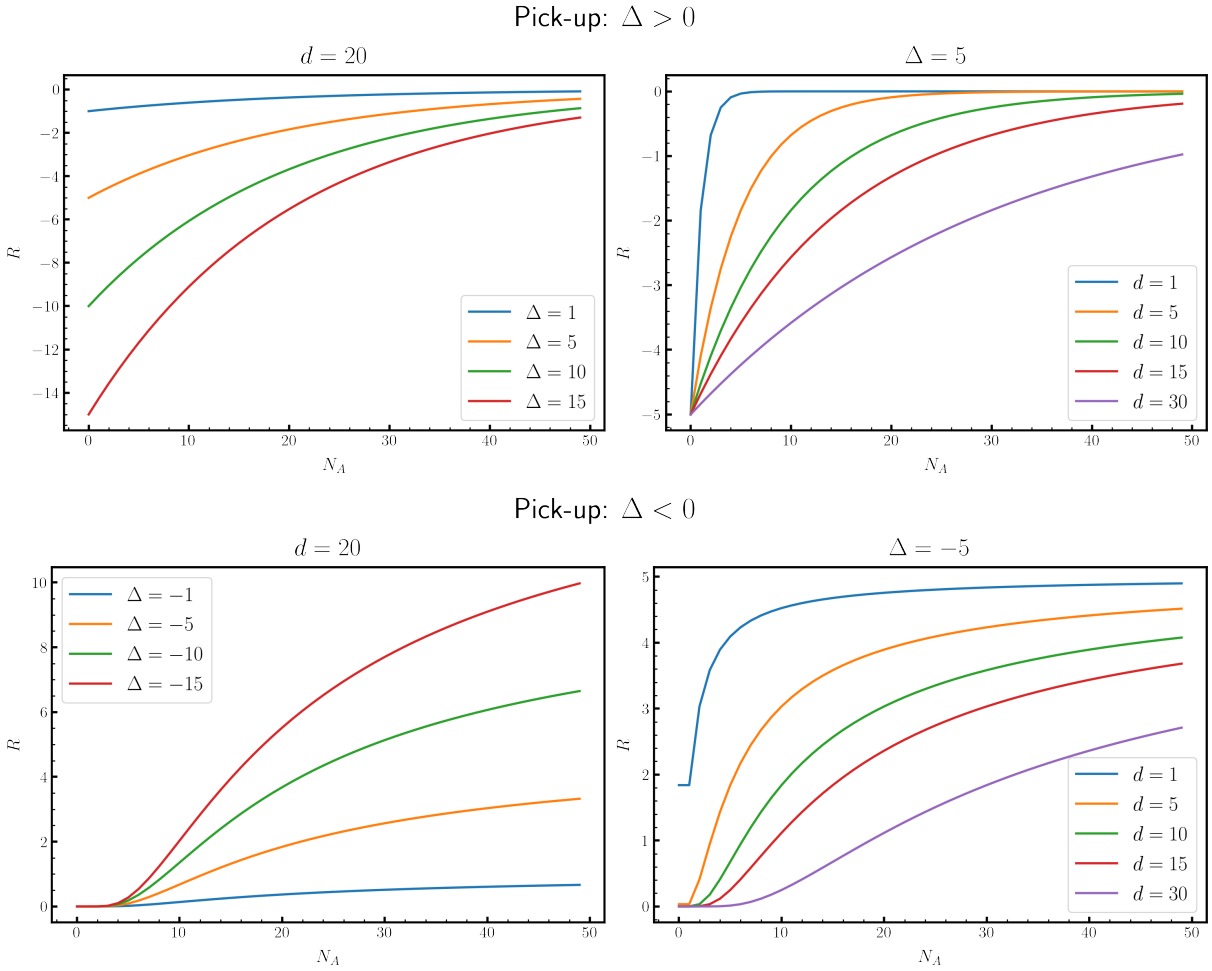

Fig. 3. Pick-up action: some examples of the reward function R_P for different values of the parameters Δ, d. **(Upper row)** Negative reward: the agent suggests to pick-up a vehicle from a zone having an expected deficit of vehicles ($\Delta > 0$). As Δ and d increase, the reward is smaller because 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.

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

with γ_P being a constant which modulates the penalty of an invalid pick-up action. Figure 3 shows some examples of the reward function R_P for different values of the parameters Δ, d.

5.4. *Lives mechanism*

As we have anticipated in Section 5, 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 vice-versa.

6. 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 motivating users to pick-up/drop-off vehicles in alternative zones can preserve a good number of satisfied trips with a reduced number of relocation and battery swap workers.

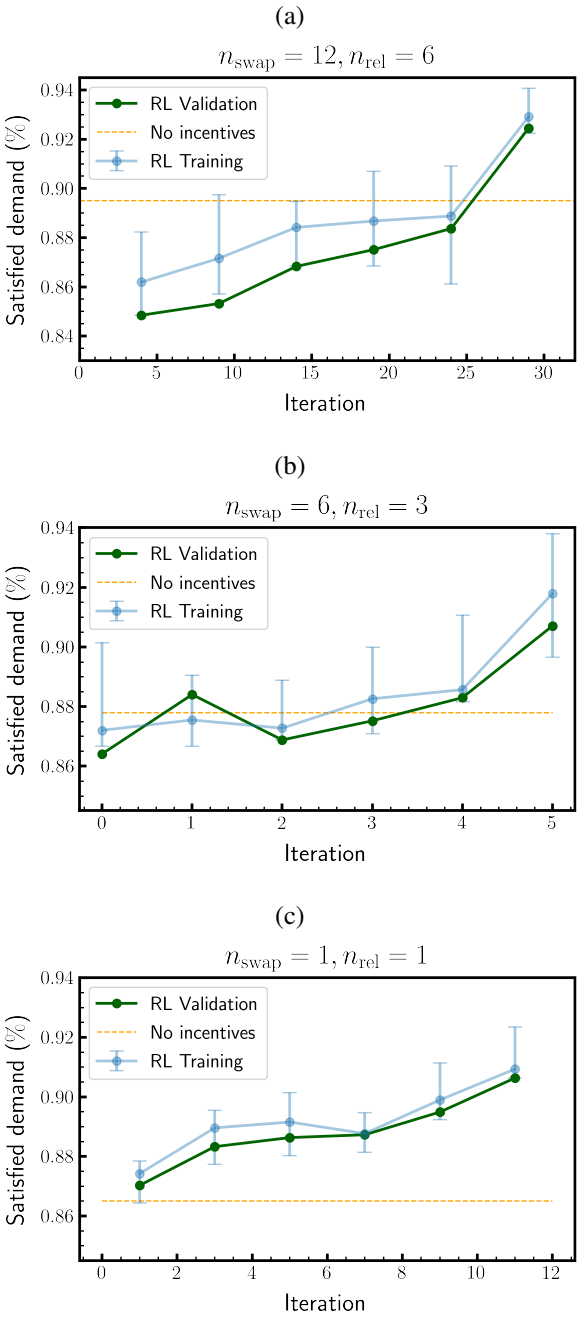

Fig. 4. Evaluation of the satisfied demand during the learning procedure for training and validation agents versus a baseline model with no incentive policy (user willingness $w = 0$). The value of the parameters is shown in the titles. **(a)** Model trained from scratch, **(b), (c)** Transfer learning.

Table 4
Results of the experiments.

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

We measured the satisfied demand D_{sat}, defined as follows:

$$D_{\text{sat}} = \frac{N_{\text{trips}} - N_{\text{unsat}}}{N_{\text{trips}}}$$

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.

6.1. Parameters of the simulator

The parameters of the simulator have been set as follows:

- the number of available e-scooters is $|\mathcal{S}| = 400$ (almost half of the Louisville fleet, that amounts to 850 vehicles, scaled down having a medium size city like Genova in mind);
- the size of the zones is $l = 200m$ (value often found in the literature [7,9]: given that alternatives among adjacent zones are proposed to users, this value means that — in the worst case where adjacent zones have just one corner in common — the user might need to face a $2 * 200 * \sqrt{2} = 566m$ walk from the planned pick-up/drop-off point to the most distant point in the alternative zone, walk that can be acceptable by an e-scooter driver that, according to the literature [1,3], is usually young);
- the battery capacity is $\beta = 425$ 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 (realistic data, as suggested in [7]);
- the user willingness is $w = 0$ for obtaining the "No incentives" baseline data, and $w = 1$ for obtaining the other data discussed in this section;
- the battery swap operations are scheduled every $T_s = 1$h (realistic value, considering the number of trips and vehicles);
- the relocation operations are scheduled every $T_r = 1$h in a working time interval $T_{\text{work}} = [9\text{AM-}6\text{PM}]$ (realistic value, considering the number of trips and vehicles).

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 satisfied demands would remain strong nonetheless (88%). The latter has been set to 1 for measuring the advantage of an incentives mechanism, 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. Baseline results have been obtained by setting willingness to 0.

6.2. Parameters of the Reinforcement Learning system

The parameters of the Reinforcement Learning system have been set as follows:

- the optimizer is Adam [16] 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 $\Delta t = 1$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.

6.3. Results of the experiments

In the first experiment, the model has been trained from scratch with the number of battery swap workers being $n_{\text{swap}} = 12$ and the number of relocation 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 4. The evolution of the satisfied demand during the learning procedure is represented in Figure 4.

Our ratio between validation and training trips is $1 : 2$: 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 4(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 4(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 in the system rebalancing and contributing to a positive increase of the satisfied demands.

7. 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 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 ODySSEUS 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 have shown encouraging results on the satisfied demand of the system, even with a strongly reduced number of workers. To assess the impact of the presence/absence of some elements and parameters and to increase our confidence in the current results an ablation study should be carried out. In the future we plan to evaluate how the system performs without the lives mechanism, to assess the impact and benefits of having them. 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 ac-

curate 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.

As a future work, it would also be worth studying the (monetary) amount of the incentive, as an example: give a lower incentive if the balance is not heavily affected by the particular trip (and so the user willingness to accept is low) while giving a higher incentive if a new pick-up or drop-out zone can heavily ameliorate the balancing (thus a higher chance that the user accept the new plan proposed by the RL agent).

A fundamental effort should be devoted to scale-up experiments on a larger temporal scale and on larger datasets (e.g., Austin open data [2]). A larger number of iterations would 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 for the future 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. Currently, many sharing electronic vehicles systems take advantage of chatbots for assisting their users. As the chatbot market is expected to grow at a compound annual growth rate of around 25% during 2021-2026 [15], making the services offered by our ESB-DQN system accessible through a natural language interface is part of our long-term vision.

Besides the many existing frameworks for building chatbots from scratch, including PandoraBots[7], DialogFlow[8], Wit.ai[9] , tools targeted towards the urban mobility domain exist, such as MindSay[10], also exist.

A preliminary working prototype of the chatbot that should provide an interface towards the user, and should also be extended to deal with pick-up and drop-off proposals, has been built using DialogFlow, able to answer questions inspired to the FAQ of the GoVolt company[11]. The "where-can-i-circulate" and the "can-another-user-ride-my-scooter" intents and the sentences used to train them are shown in Figure 5.

Acknowledgements

We acknowledge the anonymous reviewers for their thorough reading and constructive comments.

References

[1] Álvaro Aguilera-García, Juan Gomez, and Natalia Sobrino. Exploring the adoption of moped scooter-sharing systems in spanish urban areas. *Cities*, 96:102424, 2020.

[2] Austin shared micro-mobility open data, 2019.

[3] Tomasz Bieliński and Agnieszka Ważna. Electric scooter sharing and bike sharing user behaviour and characteristics. *Sustainability*, 12(22):9640, 2020.

[4] Anh Bui, Peter Slowik, and Nic Lutsey. Evaluating electric vehicle market growth across U.S. cities. In *The International Council on Clean Transportation*, 2021.

[5] Stefano Carrese, Fabio D'Andreagiovanni, Tommaso Giacchetti, Antonella Nardin, and Leonardo Zamberlan. A beautiful fleet: Optimal repositioning in e-scooter sharing systems for urban decorum. *Transportation Research Procedia*, 52:581–588, 2021. 23rd EURO Working Group on Transportation Meeting, EWGT 2020, 16-18 September 2020, Paphos, Cyprus.

[6] Annie YJ Chang, Luis Miranda-Moreno, Regina Clewlow, and Lijun Sun. Trend or fad? deciphering the enablers of micromobility in the US. *SAE International*, 2019.

[7] Alessandro Ciociola, Michele Cocca, Danilo Giordano, Luca Vassio, and Marco Mellia. E-scooter sharing: Leveraging open data for system design. In *2020 IEEE/ACM DS-RT*, pages 1–8. IEEE, 2020.

[8] DeepMind. DQN Zoo, 2020.

[9] Yubin Duan and Jie Wu. Optimizing rebalance scheme for dock-less bike-sharing systems with adaptive user incentive. In *2019 IEEE MDM*, pages 176–181. IEEE, 2019.

[7] `https://home.pandorabots.com/home.html`, accessed on January 10th, 2022.

[8] `https://cloud.google.com/dialogflow/`, accessed on January 10th, 2022.

[9] `https://wit.ai/`, accessed on January 10th, 2022.

[10] `https://www.mindsay.com/chatbot/urban-mobility`, accessed on January 10th, 2022.

[11] `https://www.govolt.it/en/faq/`, accessed on January 10th, 2022.

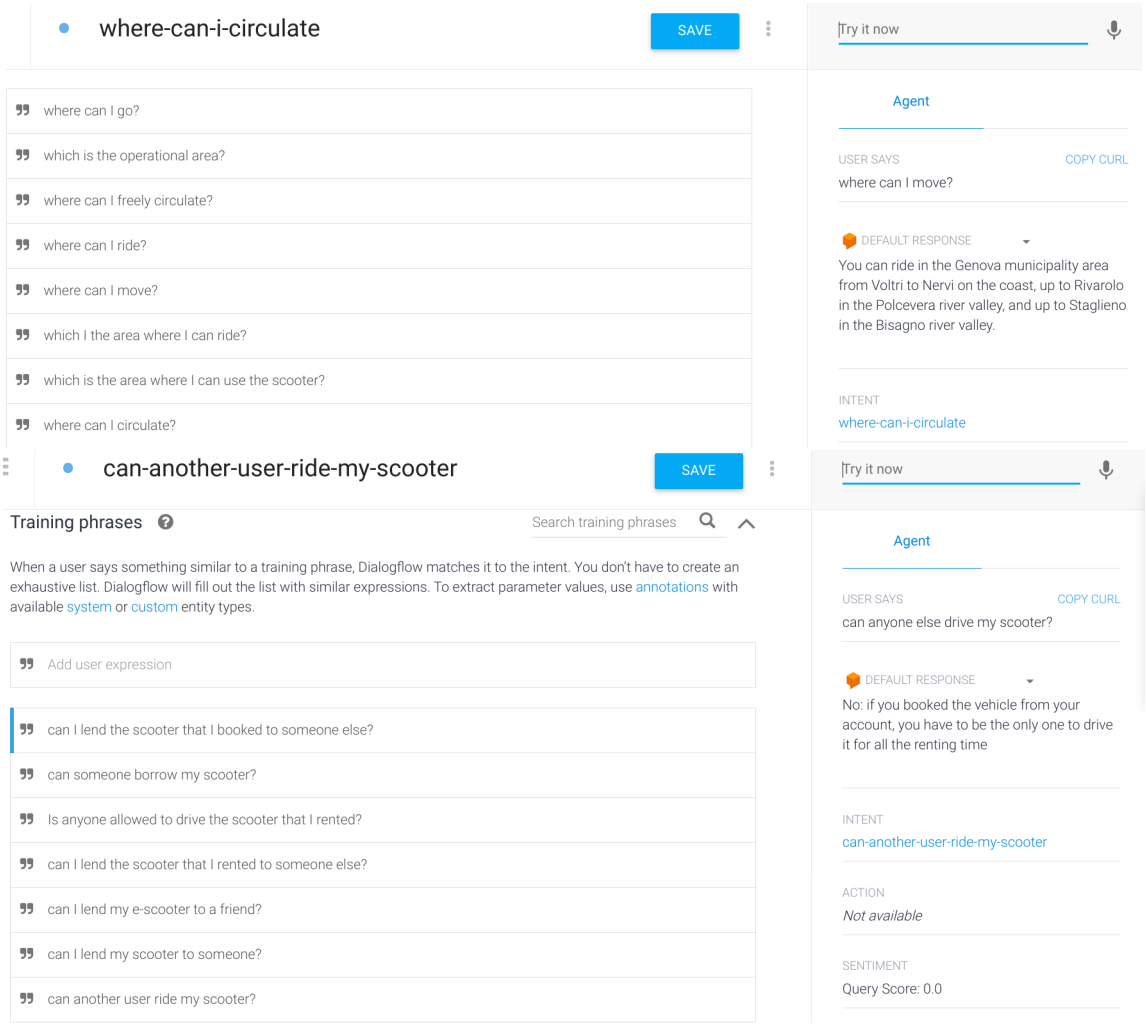

Fig. 5. Example of sentences that can be used inside DialogFlow to train the chatbot to answer questions related with the boundaries of the e-scooter operational zone, and with possibility to lend the rented e-scooter.

[10] Timo Eccarius and Chung-Cheng Lu. Adoption intentions for micro-mobility – insights from electric scooter sharing in Taiwan. *Transportation research part D: transport and environment*, 84:102327, 2020.

[11] Christine Fricker and Nicolas Gast. Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO Journal on Transportation and Logistics*, 5(3):261–291, 2016.

[12] M. Ramos García, Daniel A. Mántaras, Juan C. Álvarez, and David Blanco F. Stabilizing an urban semi-autonomous bicycle. *IEEE Access*, 6:5236–5246, 2018.

[13] Suining He and Kang G. Shin. Distribution prediction for reconfiguring urban dockless e-scooter sharing systems. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2021.

[14] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017.

[15] IMARC. Chatbot Market: Global industry trends, share, size, growth, opportunity and forecast 2021-2026. `https://www.imarcgroup.com/chatbot-market`, accessed on January 10th, 2022., 2020.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Dániel Kondor, Xiaohu Zhang, Malika Meghjani, Paolo Santi, Jinhua Zhao, and Carlo Ratti. Estimating the potential for shared autonomous scooters. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2021.

[18] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. Rebalancing bike-sharing systems: A multi-source data smart optimization. In *Proceedings of the 22nd ACM SIGKDD Inter-*

national Conference on Knowledge Discovery and Data Mining, KDD '16, page 1005–1014, New York, NY, USA, 2016. Association for Computing Machinery.

[19] Gianvito Losapio, Federico Minutoli, Viviana Mascardi, and Angelo Ferrando. Smart balancing of e-scooter sharing systems via deep reinforcement learning. In Roberta Calegari, Giovanni Ciatto, Enrico Denti, Andrea Omicini, and Giovanni Sartor, editors, *Proceedings of the 22nd Workshop "From Objects to Agents", Bologna, Italy, September 1-3, 2021*, volume 2963 of *CEUR Workshop Proceedings*, pages 83–97. CEUR-WS.org, 2021.

[20] Volodymyr Mnih, Koray Kavukcuoglu, and other authors. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[21] Ling Pan, Qingpeng Cai, Zhixuan Fang, Pingzhong Tang, and Longbo Huang. A deep reinforcement learning framework for rebalancing dockless bike-sharing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1393–1400, 2019.

[22] Julius Pfrommer, Joseph Warrington, Georg Schildbach, and Manfred Morari. Dynamic vehicle redistribution and online price incentives in shared mobility systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1567–1578, 2014.

[23] Robert Regue and Will Recker. Proactive vehicle routing with inferred demand to solve the bike-sharing rebalancing problem. *Transportation Research Part E: Logistics and Transportation Review*, 72:192–209, 2014.

[24] C. Scott Smith and Joseph P. Schwieterman. E-scooter scenarios: Evaluating the potential mobility benefits of shared dockless scooters in Chicago. Available online., 2018.

[25] Helge Spieker. Constraint-guided reinforcement learning: Augmenting the agent-environment interaction. *CoRR*, abs/2104.11918, 2021.

[26] Peiyu Yi, Feihu Huang, and Jian Peng. A rebalancing strategy for the imbalance problem in bike-sharing systems. *Energies*, 12(13), 2019.

[27] Rui Zhu, Xiaohu Zhang, Dániel Kondor, Paolo Santi, and Carlo Ratti. Understanding spatio-temporal heterogeneity of bike-sharing and scooter-sharing mobility. *Computers, Environment and Urban Systems*, 81:101483, 2020.

Appendix: Configuration of the simulator parameters and short user's guide

ESB-DQN repository on GitHub

`https://github.com/DiTo97/odysseus-escooter-dqn`

Input parameters

Input parameters take into account both simulation parameters of the ODySSEUS environment and parameters for the training/test of the Rainbow agents.

Simulation parameters

Simulation parameters can be found in the folder `esbdqn\configs\escooter_mobility` under the name `sim_conf_<City>.py` with identical structure. Each file comprises two Python objects, named `General` and `Multiple_runs`.

General object

- `city`, name of the city, either Louisville or Austin;
- `relocation_workers_working_hours`, shift hours for relocation workers;
- `bin_side_length`, side length of the square zones each operative area is split into;
- `year`, year of the trip requests to consider;
- `month_start`, `month_end`, start and end of the month of the trip requests to consider;
- `day_start`, `day_end`, start and end of the day of the trip requests to consider;
- `save_history`, whether to save the results CSV after each iteration.

Multiple_runs object

- `n_vehicles`, number of vehicles to spawn in the environment;
- `incentive_willingness`, acceptance probability for each incentive proposal;
- `beta`, battery capacity;
- `alpha`, threshold on the battery level to mark vehicles as out-of-charge in percentage between 0 and beta;
- `battery_swap`, toggle for battery swap events in the environment, either True or False;
- `n_workers`, number of battery swap workers;
- `battery_swap_capacity`, maximum number of vehicles each battery swap worker can process hourly;
- `scooter_relocation`, toggle for relocation events in the environment, either True or False;
- `n_relocation_workers`, number of relocation workers;
- `relocation_capacity`, maximum number of vehicles each relocation worker can move hourly;

All the parameters that have not been modified or are unused with respect to the original ODySSEUS simulator have been omitted here.

Agents parameters

Agents parameters can be found in the file `esbdqn\train.py`. Also, they can be submitted at runtime when launching `esbdqn\train.py` via `CLI`.

- `learning_rate`, learning rate of the Adam optimizer;
- `learn_period`, learning period of the Rainbow agents;
- `batch_size`, batch size of the networks withing the agents;
- `n_steps`, how many steps to look in the past when agents take decisions;
- `max_global_grad_norm`, global gradient norm clipping of the networks weights;
- `importance_sampling_exponent_begin_value` (and similar parameter with `end_value`), range of the importance sampling exponent;
- `replay_capacity`, experience replay buffer capacity. Should amount to about 30 repetitions of any given day;
- `priority_exponent`, priority of the timesteps stored in the experience replay buffer;
- `target_network_update_period`, update period from the online network to the offline network within each agent;
- `num_iterations`, number of training iterations;
- `max_steps_per_episode`, number of trips per episode;
- `num_eval_frames`, total number of validation trips per iteration;
- `num_train_frames`, total number of training trips per iteration (should be at least double the validation trips);
- `n_lives`, total number of lives per iteration; defaults is 50.

Experiment parameters

Each call of `esbdqn\train.py` can be named as a different experiment with its own checkpoints.

- `exp_name`, name of the experiment directory;
- `checkpoint`, toggle on whether to store a checkpoint, either True or False;
- `checkpoint_period`, period of storage of a new training checkpoint.

Output

Run `esbdqn\train.py` to train a new ESB-DQN model from scratch. Otherwise, to train starting from a checkpoint, set the checkpoint toggle to True, and ensure that there is a checkpoint within the experiment directory in the form: `<Experiment_dir>\models\ODySSEUS-<City>`.

Results of each run will be stored as CSV files within the automatically generated directory `<Experiment_dir> \results`.

To reproduce the experiments in the paper:

1. Set `incentive_willingness` to 0 to obtain all the No incentives data.
2. Set `incentive_willingness` to 1 and track the columns `eval_avg_pct_satisfied_demand` and `train_avg_pct_satisfied_demand` from the CSV files for the Validation and Training data, respectively.

All our experiments have been run on Ubuntu 18.04.