A Cost-aware Adaptive Bike Repositioning Agent using Deep Reinforcement Learning

Alessandro Staffolani, Victor-Alexandru Darvariu, Paolo Bellavista and Mirco Musolesi

***Abstract*—Bike Sharing Systems (BSS) represent a sustainable and efficient urban transportation solution. A major challenge in BSS is repositioning bikes to avoid shortage events when users encounter empty or full bike lockers. Existing algorithms unrealistically rely on precise demand forecasts and tend to overlook substantial operational costs associated with reallocations.**

This paper introduces a novel *Cost-aware Adaptive Bike Repositioning Agent* (CABRA), which harnesses advanced deep reinforcement learning techniques in dock-based BSS. By analyzing demand patterns, CABRA learns adaptive repositioning strategies aimed at reducing shortages and enhancing truck route planning efficiency, significantly lowering operational costs.

We perform an extensive experimental evaluation of CABRA utilizing real-world data from Dublin, London, Paris, and New York. The reported results show that CABRA achieves operational efficiency that outperforms or matches very challenging baselines, obtaining a significant cost reduction. Its performance on the largest city comprising 1765 docking stations highlights the efficiency and scalability of the proposed solution even when applied to BSS with a great number of docking stations.

***Index Terms*—Dynamic Bike Repositioning, Reinforcement Learning, Resource Allocation.**

I. Introduction

BIKE Sharing Systems (BSS) have emerged as a pivotal solution for enhancing the efficiency and environmental sustainability of urban transportation, effectively addressing the *last mile* challenge in cities [1]. As urban populations swell, the demand for seamless integration of these systems with existing transportation infrastructures becomes increasingly vital. BSS, popular for their affordability and eco-friendliness, offer commuters the convenience of short-term bike rentals [2]–[4]. In recent years, the BSS landscape has evolved, with cities adopting both docked and dockless models to meet diverse urban mobility needs. While docked systems, common in cities like London, Paris, and New York, depend on fixed docking stations, dockless models offer greater flexibility but at increased management costs. However, both models grapple with the challenge of maintaining optimal bike availability, particularly reducing *shortage events*, where users may encounter empty stations or full lockers [5].

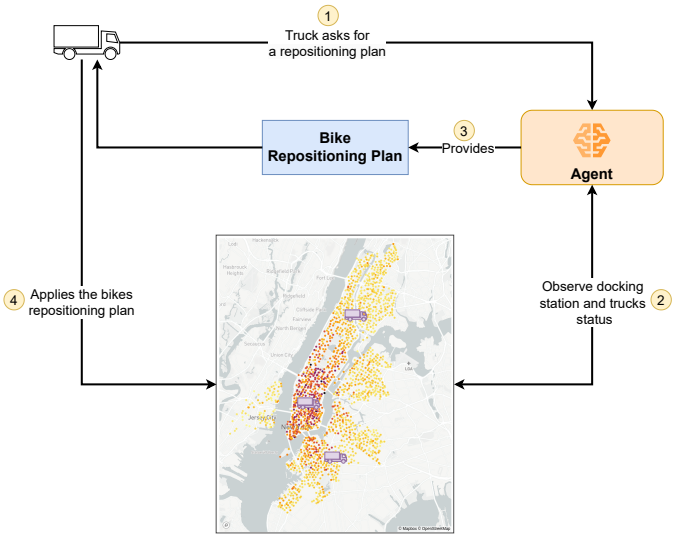

Fig. 1. Graphical summary of the CABRA rebalancing pipeline for dock-based dynamic bike repositioning. Trucks ask for a new repositioning. The *agent* observes docking stations and truck status and then provides the *bike repositioning plan*, which is applied to the system.

In response to these shortages, BSS operators employ expensive yet potentially inefficient rebalancing techniques, which can be broadly categorized into two main groups: *user*-based and *vehicle*-based approaches. User-based approaches involve providing incentives, often in the form of monetary rewards, to users who relocate bikes from congested stations to less congested ones. In contrast, vehicle-based approaches utilize fleets of trucks that traverse the BSS coverage area, both *picking up* bikes from congested stations and *dropping* them off at less congested stations. Vehicle-based approaches can be further classified into two subcategories: *static* and *dynamic* approaches. The former focuses on applying rebalancing policies exclusively during designated cooldown hours, typically occurring at night. In contrast, dynamic approaches constantly implement daily rebalancing policies to address station imbalances effectively.

To further enhance the efficiency of these systems and ensure their seamless integration into urban transportation networks, there is a pressing need for more optimized and proactive solutions [5] and not only static ones [6], [7]. This involves leveraging real-time data to optimize the redistribution of bikes across docking stations, as performed in an increasing number of applications of machine learning for

Manuscript received XX XX, XX; revised XX XX, XX.

Alessandro Staffolani, Paolo Bellavista and Mirco Musolesi are with the Department of Computer Science and Engineering, University of Bologna, Bologna, Italy. E-mail: alessandro.staffolani, paolo.bellavista, mirco.musolesi@unibo.it.

Alessandro Staffolani is also with the National Council of Research Italy, ISTI-CNR, Pisa, Italy. E-mail: alessandro.staffolani@isti.cnr.it.

Victor-Alexandru Darvariu is with the Oxford Robotics Institute, Department of Engineering Science, University of Oxford, Oxford, UK. E-mail: victord@robots.ox.ac.uk. This work was performed while the author was affiliated with University College London.

Mirco Musolesi is also with the Department of Computer Science, University College London, London, UK. E-mail: m.musolesi@ucl.ac.uk.

transportation management [8], [9]. Such strategies aim to proactively alleviate the impact of shortage events by ensuring an appropriate distribution of bikes throughout the network. Current rebalancing policies often rely on two aspects: (1) predictions of future demand based on past data; and (2) heuristic techniques for deciding reallocations. With respect to the former, predictions may be inaccurate in cases where demands diverge from historical trends, meaning bikes may be reallocated to the wrong stations. Regarding the latter, heuristic algorithms perform suboptimally, and better algorithms for deciding a repositioning plan may exist.

Given the shortcomings of prior exact and heuristic methods, we set out to address this problem using reinforcement learning, whose potential has been demonstrated recently for a variety of discrete optimization problems [10]. This does not require accurate predictions of future demands and can enable the implicit discovery of heuristic algorithms that are more powerful than existing alternatives through its reward-driven, trial-and-error mechanism. We opt for a *deep* reinforcement learning technique because the state space quickly increases in larger cities, and function approximation is required for effective generalization and scalability.

We term the proposed method *Cost-aware Adaptive Bike Reposition Agent (CABRA)*. Unlike traditional vehicle-based approaches, CABRA efficiently learns and adapts repositioning strategies by analyzing real-time demand patterns. The learned policies allow our approach to mitigate shortages proactively while being capable of reacting to rapid unseen changes in demand. In addition, a key novelty of CABRA is the joint minimization of shortage events and movement costs, representing a substantially more realistic model for BSS operators. CABRA employs deep reinforcement learning complemented by pruning rules, thus achieving scalability to over a thousand of docking nodes, as we demonstrate in the following parts of this paper.

Our Contributions. The key contributions of this paper can be summarized as follows:

- **Dynamic bike reposition modeling for dock-based BSS**: We present an innovative approach to the bike repositioning challenge in dock-based BSS, framing it as a decision-making process. This involves deploying trucks to rebalance bike availability across urban docking stations. Our model utilizes an agent that processes both the current system status and truck locations to formulate an efficient repositioning plan, as detailed in Figure 1. The plan specifies the number of bikes to be transported and the target docking stations. Guided by a numerical reward signal, the agent's decisions aim to minimize bike shortages and optimize truck repositioning times, thereby enhancing route efficiency and reducing BSS management costs. We formulate this problem within the Markov Decision Process (MDP) framework and employ deep reinforcement learning for its resolution.
- **Extensive evaluation with comprehensive datasets**: We conduct a thorough evaluation of our solution using real-world datasets. It offers an in-depth view of major BSS operators in Dublin, London, Paris, and New York, including 10-minute interval observations of docking station statuses and essential static information such as station locations and capacities. This evaluation rigorously tests the scalability and generalizability of our solution across diverse BSS landscapes, providing valuable experimental insights and firmly establishing its applicability in practical scenarios.

Main Results. Our extensive experiments using real-world datasets from major bike-sharing providers reveal that CABRA consistently outperforms traditional methods in terms of operational efficiency and cost optimization in three out of four cities. Specifically, CABRA demonstrates a minimum improvement over the second-best baseline of 27% in Dublin, 24% in London, and an increase of 42% in New York, even though the latter BSS contains over a thousand stations. However, in environments with highly irregular demand patterns such as Paris, our results indicate that greedy solutions emerge as the only viable alternative to CABRA's proactive and adaptive strategy, which nevertheless performs comparably.

II. Related Work

The bike repositioning problem, when viewed in a broader scope, aligns with the intricate nature of combinatorial problems, necessitating sophisticated decision-making and optimization strategies. For a comprehensive understanding of the role of machine learning in combinatorial optimization, Bengio et al. (2021) [10] offer an extensive survey, while Mazyavkina et al. (2021) [11] focus specifically on the application of reinforcement learning in this area. Furthermore, the complexities of the bike repositioning problem share notable parallels with the challenges seen in the Travelling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP), which also involve devising routes that visit a set of points. The applicability of reinforcement learning to these problems is well-documented in studies such as [12]–[14].

Delving specifically into bike repositioning, the primary distinction lies between *static* and *dynamic* approaches. For example, the authors of [5] formulated the static bicycle repositioning problem as a VRP, introducing a Mixed Integer Linear Program that optimizes user shortages and operational costs. The effectiveness of their model was validated using real-world data from Washington DC and Paris BSS. [15] tackled the static bike repositioning problem in dock-based BSS by framing it as a many-to-many pickup and delivery problem, using a single truck per district. They proposed a branch-and-cut algorithm to address the NP-hard problem, supplemented by a tabu search to obtain an upper bound for the optimal solution. The approach was evaluated using synthetic data. Lastly, [16] combined inventory rebalancing with vehicle routing for static BSS, formulating a Mixed Integer Program that optimizes bike availability and rebalancing routes, demonstrating superior performance over conventional methods with real-world data.

In contrast, dynamic solutions to the bike repositioning problem can be classified into different categories based on the target system, such as *dock-based* and *dockless* systems, as well as the repositioning approach employed, comprising *vehicle-based* and *user-based* approaches. Orthogonally, solutions may be divided into *prediction-based methods*, which

rely on demand prediction to inform subsequent actions, and *adaptive* solutions that leverage deep reinforcement learning agents. In the remainder of this section, we aim to provide an overview of the latest related works in the field, exploring the diverse range of dynamic solutions and highlighting their strengths and limitations.

Chen et al. (2021) [17] proposed a multi-objective and multi-agent reinforcement learning solution for dynamic dispatching of bikes in dockless bicycle sharing systems. They utilized a Gated Graph Neural Network model to predict the layout of bike stations and dispatching demand. The predictions are fed to a reinforcement learning agent placed on each truck that jointly optimizes for dispatching cost, the initial load of the truck, workload balance among the trucks, and the balance between the supply and demand of bikes. Moreover, the authors of [18] proposed an incentive-based rebalancing solution for dockless BSS. They utilized spatial and temporal features to develop a hierarchical deep reinforcement learning agent; their agent outputs monetary rewards to incentivize users to pick up bikes from congested areas and travel to non-congested parts of the city. Instead, the hybrid solution formulated in [19] employs truck fleets for dynamic repositioning as well as user incentives. Their approach incorporates a spatio-temporal clustering method to extract bike demand hotspots, followed by a deep neural network called BikeNet to forecast bike demand trends. Finally, the authors designed a reinforcement learning method to divide rebalancing tasks among users and the operating fleet of trucks in order to reduce the number of unbalanced docking stations.

Wang et al. (2018) [20] and Chen et al. (2016) [21] developed solutions for predicting the next levels of demand, subsequently applying heuristic approaches to rebalance the system. More specifically, [20] implemented a data-driven approach to predict a *safe* rebalancing range for each station, which is then utilized to rebalance bikes among full and empty docking stations while minimizing the rebalancing cost. [21] instead proposed a dynamic cluster-based framework for over-demand prediction in order to compute a weighted correlation among docking stations and to group those with similar demand patterns. Subsequently, they utilized their model to estimate the number of rented and returned bikes in each cluster; finally, Monte Carlo simulation was adopted to predict the probability that cluster demand will exceed available resources.

Our solution aims at rebalancing bikes in the context of dynamic repositioning for dock-based bike sharing systems. Differently from [20], [21], our approach employs a reinforcement learning agent in order to learn the long-term dynamics of the demand for bikes. In addition, CABRA takes into account and optimizes the time (i.e., the dispatching cost) required for the repositioning of each truck. To the best of our knowledge, this is the first attempt to address dynamic bike repositioning for dock-based bike sharing systems with such joint optimization of repositioning cost, which is also evaluated on a large scale.

In fact, Li et al. (2018) [22] proposed a similar approach to ours, in which they first created clusters of docking stations, and then used a deep reinforcement learning agent with pruning rules for finding a repositioning policy that minimizes customer loss over the long term. However, their method lacks the concept of repositioning cost, which is an important concern for operators. Furthermore, this model does not account for repositioning time, unrealistically assuming that they happen instantly. Instead, our work captures the fact that trucks that are performing repositioning operations will be unavailable while carrying out these movements. Finally, their evaluation is performed on a substantially smaller scale, using data for only one city (New York) and approximately 400 stations placed in the city center.

III. Background & Problem Definition

In this section, we provide the reader with the necessary background and discuss the fundamental reinforcement learning concepts underlying CABRA. Then, we propose our mathematical formulation of the dynamic bike repositioning problem, which we aim to solve in the successive sections.

A. RL Background

Reinforcement Learning (RL) is a decision-making framework in which problems are represented mathematically as Markov Decision Processes (MDP) [23]. Typically, an agent interacts with the environment (in our case the bike-sharing system) at discrete time steps $t = 0, 1, 2, ..., T$. At every time step t, the environment provides the agent with a representation of its current status (termed *state*) $S_t \in \mathcal{S}$, based on which the agent selects an action $A_t \in \mathcal{A}(S_t)$. $\mathcal{S}$ represents the set of possible states, while $\mathcal{A}(S_t)$ represents the set of possible actions available in state S_t. Subsequently, as a result of the chosen action, the agent receives a *reward* (R_{t+1}) and transitions to a new state (S_{t+1}).

The primary objective of the agent is to determine a *policy* $\pi(A_t|S_t)$, representing a probability distribution over actions for a given state, with the goal of maximizing the *return* (discounted sum of rewards):

$$G_t = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{1}$$

where R_{t+k+1} is the reward at time $t + k + 1$ and γ is a discount factor with $0 \leq \gamma \leq 1$ that weighs the impact of immediate rewards and that of future ones differently. Furthermore, given policy π, we introduce the *state-value function*, $V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$, which represents the expected return when following the policy π from a specific state s onwards. We also define the *action-value function* as $Q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$, i.e., the expected return when taking action a in state s then following the policy π.

In deep reinforcement learning, one approach for learning the policy π is by parameterizing it using a deep neural network with parameters θ and optimizing it through the *policy gradient theorem* [24], which provides a gradient-based approach to maximize the expected return. The policy gradient theorem states that the gradient of the expected return with respect to the policy parameters can be expressed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_a Q^{\pi_\theta}(s, a)\nabla_\theta \pi_\theta(a|s)\right] \tag{2}$$

where $\nabla_\theta \pi_\theta(a|s)$ denotes the partial derivatives of the policy with respect to each of the parameters in θ.

B. Decision-Making Algorithm

In CABRA, we employed the Proximal Policy Optimization (PPO) [25] method to effectively learn a policy. PPO is a state-of-the-art policy optimization method that strikes a balance between stable policy updates and policy performance improvement. It achieves this by constraining the update step to be within a trusted region to prevent drastic policy changes. At its core, PPO leverages the policy gradient theorem. The objective of PPO is to maximize the clipped surrogate objective, which can be defined as:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}^{\text{PPO}}\hat{A}_t\right)\right] \tag{3}$$

where $\text{clip}^{\text{PPO}} = \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)$, while $r_t(\theta)$ represents the probability ratio between the updated policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$. The advantage estimate $\hat{A}_t$ measures the relative value of each action taken at time step t. The clipping term within the surrogate objective ensures that the policy update does not deviate excessively from the previous policy, limiting the potential instability. By optimizing this objective using stochastic gradient ascent, PPO has achieved significant performance gains in a wide range of reinforcement learning tasks such as games [25], [26] and multi-agent settings [27].

C. Problem Formulation

Formally, we are given a set of docking stations (also referred to as nodes) $\mathcal{D} = \{n_1, ..., n_K\}$ of size K, a set of trucks $\mathcal{W} = \{w_1, ..., w_M\}$ of size M, with $K \gg M$, and the system is equipped with H total bikes that may be hired. Each node n_i is described by its position $p_{n_i} = \langle x_{n_i}, y_{n_i}\rangle$ in a spherical coordinate system (latitude and longitude), and by two positive integer scalars: the capacity c_{n_i} and the current available bikes $b^t_{n_i}$, with $0 \leq b^t_{n_i} \leq c_{n_i}$. Each truck w_j is also described by its position $p^t_{w_j} = \langle x^t_{w_j}, y^t_{w_j}\rangle$ at time t and two positive integer scalars: the capacity κ_{w_j} and the current load $l^t_{w_j}$ (the number of bikes in the truck), with $0 \leq l^t_{w_j} \leq \kappa_{w_j}$. Finally, we denote the total number of bikes that are currently loaned out by users at time t as λ^t.

Every time step t, each node observes the quantity $o^t_{n_i}$ indicating the difference in the number of users willing to start and those wishing to end a trip, corresponding to the user demand. At each node, a number of *shortage* events $e_{n_i} \in \mathbb{N}$ may occur if one of the following conditions is met:

$$e^t_{n_i} = \begin{cases} o^t_{n_i} - b^t_{n_i} & \text{if } b^t_{n_i} < o^t_{n_i} \text{ and } o^t_{n_i} > 0, \\ |o^t_{n_i}| - (c_{n_i} - b^t_{n_i}) & \text{if } c_{n_i} - b^t_{n_i} < |o^t_{n_i}| \text{ and } o_{n_i} < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $o^t_{n_i} > 0$ if in the time interval the number of started trips is greater than the number of ended trips, and conversely if $o^t_{n_i} < 0$. In other words, this clause corresponds to the shortage event that occurs when the number of trips that are ending at a given time step is larger than the number of available slots (i.e., some users cannot drop off the bike at the desired station).

During a time step, idle trucks are allowed to move among the nodes and perform *repositioning plans*. A repositioning plan $m^t_{w_j}$ for truck w_j is defined by the tuple $\langle n^t_{\text{target}}, q^t_{w_j}\rangle$, with n^t_{target} representing the target node where the truck will move, and $q^t_{w_j}$ the quantity to be repositioned by the truck. The quantity is either negative if the truck needs to *drop* $q^t_{w_j}$ bikes or positive if the truck needs to *collect* them. The feasibility of the repositioning is subject to constraints: for a *collect* repositioning, the station must have sufficient bikes currently docked and the truck must have the corresponding spare capacity available; while, for a *drop* repositioning, the station must not exceed capacity if the bikes currently in the truck are unloaded.

Every time a truck performs a repositioning plan $m^t_{w_j}$, it will be busy for $\tau^t_{w_j}$ number of time steps, also referred to as *repositioning time*, which is given by:

$$\tau^t_{w_j} = \frac{d(p^t_{w_j}, p^t_{n_{\text{target}}})}{v_{\text{move}}} + q^t_{w_j}\Delta_{\text{loading}} \tag{5}$$

where $d(p^t_{w_j}, p^t_{n_{\text{target}}})$ is the spherical distance between the position of the truck and the target docking station, while $v_{\text{move}} \sim \mathcal{N}(\mu_{\text{move}}, \sigma_{\text{move}})$ is the speed of movement of the truck and $\Delta_{\text{loading}} \sim \mathcal{N}(\mu_{\text{loading}}, \sigma_{\text{loading}})$ is the time duration required for loading or unloading one bike to and from the truck. Therefore, given a time horizon T, the objective is:

$$\min \sum_{t=0}^{T}\left(\sum_{i=1}^{K} e^t_{n_i} + \sum_{j=1}^{M}\tau^t_{w_j}\right) \tag{6}$$

$$\text{s.t.} \quad 0 \leq b^t_{n_i} \leq c_{n_i} \qquad \forall i \in \mathcal{D}, \tag{7}$$

$$0 \leq l^t_{w_j} \leq \kappa_{w_j} \qquad \forall j \in \mathcal{W}, \tag{8}$$

$$\sum_{i=1}^{K} b^t_{n_i} + \sum_{j=1}^{M} l^t_{w_j} + \lambda^t = H \quad \forall t \in [0, T]. \tag{9}$$

Intuitively, over a time horizon of T steps, we need to minimize the number of shortages observed on all the nodes (first term of the sum) and the repositioning time, i.e., the cost to apply the repositioning (second term of the sum), while ensuring that the number of bikes at each node and the load of each truck stay within their respective capacities. Finally, the last constraint ensures that the total number of bikes in the system (that are either available at stations, being repositioned by the trucks, or currently loaned out to users) stays constant. In order to simplify the presentation, without lack of generality, we henceforth suppose that all trucks have the same capacity κ and set the time step size to 10 minutes.

IV. Methodology

This section covers the key design aspects of the CABRA approach. Subsequently, we will provide the mathematical formulation of the MDP that is used by our agent to learn repositioning strategies for dock-based BSS.

A. Proposed Approach

The settings in which bike-sharing systems operate are typically large-scale. Usually, up to thousands of docking stations are deployed in an urban area. Therefore, it is challenging to approach the problem of repositioning bikes for the complete set of docking stations at once. Moreover, a repositioning must also consider the distance to the target node to avoid long *repositioning times*, and thus high costs for the operators.

Our method involves deploying trucks across the city. At each operational cycle, idle trucks participate in a structured repositioning process, as depicted in Figure 1. This process consists of four sequential steps: first, a truck signals the need for a repositioning plan, actively engaging with the centralized control system. Second, the control agent, armed with real-time data, analyzes the current landscape – the status of docking stations and the positioning of other trucks. Third, using the gathered system information, the agent formulates a repositioning plan, specifying the target node and bike quantities to collect or drop. Finally, the truck commences the repositioning task. It is worth pointing out that, depending on the repositioning time, there might occur time steps in which all trucks are busy and hence no actions are performed. In the following subsection, we will discuss how the truck status is reflected in the definition of the Markov Decision Process, in particular in terms of state space.[1]

B. RL Settings

We now introduce the formalization of the repositioning problem within CABRA as an MDP. This includes detailed descriptions of the action and state spaces, the reward structure, and the proposed pruning rules for effectively reducing the size of the action space.

State Space. The state space in CABRA encompasses comprehensive system-level information and granular details about individual docking stations. At each timestep, it aggregates data regarding the fleet of trucks $\mathcal{W}$ (specifically, their load and position) and the availability of bikes at all docking stations $\mathcal{D}$. The state S_t is characterized by the following features:

- *Status of nodes*: is a one-dimensional vector concatenating two elements for each node - bike availability $b^t_{n_i}$ and position p_{n_i}.
- *Current truck*: is the one-hot encoded vector representing the truck w_j whose repositioning is being decided.
- *Status of trucks*: is a $4M$-dimensional vector, where each truck w_j is represented by three attributes: current load $l^t_{w_j}$, position $p^t_{w_j}$, and a *busy* flag that is set to 1 if the truck is occupied in repositioning activities during the current timestep, and 0 otherwise.
- *Current time*: is the one-hot encoding of the current hour, day of the week, and month. This feature captures temporal patterns in demand, reflecting seasonal variations.

All state features are normalized to fall within $[0, 1]$.

Action Space. The agent, a logically centralized entity with a complete overview of the system, selects repositioning parameters $\langle n^t_{\text{target}}, q^t_{w_j} \rangle$, designating the target node and the quantity to be picked up or dropped off. Thus, the action space $\mathcal{A}(S_t)$ is split into two components: the first comprises all nodes, amounting to K possible choices in total; the second represents a range of integers from $[-\kappa, \kappa]$, where negative values denote drop actions and positive values indicate pick actions, and zero represents no action. In the output layer, our policy architecture reserves K units for deciding the target node and $2\kappa + 1$ units for determining the bike quantity, constructing a separate discrete probability distribution for each of the two components. Thus, a single policy parameterization is used for determining both action components, which is advantageous in terms of stability and inference speed compared to separate parameterizations, the alternative design that appears in some of the other RL works.

Reward Function. The reward function distinguishes between beneficial and detrimental actions. Its optimization targets the quantity in Equation 6 while adhering to constraints 7 and 8. Recall the notion of a *shortage* event introduced in Section III, which occurs when users cannot pick up or drop off their bike at a given station. CABRA differentiates between two types of shortages. The first, which we call *agent-caused shortages*, arise as a result of the allocation strategy of the agent being suboptimal. They are attributable to the agent (i.e., if it moves bikes away from a station where a demand surge occurs at the next step, or, conversely, fills all slots of a node where bikes are about to be dropped off). The second type of shortages, *environment deficits*, occur without them being attributable to the explicit intervention of the agent. Distinguishing between the two types of shortages helps isolate a signal for the agent regarding the influence of the decisions it made and reduce the noise in the feedback.

After each action, the system progresses to the subsequent timestep, evaluates new demands, and records the occurrences of *agent-caused shortages* $e^t_{n_i}$ on every docking station, *environment deficits* e^t_{env}, and *repositioning time* $\tau^t_{w_j}$. The aggregate reward for actions related to truck w_j is computed as:

$$R_t(w_j) = \omega_{\text{shortages}} \sum_{i=1}^{K} e^t_{n_i} + \omega_{\text{env}} e^t_{\text{env}} + \omega_{\text{repositioning}} \tau^t_{w_j} \quad (10)$$

where $\omega_{\text{shortages}}$, ω_{env}, and $\omega_{\text{repositioning}}$ are three scalars that are used to weigh the three reward components. We note that providing individual rewards for each truck, as opposed to aggregating them for the entire system, better indicates the effectiveness of each action and alleviates the credit assignment problem. Furthermore, we note that the cost $\tau^t_{w_j}$ is also normalized to be in $[0, 1]$.

Pruning Rules. BSS operate on a vast scale, presenting significant scalability challenges when applying machine learning techniques. To mitigate these challenges, we have implemented specific pruning rules aimed at reducing the action space, thereby simplifying the problem. While pruning the action space may in principle exclude the consideration of optimal policies, our empirical findings underscore the effectiveness of this strategy, particularly in handling large-scale instances of the problem. Specifically, we introduce a capacity threshold, δ, to define nodes as being in a *critical* state for a given time step t if:

[1]CABRA's implementation available under MIT License at https://github.com/AlessandroStaffolani/cabra-paper.

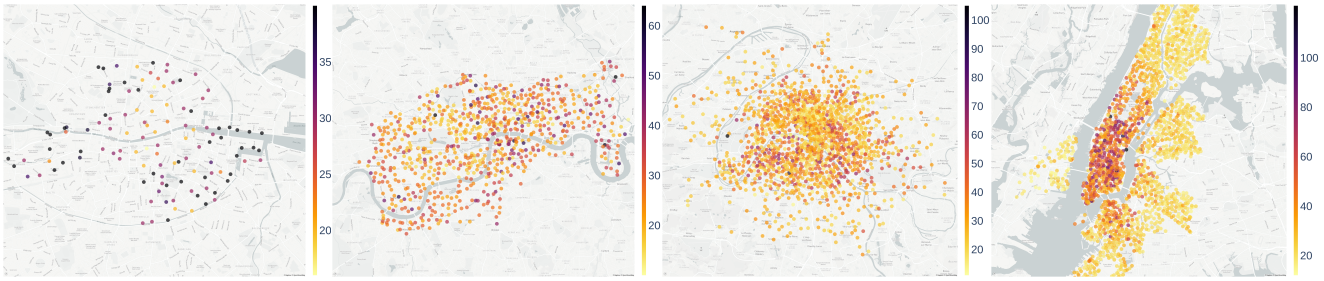

Fig. 2. Distribution of docking stations across the four cities, where varying color intensities reflect the capacity of each station.

$$n^t_{\text{critical}} = \begin{cases} 1, \text{ if } b^t_{n_i} < \delta \text{ or } c_{n_i} - b^t_{n_i} < \delta, \\ 0, \text{ otherwise.} \end{cases} \quad (11)$$

Intuitively, a node enters a critical state when the number of available bikes or empty slots dips below this threshold, indicating a high risk of imminent shortages. Our pruning approach focuses on retaining only those nodes in a critical state as potential targets for action. In cases where no nodes meet this critical state criterion, the system compels the agent to select the wait action. Notably, the computational complexity of this pruning rule is linear in the number of nodes K, and therefore it is inexpensive to compute. This is in contrast with the more expensive pruning rule used in [22], which relies on a neural-network based prediction of future demand that is applied each time a decision is taken. Additionally, we incorporate a policy that mandates alternating pick and drop actions within the same truck. This alternating action pattern is designed to optimize bike redistribution, allowing trucks first to collect bikes from nodes with excess supply and subsequently deliver them to nodes experiencing a deficit. This method enhances the operational efficiency of the redistribution process.

V. Experimental Setup

A. Datasets

We conduct our evaluation of CABRA on extensive datasets gathered from four major BSS in key European and American cities: Dublin, London, Paris, and New York. These datasets include detailed information about the locations and capacities of docking stations, as well as continuous observations of the number of bicycles and available spaces at each station within these urban bike-sharing systems. These observations are collected at intervals of approximately 10 minutes, with our evaluation focusing on the large set of real-world data from September 2021 to October 2022.

We filter out docking stations with capacities of less than three bikes, as these are identified as test stations in the raw data. Then, we have constructed the demand data by aggregating the number of trips starting and ending at each docking station. In addition, we calculate the percentage of bikes in relation to the total number of docking stations in each BSS: using this ratio, we set the initial number of bikes for each station proportionally to its capacity and the calculated percentage. Figure 2 visually illustrates the distribution and capacities of docking stations in the four cities under evaluation. Additionally, Table I presents high-level summary statistics concerning these BSS.

TABLE I
Dataset summary for each of the evaluated cities.

City	Docking Stations	Avg. Node Capacity	Bikes Percentage	Avg. Initial Bikes
Dublin	117	31.91	38.01%	12.01
London	799	26.48	43.57%	11.52
Paris	1453	31.6	25.93%	8.17
New York	1765	31.89	44.34%	14.12

TABLE II
Summary of training splits and evaluation procedure.

Data Split	11 months training, 1 validation, 1 test (disjoint temporal split).
Model Selection	Every 10 training iterations, assess reward on validation set.
Evaluation	Report reward (and its sub-objectives) on test set.
Variability	Training, validation, testing repeated across 10 random seeds.

B. Training and Evaluation Procedure

We now discuss the experimental procedure that was followed, which is summarized in Table II. In our approach, we temporally divide the datasets from the four cities into *training*, *validation*, and *evaluation* sets. The *training set* encompasses the first 11 months of data, while the *validation* and *evaluation* sets both contain 1 month of data each. During the training phase, our agent actively cycles through the data in the *training set*, generating a series of rollouts. After each rollout, we immediately execute a learning step, marking the completion of a training iteration.

To periodically check the performance of our model, we conduct a validation run on the *validation set* after every ten training iterations. If the model achieves a new best score during these validations, we save its configuration. Lastly, the best model is run on the *evaluation set* in order to assess its final performance on an unseen subset. To ensure the robustness and statistical validity of our results, we perform 10 distinct runs. Each run starts with a different random initialization of the agent's initial state. This strategy not only tests the effectiveness of our model under varied conditions, but also strengthens the reliability of our findings.

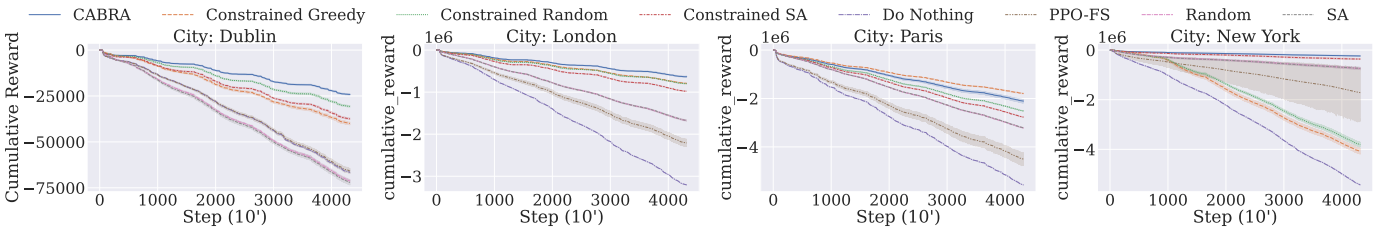

Fig. 3. Cumulative reward over the evaluation set for the four cities (the higher, the better).

TABLE III
CABRA AGENT HYPERPARAMETERS (TOP) AND ENVIRONMENT PARAMETERS (BOTTOM). *Fixed* MEANS THAT THE PARAMETER VALUE IS UNIFORM ACROSS ALL CITIES, WHILE THE OTHER PARAMETERS ARE SPECIFIC TO EACH CITY.

Parameter	*Fixed*	Dublin	London	Paris	New York
Global gradient clipping	$[-0.5, 0.5]$	-	-	-	-
Policy clip range	$[-0.2, 0.2]$	-	-	-	-
Discount factor	0.9	-	-	-	-
Entropy coefficient	0.05	-	-	-	-
Train epochs	10	-	-	-	-
Shared net units	$\{512, 512\}$	-	-	-	-
Policy net units	$\{128, 64\}$	-	-	-	-
Value net units	$\{128, 64\}$	-	-	-	-
Agent learning rate	-	0.0003	0.003	0.0003	0.003
Batch size	-	512	512	512	128
Rollout size	2048	-	-	-	-
Training iterations	-	1200	1500	2000	800
System total bikes (H)	-	1822	9351	23057	21387
Truck fleet dimension (M)	-	3	20	30	40
Truck capacity (κ_{w_j})	20	-	-	-	-
Truck movement speed average (μ_{move})	$5\frac{m}{s}$	-	-	-	-
Truck movement speed standard deviation (σ_{move})	0.8	-	-	-	-
Time for loading one bike average (μ_{loading})	$60s$	-	-	-	-
Time for loading one bike standard deviation (σ_{loading})	0.5	-	-	-	-
Critical threshold relative to node's capacity (δ)	0.2	-	-	-	-

C. Agent and Environment Setup

In our PPO implementation, we opt for a shared network architecture with several fully connected layers, each activated by a hyperbolic tangent function. We then use two distinct heads: one leads to the policy network, and the other to the value network. This allows each network to specialize in its respective function while benefiting from the shared foundational processing. Furthermore, we fine-tune the hyperparameters of our agent for each of the four cities under study. Please refer to Table III for the obtained hyperparameter values.

We configure a range of truck parameters such as fleet size, movement speed, repositioning speed, and capacity. For the specific values we use in these configurations, please refer to Table III. During our experiments across the four cities, we maintain consistency in all environment settings except for the fleet size, which is set proportionally to the number of nodes to contain a truck per approximately 40 stations.

It is worth discussing how this parameter might impact performance. For very low values on the spectrum of possible fleet sizes, we expect that all repositioning methods would perform very poorly, as the fluctuations in demand would simply be too high for an agent to "catch up" through its reallocations. On the other hand, high values would make the problem very easy to solve, as we would have sufficient redundancy to reallocate capacity even with a simple strategy. The results we present in the next section show that the differences between the reallocation strategies are significant, positioning them away from either extreme. This suggests that an effective repositioning strategy can have a practical impact. We expect that similar comparative performance would be observed for other middle ground values.

D. Baselines

In our evaluation, we assess the effectiveness of CABRA by comparing it with several baseline strategies:

- ***Random***: This approach randomly selects actions, uniformly sampling from the action space.
- ***Do Nothing***: Under this strategy, the considered BSS always opts for wait actions. This baseline highlights the inefficiencies that arise when BSS are left to self-regulate.
- ***Constrained Random***: Actions are randomly chosen from the pruned action spaces. Details about the pruning rules employed can be found in Section IV-B (RL Settings).

- ***Constrained Greedy***: This method chooses actions from the pruned spaces, prioritizing the target node that is closest to the current truck and has the most critical state. Then, it selects the highest quantity of bikes that can be picked up or dropped off, which helps prevent shortages in future time steps.
- ***Full Space PPO***: This is a reinforcement learning agent with the same state representation and policy design as proposed in CABRA, but that does not apply any pruning rules to the action spaces. We refer to this baseline as PPO-FS.
- ***Simulated Annealing (SA)***: We consider the classic SA metaheuristic [28], which perturbs the solution and uses an adaptive acceptance criterion. We also include the ***Constrained SA*** variant with a pruned action space. These variants use the same modifications (i.e., action space) and objective function (i.e., reward function) as the RL methods; hence, the techniques have the same information at decision time. The initial temperature and cooling rate parameters were tuned and set to 10^5 and 0.999 (otherwise, the SA variants would reject solutions consistently in later steps, leading to poor performance).

We would like to note that other works in this area, some of which are based on machine learning and reinforcement learning techniques, are incompatible with our problem formulation. These were discussed in Section II. Notably, they do not support a notion of repositioning cost, unrealistically assume that repositioning actions happen instantly, and do not consider the multiple conflicting objectives. Despite this, we have made an effort to ensure that our technique is compared to suitable baselines spanning heuristics, metaheuristics, and standard RL methods. This aims to meaningfully and fairly demonstrate the advantages of CABRA.

VI. Experimental Results

Figure 3 showcases the average cumulative rewards for our method compared to established baselines across the unseen evaluation sets in four cities. Figure 4 details the performance for each metric attained by CABRA and the baselines, including average cumulative agent-caused shortages (first row), environment deficits (second row), and costs (third row). Table IV provides a concise summary of these performance results, highlighting the average total scores for the reward and the three metrics. We note that CABRA only explicitly optimizes the reward function, while the three (competing) metrics can be used to further judge the obtained trade-offs.

The results consistently demonstrate that the methodology and design implemented in CABRA significantly surpass the established baselines in three out of four cities tested. This evidence underscores the efficacy of the pruning rules; notably, *PPO-FS* does not exceed the performance of the *Random* policy, except in Dublin. The smaller scale of Dublin's BSS network likely contributes to this exception. While the pruning rules effectively address shortages, they also lead to a higher average management cost. This cost arises because the rules compel the agent to act — choosing an alternative to waiting — whenever a node reaches a critical state. In contrast, other methods can opt to wait without incurring costs.

These experiments show that standard SA generally performs better than random reallocations due to its ability to reject sampled modifications if they do not improve the solution. Constrained SA performs better than SA overall, emphasizing the effectiveness of the action space reduction. Constrained SA shows particularly competitive performance for New York (where it ranks second out of all methods). However, this technique is fundamentally limited by its inability to adapt to evolving demands and its failure to account for the long-term consequences of its decisions.

Furthermore, the reported results indicate that ignoring the issue of shortages in BSS is not viable, even when considering that the *Do Nothing* approach incurs no management cost. This method consistently yields the lowest performance and suffers the highest incidence of shortages across all cities. Dublin represents a unique case where the limited number of docking stations results in comparable performance between the *Do Nothing*, *Random*, and *PPO-FS* strategies.

Dublin. In the Dublin BSS, CABRA demonstrates a notable improvement in total reward. Specifically, there is an increase of at least 27% over methods that utilize the same pruning rules, i.e., *Constrained Greedy* and *Constrained Random*. This enhancement is even more pronounced, exceeding 273% compared to the other methods. Moreover, CABRA consistently excels in individual optimized metrics, outperforming other methods that apply pruning rules. This showcases CABRA's superior ability to accurately predict bike demand, reduce shortages, and optimize operational costs.

London. Similarly to the findings in Dublin, the BSS results in London are significantly impacted by CABRA, demonstrating comparable performance improvements despite the increase in instance size from 117 to 799 docking stations. Our method shows a notable improvement in total reward compared to two constrained baselines, with increases of approximately 24% and 26%. Moreover, when compared to other baselines, the improvement is at least 264%. This experiment also reveals that the performance on individually optimized metrics consistently surpasses the constrained baselines. We attribute these superior scores to the proactive approach of our model, which includes learning bike usage patterns and preemptively addressing shortages.

Paris. The BSS in Paris presents a significant challenge for CABRA, as our method was unable to surpass the performance of the *Constrained Greedy* baseline. This lower performance might be partially attributed to the BSS larger size, encompassing 1453 docking stations, and the widespread distribution of nodes throughout the city. It is interesting to note that a comparative analysis of node distribution between Paris and New York (as illustrated in Figures 2c and 2d) reveals a key difference: in New York, nodes with higher capacities, indicating areas of more intensive bike usage, are concentrated in a smaller area, specifically on Manhattan Island; conversely, these high-capacity nodes are scattered across the city of Paris. Therefore, in order to better understand the behavior of the different baselines in cities with different topologies and traffic patterns, we analyze the regularity and unpredictability of fluctuations across the four datasets using *approximate entropy* (ApEn) [29]. The results, presented in

TABLE IV
AVERAGE TOTAL REWARDS, AGENT-CAUSED SHORTAGES, ENVIRONMENT DEFICITS, AND COSTS FOR CABRA AND BASELINE MODELS ACROSS FOUR CITIES OVER THE EVALUATION SET.

		Reward (↑)	Agent-caused Shortages (↓)	Environment Deficits (↓)	Cost (↓)
Dublin	CABRA	**−24 149.57 ± 341.24**	**24 719.9 ± 569.06**	10 923.9 ± 268.62	1 731.45 ± 25.94
	Constrained Greedy	−39 938.43 ± 681.47	36 250.6 ± 920.91	20 407.5 ± 269.54	2 811.25 ± 2.64
	Constrained Random	−30 681.55 ± 400.63	27 892.0 ± 430.92	15 731.7 ± 262.41	2 007.7 ± 8.48
	Constrained SA	−37 452.47 ± 428.95	37 018.4 ± 582.1	17 894.8 ± 259.65	2 096.93 ± 7.5
	Do Nothing	−66 697.5 ± 0.0	127 281.0 ± 0.0	**3 057.0 ± 0.0**	**0.0 ± 0.0**
	PPO-FS	−65 956.18 ± 1 321.19	122 378.6 ± 2 931.95	4 508.4 ± 689.9	516.95 ± 226.33
	Random	−71 312.45 ± 747.51	92 163.0 ± 1 164.57	24 069.2 ± 262.81	2 323.49 ± 6.78
	SA	−72 105.56 ± 1 309.58	93 255.3 ± 1 663.99	24 315.6 ± 548.13	2 324.61 ± 4.32
London	CABRA	**−634 729.21 ± 12 461.17**	**859 041.0 ± 23 159.27**	201 584.2 ± 6 582.05	7 249.01 ± 162.2
	Constrained Greedy	−790 073.32 ± 5 633.92	929 982.8 ± 7 525.9	320 369.7 ± 2 878.81	9 424.43 ± 31.09
	Constrained Random	−799 456.32 ± 2 881.23	970 955.8 ± 4 817.57	309 958.5 ± 1 456.96	8 039.84 ± 18.52
	Constrained SA	−979 540.73 ± 5 972.72	1 248 817.9 ± 6 854.88	351 051.5 ± 3 085.65	8 160.56 ± 13.97
	Do Nothing	−3 206 340.0 ± 0.0	6 189 640.0 ± 0.0	111 520.0 ± 0.0	**0.0 ± 0.0**
	PPO-FS	−2 208 448.18 ± 84 507.12	4 210 729.3 ± 169 899.67	**99 836.9 ± 3 249.08**	6 493.25 ± 317.01
	Random	−1 676 027.22 ± 11 852.75	2 606 066.2 ± 20 318.04	368 513.9 ± 2 761.33	8 960.44 ± 6.88
	SA	−1 673 464.09 ± 14 150.09	2 599 082.7 ± 26 592.32	369 445.0 ± 2 422.26	8 955.47 ± 7.34
Paris	CABRA	−2 106 609.27 ± 71 103.59	2 728 647.9 ± 81 016.71	737 581.1 ± 41 812.78	9 408.44 ± 88.7
	Constrained Greedy	**−1 794 667.56 ± 14 538.86**	**1 771 455.8 ± 15 600.91**	904 061.4 ± 9 831.56	9 756.51 ± 3.81
	Constrained Random	−2 514 256.04 ± 8 017.3	2 883 116.5 ± 10 085.19	1 068 014.2 ± 5 328.35	9 367.17 ± 3.36
	Constrained SA	−2 768 849.24 ± 8 745.22	3 349 873.2 ± 12 604.71	1 089 346.7 ± 6 363.87	9 131.88 ± 7.46
	Do Nothing	−5 569 530.0 ± 0.0	9 956 760.0 ± 0.0	591 150.0 ± 0.0	**0.0 ± 0.0**
	PPO-FS	−4 501 324.94 ± 277 077.88	7 961 912.17 ± 513 342.68	**518 438.5 ± 24 829.88**	3 860.72 ± 1 250.67
	Random	−3 218 415.59 ± 10 552.67	4 573 689.0 ± 21 512.72	926 946.1 ± 4 830.33	9 249.98 ± 5.2
	SA	−3 210 605.57 ± 12 105.36	4 561 506.4 ± 20 279.55	925 215.5 ± 3 287.27	9 253.73 ± 4.61
New York	CABRA	**−237 517.84 ± 16 158.82**	**366 400.9 ± 20 412.51**	**54 285.1 ± 6 871.84**	64.57 ± 1.72
	Constrained Greedy	−4 080 212.14 ± 112 474.32	6 191 078.1 ± 323 838.36	984 667.9 ± 63 961.54	10.38 ± 1.05
	Constrained Random	−3 810 758.45 ± 93 608.78	6 276 634.2 ± 250 494.62	672 437.5 ± 38 569.69	7.7 ± 0.57
	Constrained SA	−362 631.96 ± 20 394.61	505 970.5 ± 23 220.15	109 605.8 ± 8 934.8	61.81 ± 0.99
	Do Nothing	−5 432 880.0 ± 0.0	9 943 840.0 ± 0.0	460 960.0 ± 0.0	**0.0 ± 0.0**
	PPO-FS	−1 715 128.39 ± 1 091 528.35	3 032 675.9 ± 2 012 661.3	198 768.1 ± 87 744.52	44.69 ± 13.89
	Random	−740 104.16 ± 40 601.5	1 228 146.2 ± 68 078.08	126 001.7 ± 6 676.31	58.73 ± 0.95
	SA	−735 898.21 ± 41 161.16	1 214 591.0 ± 67 009.02	128 572.9 ± 8 014.0	59.62 ± 0.93

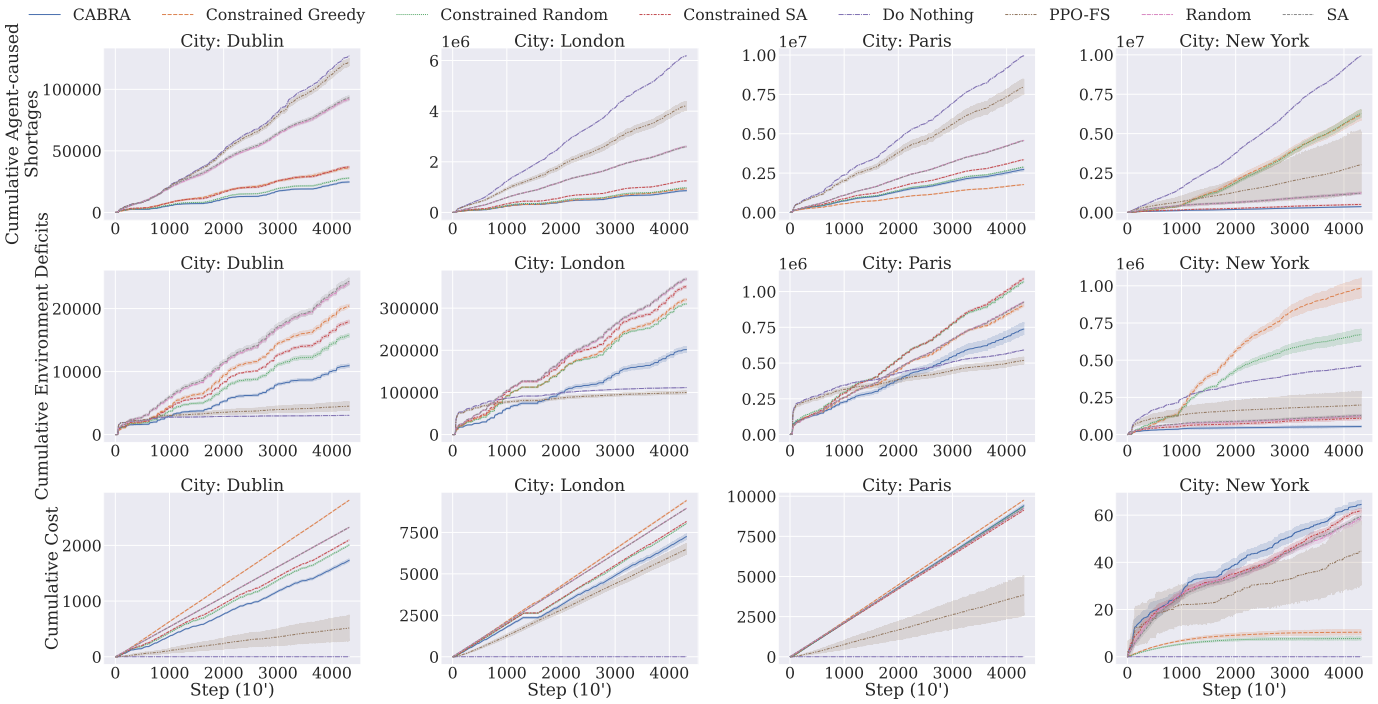

Fig. 4. Cumulative agent-caused shortages, environment deficits, and cost over the evaluation set for the four cities (the lower, the better).

TABLE V
AVERAGE APPROXIMATE ENTROPY OF THE EVALUATION SET FOR THE FOUR CITIES.

City	Avg. Approximate Entropy (ApEn)
Dublin	0.509 ± 0.036
London	0.518 ± 0.013
New York	0.535 ± 0.013
Paris	0.657 ± 0.014

Table V, show the average approximate entropy for each city, calculated by determining the ApEn for the time series of the demand at each node, and then averaging these values across the entire city. Notably, Paris exhibits a considerably higher approximate entropy compared to the other cities, indicating more unpredictable data. This higher level of unpredictability hinders the learning of efficient demand patterns, limiting the effectiveness of proactive strategies and favoring reactive greedy optimization approaches. We believe that this is the most relevant reason why the *Constrained Greedy* baseline, which prioritizes the most unbalanced nodes, emerges as the most effective method in this context. Nevertheless, CABRA still ranks second in terms of performance even in the Paris scenario, which is challenging for BSS optimization methods.
New York. In this BSS, CABRA achieves a 42% gain in total reward compared to the second-best method (Constrained SA) and at least 311% over the other baselines. This result is attained despite New York having the largest number of nodes (1765). A key contributing factor is likely the concentration of high-capacity docking stations coupled with the system's low approximate entropy. Interestingly, in this experiment, the cost incurred by our solution is the highest, yet it effectively minimizes shortages. However, this cost is relatively low compared to the costs in other cities. This efficiency can be attributed to the strategic concentration of docking stations, which enables our agent to learn a few highly effective repositioning strategies. These strategies optimize bike distribution throughout the evaluation period, demonstrating the effectiveness of our approach in a densely populated urban setting.

VII. CONCLUDING REMARKS

In this study, we introduced CABRA, an innovative, cost-aware, and adaptive bike repositioning agent employing deep reinforcement learning to dynamically optimize dock-based BSS. CABRA has demonstrated its ability in learning bike demand patterns and enabling proactive repositioning strategies that effectively reduce shortage events and optimize operational costs for BSS operators. The efficacy of CABRA was rigorously validated using real-world datasets from major providers in Dublin, London, Paris, and New York. The results consistently showed that CABRA not only outperforms traditional greedy approaches, but can also take repositioning costs into account while scaling to large BSS.

This research not only shows the efficacy of reinforcement learning as a powerful tool to enhance BSS performance, but also highlights the important role of pruning rules in managing large-scale operations in these contexts. Moreover, the study sheds light on the influence of randomness in demand patterns when utilizing machine learning, offering valuable insights for system operators and urban planners in optimizing docking station deployment to support adaptive strategies.

In large cities, a possible deployment strategy for CABRA is to divide the city into different regions and perform the optimization on each region separately. This hierarchical splitting provides a possibility for local authorities to operate the bike sharing system for the zone(s) that they are responsible for. Preliminary experiments with such a hierarchical design revealed worse performance compared to a centralized model, which has greater flexibility in resource reallocation. However, such an approach may be necessary for practical and administrative reasons in real deployments.

Looking ahead, we aim to widen the applicability of our findings across diverse bike-sharing models, and extend the CABRA approach for compatibility with dockless BSS. We also plan to implement and evaluate our approach in a live BSS, providing a practical testbed for our algorithms. It is worth noting that CABRA introduces minimal overhead, enabling real-time decision-making in practical BSS scenarios. Notably, even in the largest instance, the complete action selection process - including state creation, application of pruning rules, and forward pass in the neural network - averages under 9 milliseconds, which is suitable for real-time applications. Lastly, we aim to develop a decentralized model based on multi-agent settings [30], and to offer a comparative analysis of both methodologies.

ACKNOWLEDGMENTS

The data for this research were provided by the Consumer Data Research Centre, an ESRC Data Investment, under project identifiers CDRC 1316, ES/L011840/1, ES/L011891/1. We thank Oliver O'Brien for his assistance with accessing the data. We acknowledge a CINECA award under the ISCRA initiative, which provided HPC resources and support.

REFERENCES

[1] J. Todd, O. O'Brien, and J. Cheshire, "A global comparison of bicycle sharing systems," *Journal of Transport Geography*, vol. 94, 2021.
[2] E. Fishman, S. Washington, and N. Haworth, "Bike Share: A Synthesis of the Literature," *Transport Reviews*, vol. 33, no. 2, pp. 148–165, 2013.
[3] O. O'Brien, J. Cheshire, and M. Batty, "Mining bicycle sharing data for generating insights into sustainable transport systems," *Journal of Transport Geography*, vol. 34, pp. 262–273, 2014.
[4] Y. Zhang, D. Lin, and Z. Mi, "Electric fence planning for dockless bike-sharing services," *Journal of Cleaner Production*, vol. 206, pp. 383–393, 2019.
[5] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.
[6] M. Dell'Amico, M. Iori, S. Novellani, and A. Subramanian, "The bike sharing rebalancing problem with stochastic demands," *Transportation Research Part B: Methodological*, vol. 118, pp. 362–380, 2018.
[7] B. P. Bruck, F. Cruz, M. Iori, and A. Subramanian, "The static bike sharing rebalancing problem with forbidden temporary operations," *Transportation Science*, vol. 53, no. 3, pp. 882–896, 2019.
[8] M. Batty, "Artificial intelligence and smart cities," *Environment and Planning B: Urban Analytics and City Science*, vol. 45, no. 1, pp. 3–6, 2018.
[9] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2020.

[10] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.
[11] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.
[12] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proceedings of the 5th International Conference on Learning Representations (ICLR'17) Workshop Track*, 2017.
[13] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
[14] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proceedings of the 7th International Conference on Learning Representations (ICLR'19)*, 2019.
[15] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.
[16] J. Schuijbroek, R. Hampshire, and W.-J. van Hoeve, "Inventory rebalancing and vehicle routing in bike sharing systems," *European Journal of Operational Research*, vol. 257, no. 3, pp. 992–1004, 2017.
[17] J. Chen, K. Li, K. Li, P. S. Yu, and Z. Zeng, "Dynamic Bicycle Dispatching of Dockless Public Bicycle-Sharing Systems Using Multi-Objective Reinforcement Learning," *ACM Transactions on Cyber-Physical Systems*, vol. 5, no. 4, 2021.
[18] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, "A Deep Reinforcement Learning Framework for Rebalancing Dockless Bike Sharing Systems," *Proceedings of the 33rd AAAI Annual Conference on Artificial Intelligence (AAAI'19)*, vol. 33, no. 01, pp. 1393–1400, 2019.
[19] H. Zhu, T. Shou, R. Guo, Z. Jiang, Z. Wang, Z. Wang, Z. Yu, W. Zhang, C. Wang, and L. Chen, "RedPacketBike: A Graph-Based Demand Modeling and Crowd-Driven Station Rebalancing Framework for Bike Sharing Systems," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4236–4252, 2023.
[20] S. Wang, T. He, D. Zhang, Y. Shu, Y. Liu, Y. Gu, C. Liu, H. Lee, and S. H. Son, "BRAVO: Improving the Rebalancing Operation in Bike Sharing with Rebalancing Range Prediction," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, vol. 2, no. 1, 2018.
[21] L. Chen, D. Zhang, L. Wang, D. Yang, X. Ma, S. Li, Z. Wu, G. Pan, T.-M.-T. Nguyen, and J. Jakubowicz, "Dynamic Cluster-Based over-Demand Prediction in Bike Sharing Systems," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16)*, 2016, p. 841–852.
[22] Y. Li, Y. Zheng, and Q. Yang, "Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*, 2018, p. 1724–1733.
[23] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[24] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, 1999.
[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.
[26] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv:1912.06680*, 2019.
[27] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 24 611–24 624.
[28] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[29] S. M. Pincus, I. M. Gladstone, and R. A. Ehrenkranz, "A regularity statistic for medical data analysis," *Journal of Clinical Monitoring*, vol. 7, no. 4, pp. 335–345, Oct 1991.
[30] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in Multi-Agent Systems and Applications - 1*, pp. 183–221, 2010.

Alessandro Staffolani received MSc and PhD degrees in Computer Science and Engineering at University of Bologna, Italy. He is interested in learning-based approaches for addressing the optimization of resources by means of scheduling and orchestration, in the context of distributed systems and network infrastructure.

Victor-Alexandru Darvariu is a Postdoctoral Researcher at the Oxford Robotics Institute, Department of Engineering Science, University of Oxford. He received his PhD in Computer Science from University College London in 2023. His research interests span reinforcement learning and planning, combinatorial optimization, network science, graph neural networks, game theory, and multi-agent systems. He investigates applications of these techniques in areas as diverse as robotics, operations research, computer systems, and causal inference.

Paolo Bellavista received MSc and PhD degrees in computer science engineering from the University of Bologna, Italy, where he is a full professor of distributed and mobile systems. His research activities span from pervasive wireless computing to online big data processing under quality constraints, from edge cloud computing to middleware for Industry 4.0 applications. He has published around 300 papers, with around 120 of them in major international journals in the above fields. He serves on several Editorial Boards of leading IEEE and ACM journals.

Mirco Musolesi is Full Professor of Computer Science at the Department of Computer Science at University College London. He is also Full Professor of Computer Science at the Department of Computer Science and Engineering at the University of Bologna. Previously, he held research and teaching positions at Dartmouth, Cambridge, St Andrews and Birmingham. The focus of his lab is on Machine Learning/Artificial Intelligence and their applications to a variety of domains, including infrastructure optimization.