Management of Shared Mobility: Report

Eduardo Passos
MSc in Artificial Intelligence
University of Porto
Porto, Portugal
up202205630@up.pt

Guilherme Silva
MSc in Artificial Intelligence
University of Porto
Porto, Portugal
up202205298@up.pt

Valentina Cadime
MSc in Artificial Intelligence
University of Porto
Porto, Portugal
up202206262@up.pt

***Abstract*—This report presents a realistic discrete-time simulation of an urban shared micro-mobility system for e-scooter fleet management. The simulator models stochastic demand with commute, weather, and event effects, as well as operational constraints, such as station capacity, minimum state-of-charge (SoC) for rentals, charging delays, and relocation effort. We evaluate controller families across varied networks and scenario shifts: static planner baselines, a tick-level heuristic, episodic and contextual bandits, and a deep RL agent (Soft Actor-Critic, SAC).**

Across networks, SAC consistently ranks among the top performers and achieves the best or near-best objective in several configurations. However, tuned static baselines and the heuristic controller remain competitive in nominal conditions. Under scenario shift, long-horizon (168h) performance is primarily limited by queue accumulation under persistent spatial heterogeneity, while episodic bandits identify stable static operating modes but cannot adapt within an episode. Overall, the results emphasize that nominal performance alone is insufficient: robustness deltas and objective decomposition are necessary to reveal failure modes and trade-offs between service quality and operational cost.

***Index Terms*—Modelling, reinforcement learning, scooter fleet management, simulation, optimization**

I. Introduction

Managing a shared electric scooter (e-scooter) fleet in a city is an operational control problem where an operator must keep the vehicles available, despite stochastic demand, spatial imbalance, battery depletion, charging delays, and finite relocation/charging resources. In practice, commutes, weather, and localized events can quickly drain stock at high-demand stations, while others overflow. This leads to unmet demand and persistent queues, which accumulate over time. Thus, management strategies fragile to shifts may hide their limitations on "typical" conditions. As a result, identifying a resilient strategy is needed.

To this end, a clear definition of the system in which strategies will be evaluated is required. For e-scooter fleet management, the system can be defined as a discrete horizon of T time steps, with the system state $\mathcal{S}_t$ summarizing fleet counts, battery SoC, and queue backlog across stations. At each step, a control policy selects an action $\mathbf{a}_t \in [0,1]^4$ that parameterizes the relocation and charging planners, producing executable interventions that drive the transition $\mathcal{S}_t \to \mathcal{S}_{t+1}$ under stochastic demand and exogenous factors. Ultimately, the strategies aim to minimize the cost J_{run} (Eq.(5)), computed as the average of the per-step objective J_t (Eq.(4)).

Indeed, the aim of this project is to provide a realistic and reproducible simulation testbed for scooter fleet management. In particular, the strategies will be compared under uncertainty and scenario shift. Concretely, we (i) build a discrete-time simulator based on real city maps and operational constraints, (ii) implement multiple controller families (static planners, an interpretable heuristic, bandits, and deep RL), and (iii) evaluate 24h and 168h performance via robustness deltas and objective decomposition. We address the following research questions and hypotheses: (RQ1) the competitiveness of tuned static planners under nominal demand; (RQ2) controller robustness to persistent heterogeneity and event shocks; (RQ3) how long-horizon evaluation changes ranking and failure modes; and (RQ4) the interpretability–robustness trade-off. We hypothesize that (H1) SAC improves performance under sustained heterogeneity by adapting intervention intensity, (H2) queue accumulation drives long-horizon failures, (H3) contextual bandits adapt but may be high-variance on long horizons, and (H4) a well-designed heuristic can match top performance in benign regimes while being transparent.

The proposed simulator is primarily *prescriptive*, supporting decisions on relocation and charging. Moreover, it is *descriptive*, via explicit state tracking (availability, queue backlog, SoC), and also *predictive*, in the sense that demand intensity evolves as a stochastic time-dependent process, influenced by weather and events. Furthermore, it is *speculative* when testing various "what-if" stress scenarios. Lastly, it is only *normative* when considering that policies are compared using an explicit objective that weighs service quality against operational cost.

Next, we review related work. Section III details the simulator and system, and Section IV has the experimental protocol and results. Lastly, Section V makes final remarks.

II. Related work

Shared mobility systems, such as ride-sharing, car-pooling, and micromobility, rely on the efficient matching of vehicle supply with user demand. A primary challenge in these systems is the imbalance of fleets caused by asymmetric travel patterns. Early literature predominantly addressed this via static mathematical optimization, using Mixed Integer Linear Programming (MILP) or heuristics to minimize travel costs or penalties for unmet demand. However, static approaches fail to account for the stochastic nature of demand and evolution

of the system. Consequently, research evolved towards dynamic rebalancing and Simulation Optimization (SO), which integrates the simulation's ability to capture complex system dependencies and uncertainties with optimization techniques. Despite these advancements, significant gaps remain in applying SO to dockless systems due to their high-dimensional state spaces and modeling complexity. [1]

To address these limitations, Reinforcement Learning (RL) has emerged as a robust approach for optimizing shared mobility, including pricing, matching, and repositioning. Unlike greedy heuristics, RL agents optimize cumulative long-term rewards by interacting with the environment, being extremely useful in dynamic systems. Recent surveys highlight the efficacy of Deep Q-Networks (DQN) and policy gradient methods in learning complex dispatching and repositioning policies that outperform myopic strategies. Furthermore, Multi-Agent Reinforcement Learning (MARL) has been proposed to tackle the coordination challenges inherent in large-scale fleets, where individual driver agents must cooperate to achieve system-level equilibrium. However, realizing the full potential of RL requires overcoming challenges related to model complexity, agent coordination, and the sim-to-real gap. [2]

While the aforementioned methodologies mainly prioritize operational efficiency (maximizing profits or trip fulfillment), recent publications argue for the integration of social objectives, specifically equity. Current rebalancing models often yield suboptimal distributions for disadvantaged communities, exacerbating accessibility gaps. New frameworks are beginning to address this by incorporating equity objectives, such as the Atkinson inequality index, into rebalancing optimization. For instance, machine learning-based approaches have been employed to predict equity performance metrics, guiding rebalancing operations to minimize access inequality in dockless micromobility services. This shift highlights a meaningful gap in the broader literature: the need for unified frameworks that jointly optimize for diverse objectives: financial sustainability, operational efficiency, and equitable access, balancing them within highly dynamic and stochastic shared mobility environments. [3]

III. Materials and Methods

The system is formalized as an agent-based discrete-time simulation composed of the interactions between a single operator (the agent), the entities, and finite network resources. The agent acts by controlling specific parameters that influence either scooter relocation and charging. The system entities are the customers and the scooter fleet. Customers are modeled as transient entities represented by stochastic arrival events, while the fleet consists of scooter counts circulating between stations (nodes). These entities interact with stations, which function as resources within the system. Stations are characterized by attributes such as parking capacity and charger availability, effectively constraining the flow of entities by limiting parking and charging capacities.

The simulation operates within a dynamic environment (whose specific instantiation for the experiments conducted is defined in Table I), subject to external variations, including time-of-day effects, weather conditions, and special events. These environmental factors, combined with the stochastic user demand, constitute the *uncontrollable exogenous variables*, corresponding to the probabilistic part of the system, as detailed in III-C.

The instantaneous status of the system is captured by state variables, which correspond to the *endogenous variables* formally defined in III-B. The evolution of this state is driven by customer arrivals, trip completions, and operator interventions, which trigger specific activities including scooter rental, fleet relocation, and battery charging. The infrastructure parameters and resource limits that bound these activities form the *controllable exogenous variables*, explored in III-A. Additionally, the specific relocation and charging plans built by the operational policies at each time step are included in the controllable exogenous variables, as they are generated externally to the simulation dynamics, by the agent.

A. Simulation Configuration

The simulation environment is characterized by a set of controllable exogenous parameters (implemented in the `SimConfig` dataclass).

- Δt: Discrete simulation time step duration (minutes).
- H: Total simulation horizon (hours).
- $C \in \mathbb{N}^N$: Vector of station capacities, where C_i is the maximum number of scooters allowed at station i.
- $\mathcal{T} \in \mathbb{R}^{N \times N}$: Travel time matrix, where τ_{ij} represents the effective transit time between station i and j.
- $\mathcal{D} \in \mathbb{R}^{N \times N}$: Distance, where d_{ij} represents the distance (km) between station i and j.
- $K \in \mathbb{N}^N$: Vector of station chargers at each node.
- $\rho \in \mathbb{R}^N$: Charging rate vector, where ρ_i denotes the SoC gain per hour at station i.
- B: Total battery capacity per scooter (kWh).
- c_{energy}: Unit cost of energy (€/kWh).
- s_{min}: Minimum SoC threshold required for a scooter to be eligible for rental. Always set to 15%.
- $\gamma_{res} \in \{0, 1\}$: Operation policy parameter indicating if scooters connected to the grid should be reserved (unavailable for rental) during the active charging time step. For the experiments, this is always set to true.

B. Simulation State

At any time t, the system state is defined by endogenous variables tracking the fleet and demand status (implemented as attributes of the `Sim` class):

- $\mathbf{x}_t \in \mathbb{N}^N$: Vehicle count at each station.
- $\mathbf{s}_t \in [0, 1]^N$: Average SoC at each station.
- $\mathbf{m}_t \in \mathbb{R}^N$: Aggregated energy "mass" at each station. Calculated as $m_{i,t} = x_{i,t} \cdot s_{i,t}$.
- $\mathbf{q}_t \in \mathbb{N}^N$: Number of users currently in queue.
- Ψ_t: Set of active trips currently in transit, implemented as a min-heap priority queue sorted by arrival time. A trip is a tuple $(t_{arrival}, j, \delta_{soc}, s_{depart})$, denoting arrival time, destination station, SoC usage, and departure SoC.

C. Simulation Dynamics

The state transition $\mathcal{S}_t \rightarrow \mathcal{S}_{t+1}$ occurs via the following sequential steps, in which the uncontrollable exogenous variables manifest as stochastic events:

1) Demand and Departures: Demand arrivals $\mathbf{A}_t$ follow a non-homogeneous Poisson process:

$$\lambda_{i,t}^{eff} = \lambda_i \cdot f_{time}(t) \cdot f_{weather}(t) \cdot f_{event}(t) \tag{1}$$

where λ_i is the base arrival rate at station i, and $f_{\text{time}}(t)$, $f_{\text{weather}}(t)$, and $f_{\text{event}}(t)$ are non-negative multiplicative factors that model diurnal variation, weather attenuation, and localized event surges, respectively.

The number of served users is limited by the rentable stock of scooters ($s_{i,t} > s_{min}$).

User destinations are sampled from the origin-destination probability matrix $P \in \mathbb{R}^{N \times N}$, where p_{ij} represents the probability of a user departing from station i terminating their trip at station j. To simplify the simulation and focus on rebalancing dynamics rather than complex routing patterns, we assume a uniform distribution:

$$p_{ij} = \frac{1}{N}, \quad \forall j \in \{1, \ldots, N\} \tag{2}$$

When a user departs, a vehicle is removed from $\mathbf{x}_{i,t}$, and a trip event is pushed to the trip heap Ψ.

2) Trip Completion: Trips scheduled to arrive at time t are processed. If the stations if full, the system reroutes the scooter to the nearest station with available capacity, updating the used resources accordingly.

3) Relocation Actions: Following a relocation plan, an operator moves scooters between stations, updating stocks and energy mass accordingly. The relocation cost is computed as:

$$R_t = \sum_{(i,j,k) \in \text{Plan}_t} k \cdot d_{ij} \tag{3}$$

where k is the number of scooters relocated from station i to j, constrained by available stock at i and capacity at j.

4) Charging Logic: Charging is applied to idle scooters connected to plugs according to a charging plan. The number of charging scooters $k_{i,t}^{plug}$ is constrained by station stock $x_{i,t}$ and available chargers K_i.

5) Optimization Objective: The objective is to minimize the *time-averaged* normalized cost over an episode of length T time steps, where $T = 60H/\Delta t$ and H is the horizon in hours. The step-wise cost is defined as:

$$J_t = w_U \frac{U_t}{U_0} + w_R \frac{R_t}{R_0} + w_C \frac{C_t}{C_0} + w_Q \frac{Q_t}{Q_0}. \tag{4}$$

The reported episode objective is the time average:

$$J_{\text{run}} = \frac{1}{T} \sum_{t=0}^{T-1} J_t. \tag{5}$$

For completeness, the cumulative episode cost is $J_{\text{tot}} = \sum_{t=0}^{T-1} J_t = T \cdot J_{\text{run}}$.

The cost components correspond to the Key Performance Indicators (KPIs) used to evaluate system performance.

- U_t: **Unavailability rate**. Defined as $U_t = 1 - \frac{\text{served\_new}_t}{\max(1, \text{demand}_t)}$, i.e., the fraction of new arrivals that cannot be served immediately at step t. We also report **availability** as $A_t = 1 - U_t$ for interpretability.
- R_t: **Relocation distance**. Total kilometers traveled by the operator to rebalance scooters at step t.
- C_t: **Charging cost**. Charging cost in euros at step t, computed from energy consumed (kWh) and the unit electricity price.
- Q_t: **Queue backlog**. Total number of users waiting across all stations at step t; unlike U_t, this penalizes persistent congestion.

The coefficients $w_{(\cdot)}$ represent the importance weights of each objective. To ensure numerical stability and comparable magnitudes between disparate units (ratios, kilometers, Euros, and user counts), each term is normalized by per-network scale factors (U_0, R_0, C_0, Q_0) calibrated as in Table I.

D. Methods

In this section, the implementation choices for the agent-environment interaction loop are described. However, a distinction must first be made between *operation policies* (or planners) and *control policies* (learned policies). They represent connected but distinct layers of decision-making within the system, resulting in confusion regarding their similar terminology. To solve this, a renaming of the former was made.

1) Operation Policies vs. Control Policies: Scooter relocation and charging strategies, with which the agent intervenes in the system, take the form of a list of movements of k scooters between station i and j, and the number of scooters to be charged at each station, respectively. These plans are generated by algorithms referred to as *operation policies*, but more distinctly as *planners*. The planners are heuristic algorithms with explicitly defined rules, in contrast to *control policies* learned through Reinforcement Learning (RL) techniques, which adapt based on experience and feedback from the environment, while mapping observations (system states) to actions.

Furthermore, the agent's action vector contains the parameters used by the planners to generate these relocation and charging plans, $\mathbf{a}_t = [a_0, a_1, a_2, a_3]$. Therefore, although the generated plans directly affect system behavior, they are not an independent control policy, transforming observations into actions. Instead, they represent an intermediate abstraction layer that heuristically translates the agent's actions into executable interventions in the environment.

As the decision-making authority and optimization objective remain with the agent's learned control policy, the renaming from *operation policies* to *planners* highlights this conceptual separation. It clarifies that planners are not granted the same level of control, even though they derive their actions from the agent's outputs and may appear similar at a nomenclatural level.

2) Agent-Environment Interaction: The simulation environment was implemented according to a standard reinforcement learning interface, similar to that of Gymnasium [4]. This was achieved through the implementation of the `reset` and

`step` methods, which initialize the environment and apply an action, respectively. This design provides better modularity and separation of concerns, and allows for near-seamless integration with RL libraries, only requiring the wrapping of the environment to match the expected input and output data structures' formats.

On the agent's actions, the previously explained abstraction from concrete plans to normalized parameters is crucial. It generalizes and simplifies the agent's decision-making process, allowing it to focus on high-level strategy rather than low-level operational details, enabling the use of continuous control algorithms that can efficiently explore and optimize within this bounded action space.

As a result, the overall interaction loop at each time step t is as follows: the agent selects an action $\mathbf{a}_t$ according to its control policy $\pi(\mathcal{S}_t)$, either including the system state $\mathcal{S}_t$ as input or not, as will be detailed in III-D4. Each of the four components of the action vector is mapped to the specific parameters of the relocation and charging planners being used, which generate the respective plans. These plans are executed within the simulation environment, leading to a state transition to $\mathcal{S}_{t+1}$, and the agent receives a reward based on the defined objective function. This loop continues until the end of the simulation horizon, allowing the agent to learn and adapt its control policy over time.

3) Planners Implemented:

a) Relocation Planners:

- **Greedy Relocation**: Identifies stations with surplus of scooters (fill ratio $>$ `high`) and deficit stations (fill ratio $<$ `low`), then iteratively moves scooters to balance toward a `target` fill ratio. Donor stations are prioritized by proximity. The action components map to: $a_0 \rightarrow$ `low` $\in [0.1, 0.3]$ and $a_1 \rightarrow$ `high` $\in [0.6, 0.9]$, $a_2 \rightarrow$ `target` $\in [0.4, 0.8]$.
- **Budgeted Relocation**: Extends greedy logic with a strict constraint on total relocation distance (`km_budget`), explicitly capping operational costs per time step. This prevents excessive redistribution in cost-sensitive scenarios. By adding complexity to the plan generation, it is used for comparison with greedy relocation.
- **No-op Relocation**: Returns an empty plan, serving as a baseline to quantify the impact of active rebalancing.

b) Charging Planners:

- **Greedy Charging**: Prioritizes stations with high demand and low SoC using score $\lambda_i(1 - s_i)$, allocating plugs in descending order of urgency. The agent controls the fraction of total charging capacity utilized via $a_3 \rightarrow$ `charge_budget_frac` $\in [0.05, 0.40]$. This ensures scooters are charged where they are most needed.
- **Slack Charging**: Applies a demand-gating mechanism, only charging at stations below a demand quantile threshold, and prioritizing low SoC. This mitigates availability loss by concentrating charging at low-demand stations during off-peak periods. The goal was to approach the charging strategy from the opposite perspective of greedy charging, and compare their performances.
- **No-op Charging**: As a baseline, plugs zero scooters, isolating the effect of charging decisions in comparative experiments.

4) Agents Implemented: The system evaluates four distinct agents, each representing a different approach to learning planner parameters.

a) Heuristic Agent: A rule-based reactive controller that computes actions from observation features without learning. It maintains exponentially weighted moving averages (EWMA) of recent relocation distance and queue rate to detect operational trends. Actions are derived through explicit logic and change every simulation tick: relocation thresholds (a_0, a_1) adapt to queue pressure and capacity imbalance, the target fill ratio (a_2) responds to peak-hour detection and system throttling, and charging intensity (a_3) scales with low-SoC tail percentiles. This agent represents white-box decision-making, through manually crafted rules based on domain knowledge, providing interpretability and absolute transparency.

b) Episode-Level Bandit (UCB1): A stateless multi-armed bandit that selects discrete planner parameter combinations, in the form a fixed action (arm) that is selected every episode. The agent applies Upper Confidence Bound (UCB1) selection, balancing exploitation of high-performing arms with exploration of ugntried configurations. After all episodes are run, the agent chooses to the arm with the highest average reward. As a very simple RL approach, it serves as a baseline to analyze context-aware and continuous control agents.

c) Contextual Bandit with Receding Horizon (LinUCB): A context-aware bandit that selects arms within each simulation episode based on aggregated system features. The agent extracts a 16-dimensional context vector from observations (fill ratio statistics, SoC percentiles, queue metrics, time-of-day) and applies disjoint LinUCB with per-arm linear models. Arms correspond to discrete actions, reselected every `block_minutes`. This receding-horizon strategy enables intra-episode adaptation to evolving conditions (e.g., shifting from low to high charging budget as SoC degrades), while maintaining exploration through ridge-regularized least-squares updates. The use of context and more frequent decision-making helps the agent respond quickly to changing conditions, while still learning effectively from limited data. This makes it a good choice for environments such as this project's, where demand patterns change constantly.

d) Soft Actor-Critic (SAC): A Deep RL agent that learns a continuous control policy through off-policy actor-critic optimization. Observations are flattened into a single vector from per-station state and temporal features. The agent can use frame-skipping, reducing decision frequency to stabilize learning in long-horizon episodes and counteract noise from the stochastic processes. SAC's entropy-regularized objective encourages exploration while maximizing cumulative reward, enabling it to discover different action sequences that balance service quality and operational cost across the different scenarios. This model was chosen for its strong performance in continuous control tasks, and for being able to capture relationships such as the trade-offs between relocation aggressiveness

and charging intensity, which is expected to yield superior results in a complex, stochastic environment such as this one.

5) Implementation Tools: The experimental simulation was implemented in Python, using the following libraries:

- **Gymnasium** [4]: The environment follows closely the standard Gymnasium interface, exposing the state representation observed and the action space to facilitate the integration of RL agents.
- **StableBaselines3** [5]: Used for implementing advanced learning-based controllers, specifically the SAC agent used for continuous control experiments.
- **NumPy** [6]: For high-performance vectorized state updates, numerical computations and stochastic generations.
- **NetworkX** [7]: For graph traversal, distance matrix computation, and managing network connectivity.

E. External Data

The simulation environment is grounded in realistic topological data. The road network is extracted using OSMnx [8] from real cities, specifically selecting the bike-accessible network, as it most closely matches the one used by electric scooters. Each node is connected to every other node, with the edges retaining the shortest-path distance between them. This is done to abstract away the minute details of routing, which provide little meaning to the problem at hand, while preserving realistic inter-station distances. The travel time between stations is derived from these distances, assuming a constant average speed of 15 km/h, typical for shared electrical scooters in urban settings, and within the speed limits defined in Portuguese traffic legislation. [9].

F. Solution Design and Experimental Setup

The simulation experiments were conducted using several environment configurations, each defined in a dedicated YAML file. Table I summarizes the main parameters for all tested scenarios.

Additionally, all configurations include the full distance and travel time matrices, as well as the geographic coordinates and identifiers (IDs) for each station.

Four scenarios were designed to evaluate agent robustness under varying demand conditions:

- **Baseline**: Homogeneous demand with standard time-of-day variation across all stations, representing typical operational conditions with uniform λ_i and regular diurnal patterns.
- **Hotspot OD**: Introduces spatial demand heterogeneity designating specific stations as popular destinations, increasing their incoming trip probabilities in the origin-destination matrix P. This simulates real-world hotspots like commercial centers or transit hubs that attract disproportionate demand.
- **Heterogeneous-λ**: Assigns non-uniform base arrival rates λ_i across stations, modeling intrinsic differences in station popularity due to location attributes (e.g., transit hubs vs. residential areas).

TABLE I
SUMMARY OF ENVIRONMENT CONFIGURATIONS USED IN EXPERIMENTS (PHASE A: 24H; PHASE B: 168H).

Config	Porto10[a]	Porto20[b]	Porto20[c]	Lisbon20[d]
Stations, N	10	20	20	20
Time step, Δt	5	5	5	5
Horizon, H (hours)	24 / 168	24 / 168	24 / 168	24 / 168
Capacity p/station, C	12	12	12	12
Chargers p/station, K	12	12	12	12
Charge rate, ρ	0.35	0.35	0.35	0.35
Battery, B	0.50	0.50	0.50	0.50
Energy cost, c_{energy}	0.20	0.20	0.20	0.20
Base demand, λ	0.35	0.35	0.35	0.35
Base seed	42	42	42	42
Scoring weights				
w_{unavail}	5.0	5.0	5.0	5.0
w_{reloc}	1.0	1.0	1.0	1.0
w_{charge}	1.0	1.0	1.0	1.0
w_{queue}	4.0	4.0	4.0	4.0
Normalization scales				
U_0 (unavail.)	0.484	0.720	0.746	0.250
R_0 (reloc. km)	0.809	0.965	0.816	1.00
C_0 (charge €)	0.00664	0.0279	0.0273	0.0100
Q_0 (queue)	49.13	205.71	234.70	10.00

Minimum distance between any pair of stations: [a] 500m, [b] 600m, [c] 900m, [d] 700m

- **Event Heavy**: Applies stochastic event multipliers $f_{event}(t)$ on top of baseline demand, emulating even more extreme surges from concerts, sports matches, or weather disruptions that stress system capacity and charging reserves. Note that this does not increase demand on stations that are not affected by events.

IV. RESULTS AND DISCUSSION

A. Experimental protocol, artifacts, and aggregation

All methods are evaluated in the agent-planner-simulator loop described in Section III-D2. Two horizons are considered: a short-horizon setting of 24 hours (Phase A) and a long-horizon setting of 168 hours (Phase B), both using a 5-minute time step. For each `(horizon, network, scenario, method)` configuration, multiple independent random seeds are executed (10–30, depending on runtime) to estimate mean performance and variability under stochastic demand, weather, and event realizations.

Each run produces a structured JSON file, with one JSON file per scenario. Baseline planners are evaluated via `scripts.eval_policy.py`; learning-based controllers and adaptive heuristics (Heuristic, UCB1, LinUCB, SAC) store both a method-level summary and, when applicable, traces (e.g., bandit arm selections). Aggregation is performed by `analyze_results.py`, which normalizes run schemas, and exports two primary artifacts: `master_runs.csv` (objective and KPIs with mean±std over seeds) and `robustness.csv` (scenario deltas relative to Baseline for each `(phase, network, method)`).

The results in this section report Phase A (24h) and Phase B (168h) evaluation outputs. Intermediate training/debug phases are excluded to avoid confounding evaluation statistics.

B. Objective and reported metrics

We rank methods by the episode objective J_{run} defined in Section III-C5 (Eq. (4)); lower is better. Because the normalization scales (U_0, R_0, C_0, Q_0) are calibrated per network (Table I), absolute objective values are most meaningful when comparing methods *within* the same network.

To interpret trade-offs that can yield similar objective values, we additionally report the underlying service and operational KPIs: availability $A = 1 - U$ (and unavailability U), unmet-demand summaries, queue backlog statistics, relocation distance, and charging cost.

C. Main results under Baseline demand

Tables II and III summarize Baseline-scenario performance for 24h and 168h horizons, respectively. "Best baseline" denotes the best-performing *static* planner configuration within the baseline family for the same network and horizon (selected by minimum Baseline J_{run}). The key interpretation points are the following.

First, tuned static planners are consistently competitive in the Baseline scenario. This indicates that the planner abstraction is effective: a significant fraction of achievable performance can be obtained by selecting appropriate relocation and charging intensities, even without stateful learning. This is operationally relevant, as static planners are simple to deploy and easy to audit.

Second, interpretable adaptive controllers can closely track the best-performing methods under Baseline demand. In particular, the tick-level heuristic frequently matches the best baseline and remains competitive with SAC across multiple networks. This supports the practical value of white-box control: high performance can be achieved with transparent rules when demand is near the assumptions encoded by the planner parameterizations.

Third, long-horizon evaluation amplifies differences between controllers. Over 168 hours, small systematic biases (e.g., under-charging, delayed relocation response) can compound into persistent backlogs. As a result, methods that appear comparable at 24 hours can separate more clearly at 168 hours, especially on larger or sparser graphs.

Quantitatively, rankings vary by network and horizon. On Porto10, SAC is the best method under Baseline demand for both horizons: at 24h it attains 10.97 ± 1.73 versus the best static baseline at 11.41 ± 2.42, and at 168h it attains 10.85 ± 1.03 versus 13.61 ± 1.80 (Tables II–III). In contrast, on Porto20 (600m) the best static baseline remains strongest (24h: 9.96 ± 0.87; 168h: 23.11 ± 2.93), while SAC and the heuristic are moderately worse on the long horizon (both ≈ 27). For the sparser Porto20 (900m) and Lisbon20 networks, long-horizon objectives increase substantially and the best baseline remains the strongest reference (e.g., Porto20 (900m) 168h: 682.30 ± 81.72), whereas LinUCB exhibits occasional catastrophic behavior (e.g., Lisbon20 168h: 2328.75 ± 553.70), consistent with instability under long-horizon feedback and exploration.

TABLE II
BASELINE-SCENARIO OBJECTIVE J_{run} FOR THE 24-HOUR HORIZON (MEAN $\pm$ STD OVER SEEDS). LOWER IS BETTER.

Method	Porto10	Porto20 (600m)	Porto20 (900m)	Lisbon20
Best baseline	11.41 ± 2.42	9.96 ± 0.87	119.23 ± 22.76	98.43 ± 17.64
Heuristic	11.51 ± 1.39	11.15 ± 0.95	126.48 ± 22.81	99.54 ± 18.95
UCB1 bandit	12.35 ± 1.74	10.91 ± 1.23	147.72 ± 26.96	119.57 ± 23.37
LinUCB bandit	11.33 ± 2.76	13.30 ± 1.34	138.64 ± 28.52	110.61 ± 33.99
SAC	10.97 ± 1.73	10.87 ± 1.04	129.02 ± 24.58	104.63 ± 22.78

TABLE III
BASELINE-SCENARIO OBJECTIVE J_{run} FOR THE 168-HOUR HORIZON (MEAN $\pm$ STD OVER SEEDS). LOWER IS BETTER.

Method	Porto10	Porto20 (600m)	Porto20 (900m)	Lisbon20
Best baseline	13.61 ± 1.80	23.11 ± 2.93	682.30 ± 81.72	438.01 ± 68.67
Heuristic	13.01 ± 1.53	27.26 ± 2.93	740.86 ± 83.92	518.53 ± 71.23
UCB1 bandit	14.78 ± 2.78	44.13 ± 4.25	1075.04 ± 92.42	879.80 ± 88.14
LinUCB bandit	173.41 ± 101.15	125.09 ± 31.02	730.37 ± 127.13	2328.75 ± 553.70
SAC	10.85 ± 1.03	27.28 ± 5.69	757.22 ± 128.84	528.16 ± 122.67

D. Robustness to scenario shift

Robustness is assessed as scenario-induced degradation relative to Baseline for the same network, horizon, and method:

$$\Delta J(\text{scenario}) = J(\text{scenario}) - J(\text{baseline}). \tag{6}$$

This delta is computed per `(phase, network, method)` and materialized in `robustness.csv`. Reporting deltas avoids conflating scenario effects with per-network normalization differences.

At the scenario level, persistent spatial heterogeneity dominates long-horizon degradation, but the ordering depends on how heterogeneity is introduced. For Porto10 at 168h, the selected best static baseline (`sweep__km10__c0`) attains $J_{\mathrm{run}} = 13.61 \pm 1.80$ under Baseline demand. Under scenario shift, Event Heavy produces only a minor change ($\Delta J_{\text{event}} = 0.41$, i.e., $J_{\mathrm{run}} = 14.02$), whereas sustained spatial imbalance causes large degradation: Heterogeneous-λ increases the objective by $\Delta J_{\text{hetero}} = 68.03$ ($J_{\mathrm{run}} = 81.64$), and Hotspot OD is the most damaging with $\Delta J_{\text{hotspot}} = 76.27$ ($J_{\mathrm{run}} = 89.88$), as reported by `dJ_vs_baseline` in `robustness.csv`.

Table IV reports the aggregate robustness summary (mean$\pm$std over all network instances and stress scenarios) for each method. Two patterns are consistent with the system dynamics. First, persistent spatial heterogeneity tends to be more damaging than transient event bursts, because chronic imbalance accumulates queue backlog and sustained shortage at high-demand stations. Second, long-horizon robustness is materially harder than short-horizon robustness: a controller that is slightly miscalibrated can accumulate deficits and never fully recover over 168 hours.

The magnitude of degradation increases substantially from 24h to 168h. For instance, the heuristic increases from $\Delta J = 31.67 \pm 36.61$ at 24h to 141.10 ± 214.31 at 168h, and SAC exhibits a similar rise from 32.39 ± 38.89 to 141.30 ± 255.05 (Table IV). The episodic UCB1 bandit also degrades more strongly on the long horizon (142.73 ± 200.92 versus 37.29 ± 44.94), reflecting its inability to adapt within an episode when mismatch persists.

TABLE IV
ROBUSTNESS TO SCENARIO SHIFT (MEAN ± STD).
$\Delta J = J(\text{SCENARIO}) - J(\text{BASELINE})$; LOWER IS BETTER.

Method	ΔJ (24h)	ΔJ (168h)
Best baseline	46.26 ± 60.42	291.50 ± 392.69
Heuristic	31.67 ± 36.61	141.10 ± 214.31
UCB1 bandit	37.29 ± 44.94	142.73 ± 200.92
LinUCB bandit	34.36 ± 43.91	-42.71 ± 751.25
SAC	32.39 ± 38.89	141.30 ± 255.05

A critical outcome is the variability of LinUCB on the 168-hour horizon: its aggregate robustness summary is $\Delta J = -42.71 \pm 751.25$ (Table IV), where the negative mean indicates that some stress scenarios can be easier than Baseline for LinUCB in certain configurations, but the extremely large standard deviation evidences occasional catastrophic failures. This supports the need for explicit stability controls (e.g., conservative exploration schedules, feature normalization, or fallback modes) before deployment.

E. Objective decomposition and failure modes

To diagnose why methods succeed or fail, we rely on objective decomposition plots that report the mean contribution of each normalized term (unavailability, relocation, charging, queue). As per Figure 1, the dominant long-horizon failure mode is *queue accumulation*: once persistent backlog forms under spatial heterogeneity, the queue term dominates the objective and overwhelms modest savings in relocation or charging costs. In practical terms, controllers that under-intervene may appear cost-effective early in an episode but eventually enter a queue-dominated regime from which recovery is slow.

Conversely, strong performance in hard scenarios typically requires sustained corrective action: either more frequent and/or more aggressive relocation, or charging patterns that preserve usable SoC at high-demand stations. This trade-off is central to the management problem: improved service quality generally requires paying operational effort, and the decomposition makes that exchange explicit.

F. Bandit behavior and interpretability

Bandit-based controllers provide an interpretable decision layer because each arm corresponds to a discrete operational mode (a fixed planner-parameter configuration). The episodic UCB1 bandit selects one arm per episode and updates its estimates at the end of the episode, trading off exploration and exploitation via an upper-confidence rule. As a result, UCB1 typically converges toward a small subset of arms, concentrating pulls on modes that deliver stable average returns, as illustrated in Figure 2. This behavior is expected and operationally useful: it identifies good static regimes without requiring gradient-based training.

However, episodic bandits are limited by design. Because they commit to a single mode for an entire episode, they cannot adapt within an episode to diurnal cycles, evolving SoC distributions, or localized event surges. Their achievable performance is also bounded by the expressiveness of the arm set: if no arm encodes the necessary intervention pattern for severe heterogeneity, the bandit cannot match more expressive adaptive controllers.

V. CONCLUSION AND FUTURE WORK

A. Conclusions

This work developed a realistic discrete-time simulator for shared electric scooter fleet management and used it to evaluate multiple controller families under uncertainty, including weather variation, stochastic events, and spatially heterogeneous demand. Evaluation across two horizons (24h and 168h) and multiple network instances demonstrates that controller ranking is not invariant: methods that are competitive under nominal Baseline demand may degrade under scenario shift, especially when heterogeneity is persistent over time.

A core empirical takeaway is that long-horizon performance is dominated by the ability to prevent *queue accumulation* under sustained imbalance. Objective decomposition plots show that once backlog forms, the queue term dominates J_{run} and recovery becomes slow, making under-intervention a high-risk strategy even if it reduces short-term operational cost. Conversely, improved service quality generally requires paying relocation and/or charging effort, and the decomposition makes these trade-offs explicit rather than implicit.

Finally, interpretability introduces a practical trade-off. The heuristic and bandit-based controllers can be audited and explained, and can perform competitively in benign regimes. However, bandits are constrained by the expressiveness of their arm set and, in the episodic case, by their inability to adapt within an episode. SAC can learn richer, continuous parameter schedules, but introduces opacity and requires additional safeguards for real-world deployment.

B. Main contributions

Scientifically, this report contributes a scenario-based evaluation methodology that separates nominal performance from robustness via within-configuration deltas ΔJ, complemented by objective decomposition to support causal interpretation of failure modes.

From an application perspective, the simulator clarifies how operational constraints (finite capacity, minimum SoC, charging delay, relocation effort) shape achievable service levels, and how different controllers allocate intervention effort under demand heterogeneity.

Technologically, the work provides a reproducible experimentation pipeline (scenario generation, controller evaluation, structured JSON logging, automated aggregation to CSV, and plot generation), enabling rapid iteration while maintaining an audit trail from raw runs to reported results.

C. Future work

Future work should prioritize (i) operational realism and (ii) robustness under heterogeneity. On the modeling side, relocation should be constrained by explicit operational resources (truck capacity, routing, staffing, and time windows).

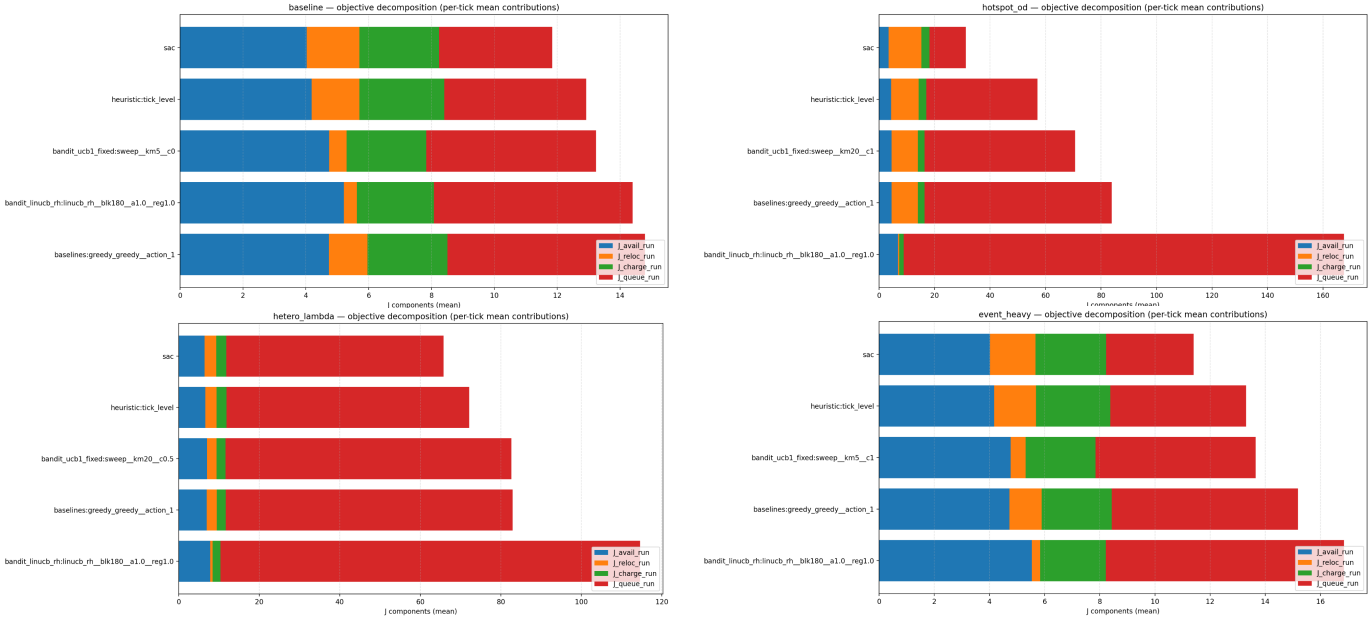

Fig. 1. Objective decomposition plots for the evaluated scenarios. Each plot reports the per-tick mean contribution of the normalized objective terms (unavailability, relocation, charging, and queue) for each method. Scenario identification is provided in the plot titles.

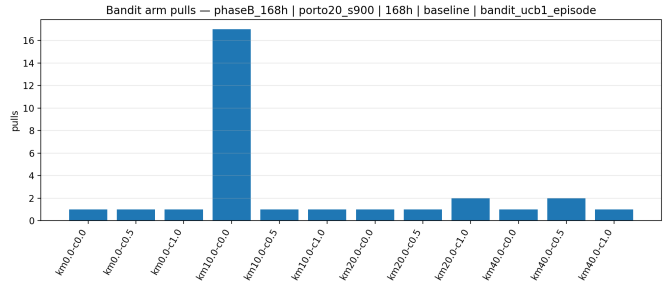

Fig. 2. Episodic UCB1 arm pulls (Porto20_s900, 168h). Concentration of pulls indicates convergence to a small subset of discrete operational modes.

On the control side, contextual bandits should incorporate stronger feature normalization and richer context (including graph-aware features) to reduce long-horizon variance and improve regime discrimination. For SAC, improvements include longer training budgets, curriculum training from easier to harder scenarios, and state representations that better capture network structure. Finally, the simulator can be extended with operational levers such as dynamic pricing, user incentives, and proactive repositioning, enabling new research directions in end-to-end micromobility optimization under uncertainty.

References

[1] S. Mijnster, "The potential of optimization and simulation to better match supply and demand in shared mobility systems," MSc Literature Review, Delft University of Technology, Jul. 2022, report number: 2022.MME.8683. [Online]. Available: https://repository.tudelft.nl/file/File_12461ccf-060e-436b-84c3-a9142c8c1b85?preview=1

[2] Z. T. Qin, H. Zhu, and J. Ye, "Reinforcement learning for ridesharing: An extended survey," *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103852, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X22002716

[3] L. M. Villa-Zapata, D. Rodriguez-Roman, J. E. Flórez-Coronel, J. M. González-López, and A. M. Figueroa-Medina, "Incorporating equity in the vehicle rebalancing operations of dockless micromobility services," *Latin American Transport Studies*, vol. 2, p. 100009, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2950024924000015

[4] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[5] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[7] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[8] G. Boeing, "Modeling and analyzing urban networks and amenities with osmnx," *Geographical Analysis*, vol. 57, no. 4, pp. 567–577, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/gean.70009

[9] Diário da República, "Decreto-Lei n.º 102-B/2020, de 9 de dezembro," https://diariodarepublica.pt/dr/detalhe/decreto-lei/102-b-2020-150757538, Dec. 2020, diário da República, 1.ª série, n.º 237. Acedido em 10 jan. 2026.