Research Article

Dynamic Capacitated Arc Routing Problem in E-Bike Sharing System: A Monte Carlo Tree Search Approach

Shiqi Tan ¹, Zhiheng Li,² and Na Xie ³

¹Department of Automation, Tsinghua University, Beijing 100084, China

²Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China

³School of Management Science and Engineering, Central University of Finance and Economics, Beijing 100081, China

Correspondence should be addressed to Na Xie; xiena@cufe.edu.cn

Received 17 June 2021; Accepted 25 September 2021; Published 23 October 2021

Academic Editor: Paola Pellegrini

Copyright © 2021 Shiqi Tan et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper studies a dynamic capacitated arc routing problem for battery replacement in an e-bike sharing system, where workers replace batteries for underpowered e-bikes along street segments dynamically. The objective is to replace as many batteries as possible and minimize pickup failures. The temporal dependency of the routing decisions, the conflict of the workers' operations, and the stochastic and dynamic nature of user demands all make this a difficult problem. To cope with these difficulties, a "Partition-First, Route-Second" bi-level solution framework is adopted to describe the problem in two different time scales. On the large time scale, a spatiotemporal partitioning method, which divides the road network into nonoverlapping subzones, is proposed to decompose the problem. On the small time scale, this paper models the routing decision process of individual worker as a Markov Decision Process. We adopt a lookahead policy that simulates future information and decisions over some horizons to evaluate the long-term influence of current feasible decisions. A Monte Carlo Tree Search algorithm is also used to improve the simulation efficiency. By performing numerical computation experiments on a test case study and comparing with some benchmarking policies, we demonstrate the effectiveness and efficiency of the suggested method.

1. Introduction

The "Mobile Internet" and "Sharing Economy" have greatly promoted the development of shared micromobility services, represented by free-floating bike sharing system, shared electric scooter system, and e-bike sharing system, which provide a more convenient and environmentally friendly "start/last mile" travel solution. The E-Bike Sharing System (EBSS) is a relatively new form of shared micromobility service, which could provide auxiliary power through the battery and help users reduce the physical energy consumption when they are riding against the wind, uphill, with loads, or over long distances and thus can significantly improve user experience. In China, the country with one of the world's largest markets for electric bicycles, the shared micromobility service providers, such as Hello-bike and Qingju Bike, have already launched EBSS in many cities, especially those with lots of slopes in the road network.

Despite the improvement in user experience, EBSS faces more challenges in daily operation. A new challenge is how to replace the detachable batteries of underpowered e-bikes scattered throughout the city in a timely and cost-effective way. At present, the EBSS providers in China usually take the approach of setting up "virtual parking spots" as illustrated in Figure 1. The virtual parking spots are areas where parking is allowed as determined by the built-in GPS positioning and are usually distributed along the street segments. Users who park e-bikes outside these areas will face punitive dispatch fees. In this way, it not only reduces the worker's efforts to find an e-bike but also restrains the messy parking behavior.

In the above context, since the underpowered e-bikes are mostly aggregated over the virtual parking spots along the street segments, the routing problem of battery replacement in EBSS can be modeled as a capacitated arc routing problem (CARP), just like the Chinese postman problem [1]. An optional way is to replace batteries at night when the user

FIGURE 1: An illustration of the distribution of virtual parking spots. Users can easily query the surrounding and the nearest virtual parking spots through the EBSS platform.

demands can be ignored, corresponding to a static version of CARP (SCARP). In the static version, the distribution of underpowered e-bikes is fixed and known in advance, and a set of minimum cost routes that service all underpowered e-bikes is expected to be found for a fleet of capacitated vehicles located at a given depot. However, there is a drawback with this passive static way: it cannot guarantee the number of e-bikes with sufficient power maintained at a high level throughout the next day, resulting in lower utilization rates of the system, especially during the evening peak.

Therefore, this paper mainly considers a dynamic version of the CARP (DCARP) in EBSS: the battery replacement is performed during daytime, when the user's participation must be considered and new underpowered e-bikes may stochastically appear after the fleet departs from depot. Although there is a large amount of studies related to the dynamic capacitated vehicle routing problem, which was surveyed by Pillac et al. [2] and Psaraftis et al. [3], direct studies on DCARP are relatively few. Some literatures [4–7] investigated the periodic CARP in certain scenarios and proposed various heuristic algorithms; some literatures [8–12] studied the dynamic case where the demand or deadheading cost of each arc is stochastic and proposed some repair operators or adaptive algorithms rather than just an initial static solution.

However, the DCARP in the scenario of battery replacement in EBSS has the following difficulties: the first difficulty is the temporal dependency of the routing decisions. The current routing decisions for battery replacement will affect the success of subsequent pickup demands, which in turn affects the future state of e-bikes and the future routing decisions. This intricate temporal dependency results in a significant increase in the complexity of the problem. The second difficulty is the conflict of the workers' operations. Since there is usually more than one worker, it is important to design a cooperation mechanism to avoid simultaneous operations on the same street segment by different workers and to improve the operation efficiency throughout the entire system. The third difficulty is how to anticipate the future influence of current decisions with stochastic and dynamic user demands. In the dynamic case, the goal is not only to maximize the number of batteries replaced but also to maximize the successful attempts to pick up e-bikes, and this requires anticipating potential future user demands when making current decisions.

These difficulties make it hard to compute a fixed decision sequence in advance. To cope with these difficulties, this paper adopts a “Partition-First, Route-Second” bi-level solution framework to describe this problem in two different time scales, as shown in Figure 2.

On the upper level, the road network is partitioned into subzones to decompose the problem into multiple smaller subproblems. This idea of partitioning streets into clusters has been widely used in the classical CARP and its variants [13–17]. In the setting of our problem, the road network partition provides constraints of feasible operation area and reward to each worker, preventing the potential conflict of workers' operations. The partitioning also reduces the temporal dependence between different workers' routing decisions to some extent, due to the short-distance nature of e-bike trips. To adapt the partition to the real-time state, a spatiotemporal partitioning method is proposed on a large time scale, where an initial partition is generated at the beginning of a day and will be adjusted dynamically with the real-time state of e-bikes based on some heuristic algorithms.

On the lower level, the routing decision process of each worker is modeled as a Markov Decision Process (MDP) on a small time scale, which is small enough to capture the details of the user demands and worker operations. In the MDP, decisions should be made about which adjacent street segment to visit next when a worker has finished the task on the previous street segment. This paper adopts a lookahead policy with Monte Carlo Tree Search (MCTS) algorithm, which focuses on solving an approximation of the problem over some horizon to anticipate the future impacts [18, 19]. At a decision point of a worker, we model all feasible sequences of routing decisions for the worker over some horizon into a decision tree, then search and simulate it in the simulator based on the MCTS algorithm, which will allocate more simulation runs to more highly rewarding routes, and thus save computational resources in the simulation process.

The main contributions of this paper are as follows. We model the routing problem of battery replacement in EBSS into a DCARP, based on the distribution of virtual parking spots. Then, we adopt a “Partition-First, Route-Second” solution framework to cope with the difficulties of this problem. We take both user satisfaction and operational efficiency into account and adopt a lookahead policy with MCTS algorithm to balance the relationship of exploration and exploitation during the simulation.

The next parts of this paper are structured as follows. In Section 2, we briefly review the related literature. In Section 3, we summarize the notations used in this paper and formulate the problem. In Section 4, a spatiotemporal partitioning method is proposed in detail. In Section 5, we introduce the lookahead policy with MCTS algorithm. In Section 6, we present the design of the system simulator. In Section 7, we analyze the results of numerical studies. Finally, we conclude the paper in Section 8.

2. Literature Review

As a convenient way for short-distance travel, bike sharing system has been expected to help mitigate traffic congestion, alleviate the growing environmental issues, and promote

FIGURE 2: The “Partition-First, Route-Second” bi-level solution framework.

sustainable transportation development [20]. However, the development of the bike sharing systems has also undergone a long trial-and-error process. The first generation of bike sharing systems was dockless and operated as nonprofits but failed due to theft and damage [21]. Then, to facilitate maintenance, station-based bike sharing systems became the dominant form for a long time and went through stages such as coin-deposit systems, IT-based systems, and demand-responsive multimodal systems that made it easier for users to pick up or return bikes [22]. Besides, numerous studies [23–27] have investigated the bike allocation and reposition problems in station-based bike sharing systems to rebalance the system from nonhomogeneous asymmetric demand processes. However, the user base of the station-based bike sharing system has been limited, with the constraint that it is difficult to build dense stations due to the high cost and road space required.

In recent years, with the support of emerging technologies such as mobile payment and electronic locking, the Internet-based free-floating bike sharing system has become the main mode, where users are allowed to pick up and return bikes at any location near their origin and destination. This increases the coverage of the system and attracts more users. However, with the rapid expansion of free-floating bike sharing system, a series of new problems have arisen, such as the recycling of faulty bikes [28] and the messy parking behaviors [29]. Therefore, the promotion of setting virtual parking spots in some areas is necessary [29]. The EBSS, which needs frequent replacement of batteries for underpowered e-bikes, has adopted the approach of setting up virtual parking spots in many cities in China. These virtual parking spots are quite different from the parking stations in traditional station-based bike sharing systems. The virtual parking spots do not require the construction of physical parking stations or docks and thus could be set up more densely throughout the city at a lower cost and in a more flexible way. However, to the best of our knowledge, there is no research on the problem of dynamic battery replacement in the EBSS with virtual parking spot setting so far.

From another perspective, the dynamic battery replacement problem in EBSS is similar to the dynamic

rebalancing problem in traditional station-based bike sharing systems in terms of user demand distribution and system operation mechanism. Thus, it would be helpful to review some of the dynamic rebalancing policies in station-based bike sharing systems. The rebalance problem is usually modeled as a pickup and delivery problem in the literature. Brinkmann et al. [23] proposed a short-term rebalancing policy, and according to this policy, the vehicle is sent to the nearest shared bike station where the safety buffer is breached. This policy is simple and computationally efficient but does not take future information into account. Ghosh et al. [24] simplified the relocation problem by aggregating the stations and trips and proposed a mixed-integer linear program on a rolling horizon. The high computational cost of this algorithm makes it difficult to meet the requirements of real-time applications. Brinkmann et al. [25] presented a dynamic lookahead policy to anticipate potential future demands over a predefined horizon by rollout simulation. However, this policy is less efficient when multiple vehicles should be considered and when the decision process occurs too frequently. Li et al. [26] used neural network to approximate the long-term value function of current decisions via a reinforcement learning approach. This deep reinforcement learning approach has the appeal of fast decision speed and the ability to consider long-term effects. However, it is still a challenging work to learn an accurate and stable value function neural network due to the stochastic exogenous information and complexity of the environment in EBSS.

Taking the distribution of virtual parking spots into consideration, we model the battery replacement problem in EBSS as a dynamic capacitated arc routing problem. The capacitated arc routing problem is aimed to determine an optimal traversal of an arc subset of a graph by some capacitated vehicles [1] and has been widely studied in the past years, such as by converting to the capacitated vehicle routing problem [30, 31], or directly by exact optimization methods [32, 33], or by some lower bounding procedures [34–37] and heuristic methods [13, 38–41]. Among them, the “Partition-First, Route-Second” heuristic has been widely used in CARP and its variants and has been proved to be effective [13–17]. This

paper adopts this heuristic idea to decompose the original problem into some smaller subproblems by partitioning the arcs into some subzones.

The dynamic version of CARP considers the stochastic and dynamic demand or deadheading cost, which greatly increases the problem complexity. Christiansen et al. [8] considered a variant of the CARP with stochastic demands on each arc and proposed a branch-and-price algorithm. They incorporated the stochastic demand into the pricing problem. Laporte et al. [9] proposed an adaptive large-scale neighborhood search heuristic method for the CARP with stochastic demands. Tagmouti et al. [10, 11] studied the CARP with time-dependent service costs and proposed a variable neighborhood descent heuristic method. Mei [12] aimed to find a robust solution for all possible environments instead of the optimal solution for a specific environment for the dynamic CARP and designed a repair operator which could repair the solution to feasible dynamically. However, these methods may fail to anticipate the difficulties of the DCARP in EBSS, such as the temporal dependency of the routing decisions and the influence of the future demand trend. Thus, this paper proposes a lookahead policy based on MCTS algorithm for the DCARP in EBSS, which is an efficient algorithm for sequential decision problems [19].

3. Notation and Problem Formulation

In this section, we summarize the notations used in this paper in Section 3.1, formulate the DCARP in Section 3.2, and introduce the decomposition of the decision process of multiple workers and model the routing decision process of an individual worker as MDP in Section 3.3.

3.1. Notation. The main notations and their values used in this paper are summarized in Table 1.

3.2. Problem Formulation. Figure 3 shows a simple scenario of EBSS, which consists of the road network, e-bikes scattered among the virtual parking spots or used on the road, and vehicles/workers replacing batteries for underpowered e-bikes.

The road network is abstracted as a connected graph $G = (V, E)$, where $V = \{v_0, v_1, v_2, \dots, v_{N^v}\}$ is the vertex set and $E = \{(v_i, v_j) | v_i, v_j \in V \text{ and } i < j\}$ is the edge set. In the vertex set, v_0 is the depot and $\{v_i | i = 1, 2, \dots, N^v\}$ is the set of intersections. Assume that each edge has two directions and the vehicle can only serve one direction at a time. Thus, an edge (v_i, v_j) can be split into two directed arcs: $\langle v_i, v_j \rangle$ and $\langle v_j, v_i \rangle$, as shown by the edge (v_1, v_2) in Figure 3. Then, the road network can also be represented as a directed graph: $G = (V, A)$, where $A = \{v_i, v_j | v_i, v_j \in V \text{ and } i \neq j\}$ is the arc set. The undirected graph representation is used in the road network partitioning and the directed graph representation is used in the routing decision process.

The operating day is divided into T discrete time steps: $t \in [1, 2, \dots, T]$, according to a predefined time interval Δt (1 minute). The dynamic user demand is reflected in the stochastically generated pickup and return demands in each

virtual parking spot at each time step, which leads to dynamic changes in the location and battery power level of e-bikes.

The number of e-bikes in the system is supposed to be fixed at N^{eb} , without new e-bikes being put into the system or removed from the system. The battery range of each e-bike is in the range of $[0, \psi_{\max}]$ (km). When the battery range is below a threshold $\psi_{\min}$ (km), an e-bike is regarded as underpowered and unavailable to users. For the sake of simplicity, we assume that the time cost for a worker to replace a battery is a constant: τ^b (e.g., 1 minute).

Assume there are N^w workers (or vehicles) in the system, denoted by the set $W = \{w_1, w_2, \dots, w_{N^w}\}$. Each vehicle has the same battery capacity of B (e.g., 40). Suppose t_{begin} (e.g., 7:00 am) and t_{end} (e.g., 10:00 pm) are the beginning and the end of working hours. During working hours, workers will continue to replace batteries for underpowered e-bikes. Once the available batteries are used up, workers need to return to the depot and reload batteries. Suppose vehicles travel with a certain speed that is independent of time, leading to a positive deadheading cost c_{ij} of each arc $\langle v_i, v_j \rangle \in A$. Assume that the time cost of reloading a battery in the depot is τ^d (e.g., 0.5 minutes) and there are always enough fully charged batteries in the depot, so the time cost of reloading a vehicle's battery in the depot is $\tau^d \cdot B$.

Under the limited working hours and time cost constraints introduced above, on the one hand, we hope to route the vehicles in a way that replaces as many batteries as possible during a limited time; on the other hand, we also hope to prioritize services for those arcs that have the potential to reduce user pickup failures in the following time. Take worker w_1 in Figure 3 for example; if there will be a user picking up e-bike on arc $\langle v_4, v_7 \rangle$ in the following time, it would be better for w_1 to serve arc $\langle v_4, v_7 \rangle$ first instead of arc $\langle v_6, v_7 \rangle$. However, directly maximizing the sum of the number of replaced batteries and the number of successful pickups may be not appropriate. In fact, replacing one more battery has a higher expected reward than reducing one more pickup failure because a fully charged e-bike can usually be used multiple times and can reduce more potential pickup failures in the following days. For this consideration, the optimization objective function is designed as a weighted sum of these two factors, as shown in the following equation:

$$\max \sum_{t \in [1, 2, \dots, T]} \sum_{\langle v_i, v_j \rangle \in A} [(\psi_{\max} - \psi_{\min}) \cdot b_{tij} + \bar{\psi}_{\text{user}} \cdot d_{tij}], \quad (1)$$

where b_{tij} is the number of batteries replaced on arc $\langle v_i, v_j \rangle$ at time step t , d_{tij} is the number of successful pickups on arc $\langle v_i, v_j \rangle$ at time step t , and $\bar{\psi}_{\text{user}}$ indicates the expectation of battery power consumption for each e-bike trip. Therefore, $\sum_{t \in [1, 2, \dots, T]} \sum_{\langle v_i, v_j \rangle \in A} \bar{\psi}_{\text{user}} \cdot d_{tij}$ denotes the battery power consumed by the users, indicating the short-term rewards during current day, while $\sum_{t \in [1, 2, \dots, T]} \sum_{\langle v_i, v_j \rangle \in A} (\psi_{\max} - \psi_{\min}) \cdot b_{tij}$ denotes the battery power replenished by the workers, indicating the long-term rewards of the system sustainability.

TABLE 1: Notations.

Notation	Meaning
Model	
N^v	Number of intersections of the road network
v_i	Intersection of the road network, for $i \in [1, 2, \dots, N^v]$
v_0	The depot
V	Set of vertexes, including the intersections and depot, $V = \{v_0, v_1, v_2, \dots, v_{N^v}\}$
(v_i, v_j)	Street segment (edge) of the road network $v_i, v_j \in V$,
E	Set of edges, $E = \{(v_i, v_j) v_i, v_j \in V \text{ and } i < j\}$
v_i, v_j	Directed street segment (arc), $v_i, v_j \in V$
A	Set of arcs, $A = \{v_i, v_j v_i, v_j \in V \text{ and } i \neq j\}$
G	Graph of the road network, $G = (V, E)$ in the undirected representation or $G = (V, A)$ in the directed representation
T	Number of time steps in an operating day (1440)
t	Time step on the small time scale, $t \in [1, 2, \dots, T]$
Δt	Interval between time steps (1 min)
$t_{\text{beginning}}$	The beginning of the working hours (7:00 am)
t_{end}	The end of the working hours (10:00 pm)
ψ_{max}	Maximum range of e-bike battery (40 km)
ψ_{min}	Minimum battery range threshold for an e-bike to be available (3 km)
τ^b	Serving time cost of replacing battery for an e-bike (1 minute)
N^{eb}	Number of e-bikes in system (1000)
N^w	Number of workers/vehicles in system (4)
w_i	Worker, $i \in [1, 2, \dots, N^w]$
W	Set of workers, $W = \{w_1, w_2, \dots, w_{N^w}\}$
B	Battery capacity of each vehicle (30)
c_{ij}	Deadheading time cost of arc v_i, v_j
τ^d	Time cost of reloading a battery in the depot (0.5 minutes)
$\bar{\psi}_{\text{user}}$	Expectation of battery range consumption for e-bike trips (2.75 km)
b_{tij}	Number of batteries replaced on arc v_i, v_j at time step t , $t \in [1, 2, \dots, T]$
d_{tij}	Number of successful pickups on arc v_i, v_j at time step t , $t \in [1, 2, \dots, T]$
G_i	Subzone of the road network, $i \in [1, 2, \dots, N^w]$
V_i	Vertex set in subzone G_i , $i \in [1, 2, \dots, N^w]$
A_i	Arc set in subzone G_i , $i \in [1, 2, \dots, N^w]$
E_i	Edge set in subzone G_i , $i \in [1, 2, \dots, N^w]$
p	Time period on the large time scale
Δp	Time interval between time periods (180 minutes)
K^m	The decision point set of worker w_m , $m \in [1, 2, \dots, N^w]$
t_k^m	Time step for the k th decision point of worker w_m , $t_k^m \in [1, 2, \dots, T]$, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
η_k^m	A vector of length $ A $ indicating the available e-bike number on the arcs in the subzone of worker w_m at the k th decision point (the values of arcs in other subzones are masked to 0), $m \in [1, 2, \dots, N^w]$, $k \in K^m$
ϕ_k^m	A vector of length $ A $ indicating the underpowered e-bike number on the arcs in the subzone of worker w_m at the k th decision point (the values of arcs in other subzones are masked to 0), $m \in [1, 2, \dots, N^w]$, $k \in K^m$
L_k^m	A one-hot vector of length $ A $ indicating which arc worker w_m is located on at the k th decision point, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
C_k^m	A nonnegative number indicating the remaining battery number in the vehicle of worker w_m at the k th decision point, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
S^m	State set of worker w_m , $m \in [1, 2, \dots, N^w]$
s_k^m	State of worker w_m at the k th decision point, $s_k^m \in S^m$, and $s_k^m = (t_k^m, \eta_k^m, \phi_k^m, L_k^m, C_k^m)$, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
$X_{s_k^m}$	Feasible decision set under state s_k^m
x_k^m	Decision of worker w_m at the k th decision point, x_k^m is a one-hot vector of length $ A $ indicating the arc to visit next, $x_k^m \in X_{s_k^m}$, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
r_k^m	Instant reward for worker w_m at the k th decision point, $m \in [1, 2, \dots, N^w]$, $k \in K^m$
π_m	Policy for worker w_m , $m \in [1, 2, \dots, N^w]$
π_m^*	Optimal policy for worker w_m , $m \in [1, 2, \dots, N^w]$
Method	
m_{ij}	The link relationship of the i th and the j th edge in E , $i, j \in [1, 2, \dots, E]$
M	Adjacency matrix of edge set, $M = (m_{ij})_{ E \times E }$
e_{ij}	A 0-1 variable to indicate whether the j th edge in E belongs to the subset E_i , $i \in [1, 2, \dots, N^w]$, $j \in [1, 2, \dots, E]$
e_i	A 0-1 vector to indicate the edges in subset E_i , $e_i = (e_{ij})_{j \in [1, 2, \dots, E]}$, $i \in [1, 2, \dots, N^w]$
u_{pj}	The number of underpowered e-bikes on the j th edge at time period p , $j \in [1, 2, \dots, E]$
u_p	The vector of the number of underpowered e-bikes on each edge at time period p , $u_p = (u_{pj})_{j \in [1, 2, \dots, E]}$
l_i	The length of the i th edge (km), $i \in [1, 2, \dots, E]$

TABLE 1: Continued.

Notation	Meaning
l	The vector of the length of each edge, $l = (l_i)_{\forall i \in [1, 2, \dots, E]}$
β	Temperature in the simulated annealing algorithm
β_0	Initial temperature in the simulated annealing algorithm (100)
α	The decay factor in temperature in the simulated annealing algorithm (0.9)
$\text{MaxIter}^{\text{inner}}$	Inner iteration limit in the simulated annealing algorithm (10)
$\text{MaxIter}^{\text{outer}}$	Outer iteration limit in the simulated annealing algorithm (20)
I_r	Simulation runs for each feasible decision in the rollout policy
$I_{\max}$	Maximum iteration number in the MCTS algorithm
H	Lookahead simulation horizons (180 time steps)
γ	A weighting factor in the UCB algorithm (5)
λ	A time discount parameter (0.95)

FIGURE 3: A sample scenario of EBSS.

3.3. Decomposition of the Problem. For the above dynamic capacitated arc routing problem, it is hard to compute the optimal route for each worker in advance due to the stochastic and complex nature of the system. Therefore, we seek to formalize the problem as a sequential decision process in cooperative multiagent system. A decision point occurs when a worker completes the task on the current arc and a decision about which one of the adjacent arcs to visit next should be made. It should be noted that as the number of tasks and deadheading cost varies for each arc, the executing duration of each decision may be different and the decision processes between different workers are thus asynchronous. At each time step, only a subset of the workers may make a decision while others keep on executing the tasks on the current arcs.

To simplify the problem, we assume that each worker is assigned a nonoverlapping subzone and is allowed to perform the battery replacement operation only within his

subzone. As there are N^w workers, the road network $G = (V, A)$ will be partitioned into N^w subzones: $\{G_1, G_2, \dots, G_{N^w}\}$, satisfying the condition that $G_i = (V_i, A_i)$ is strongly connected graph for $i = 1, 2, \dots, N^w$ and $A_1 \cup A_2 \cup \dots \cup A_{N^w} = A$, $A_i \cap A_j = \emptyset$ for $i \neq j$. In this case, the routing decisions of each worker have very limited effects on other workers, and we can decompose the cooperative multiagent decision problem in the whole road network into multiple mono-agent decision problems in each subzone approximately.

As shown in Figure 4, the arc set A is partitioned into two subsets A_1 and A_2 for workers w_1 and w_2 , and the decision process of each worker is modeled as a Markov Decision Process. The operating day is divided into discrete time steps t according to a small time interval Δt (e.g., 1 minute). At each time step, a worker can make a routing decision or continue with his current task. For example, the decision points for worker w_1 in Figure 4 are time steps $t + 2, t + 4$,

FIGURE 4: An illustration of the decomposition of decision processes.

and $t + 6$. We also divide the day into some discrete time periods p according to a larger time interval Δp (e.g., 180 minutes). At each time period (e.g., at p_1 and p_2 in Figure 4), the partition can be adjusted according to the real-time e-bike state.

To model the routing decision process of each worker as a MDP, the following assumptions are made:

- (i) After a worker leaves the depot, he follows the shortest path (i.e., calculated by the Dijkstra algorithm) to the nearest vertex of his subzone. During this process, the worker does not replace battery. Once the remaining batteries in the vehicle run out, the worker follows the shortest path back to the depot to reload the batteries and then goes to the previous location and continues his operation.
- (ii) As new underpowered e-bikes may be generated on the arcs that have been served, multiple visits and multiple services to an arc are allowed. However, we suppose that temporary stops of vehicles are not allowed and that the vehicle's power is always sufficient during working hours.

Based on the above assumptions, the MDP for the m th worker w_m consists of the following components:

A state: S^m is the feasible state set of worker w_m . $s_k^m \in S^m$ is state of the k th decision point of worker w_m . The state $s_k^m = (t_k^m, \eta_k^m, \varphi_k^m, L_k^m, C_k^m)$ includes three main parts: the system time, the state of e-bikes, and the state of workers. The time step of the k th decision point of worker w_m is denoted as t_k^m ; the state of e-bikes includes η_k^m and φ_k^m , indicating the available e-bike number and the underpowered e-bike number on the arcs in the subzone of worker w_m at the k th decision point, respectively. The state of worker includes the locations L_k^m and the remaining battery number C_k^m of worker w_m at the k th decision point. A detailed description of the state information is given in Table 1.

A decision: $X_{s_k^m}^m$ denotes the feasible decision set under state s_k^m . $X_{s_k^m}^m$ is the intersection of the set of arcs adjacent to the current arc of w_m and the set of arcs in the

subzone of w_m . For example, in Figure 4, $\{v_4, v_0, v_4, v_1, v_4, v_7\}$ is the set of arcs adjacent to v_7, v_4 . However, v_4, v_1 belongs to different subzones from v_7, v_4 . Therefore, when a worker is making a routing decision on arc v_7, v_4 , the feasible decision set is $\{v_4, v_0, v_4, v_7\}$. The k th decision of worker w_m is denoted as $x_k^m \in X_{s_k^m}^m$, representing the next arc to visit.

State transition function: $P^m(s_{k+1}^m | s_k^m, x_k^m)$ is the transition probability function for worker w_m and describes the probability that the system will move to the next decision state s_{k+1}^m after making a decision x_k^m under the current decision state s_k^m . The next decision state s_{k+1}^m will also be influenced by the stochastic exogenous information in the system (e.g., the stochastic user demand) and the operations of other workers.

Reward function: $r_k^m = R^m(s_k^m, x_k^m)$ is the instant reward for worker w_m at the k th decision point. $R^m(s_k^m, x_k^m)$ denotes the reward function for worker w_m , reflecting the instant reward received after taking an decision x_k^m under state s_k^m . r_k^m can be derived from the weighted sum of the number of successful pickups and the number of batteries replaced within the subzone of w_m between time step t_k^m and t_{k+1}^m (if k is the last decision point of w_m , then t_{k+1}^m is the last time step of the day), as shown in the following equation:

$$r_k^m = \sum_{t=t_k^m}^{t_{k+1}^m-1} \sum_{v_i, v_j \in A_m} [(\psi_{\max} - \psi_{\min}) \cdot b_{tij} + \bar{\psi}_{\text{user}} \cdot d_{tij}]. \quad (2)$$

Since the subzones do not overlap, the cumulative reward in a subzone is largely related to the decisions of the worker in that subzone and less related to the decisions of other workers, and the sum of the cumulative rewards of all subzones is the total system reward, which is consistent with the optimization objective in equation (1).

Policy: π_m describes the policy to make a decision under current state for worker w_m : $x_k^m = \pi_m(s_k^m)$.

Suppose that s_0^m is the initial state of worker w_m and K^m is the decision point set of worker w_m . For worker w_m , our goal is to find an optimal policy π_m^* that maximizes his expected accumulation of rewards:

$$\pi_m^* = \operatorname{argmax}_{\pi_m} \mathbb{E} \left[\sum_{k \in K^m} R^m(s_k^m, \pi_m(s_k^m)) | s_0^m \right]. \quad (3)$$

4. Spatiotemporal Partitioning Method

As introduced in the previous sections, on the upper level of the solution framework, we decompose the decision processes of multiple workers by partitioning the road network into subzones. Therefore, the way the road network is partitioned has a great impact on the allocation of labor force and operation efficiency. In this section, we first propose the objective function of the road network partitioning in Section 4.1 and then introduce the algorithm of initial partition generation and partition adjustment at each time period in Sections 4.2 and 4.3.

4.1. Objective. The objective of road network partitioning should not only take into account the static topology of the road network but also make the amount of tasks, i.e., the number of underpowered e-bikes, in each subzone as consistent as possible. This is to avoid situations where some workers are overworked while others are underworked. We use an edge (i.e., a street segment) as the partition unit, so the two arcs of an edge will always be divided into the same subzone. Therefore, the edge set E will be partitioned into N^w (i.e., the number of workers) subsets: $E_1, E_2, \dots, E_{N^w}$, satisfying that $E_1 \cup E_2 \cup \dots \cup E_{N^w} = E$, $E_i \cap E_j = \emptyset$ for $i \neq j$, and the subgraph induced by E_i is connected graph for $i = 1, 2, \dots, N^w$. If the subgraph G_i induced by above E_i is

represented in the form of a directed graph: $G_i = (V_i, A_i)$, it is obvious that $G_i = (V_i, A_i)$ satisfies the constraints for the subzone described in Section 3.3.

Since the graph is partitioned on edges instead of vertexes, here we use the adjacent matrix $M = (m_{ij})_{|E| \times |E|}$ to represent the link relationship of edges in set E . $m_{ij} = 1$ indicates that the i th and j th edges in E are linked; otherwise, $m_{ij} = 0$. An edge is linked to another edge when and only when they have common vertexes. Suppose an edge is not linked to itself. For example, in Figure 3, edge (v_5, v_8) is linked to edges (v_0, v_5) , (v_2, v_5) , (v_6, v_8) , and (v_8, v_9) .

We chose three indicators to evaluate the road network partition: the first indicator is that the number of underpowered e-bikes in each subzone should be as similar as possible, which facilitates the efficiency of battery replacement and makes the workers' income (which is usually related to how many batteries they replaced) as even as possible. The second indicator is that the sum of the lengths of the edges in each subzone should be as similar as possible to make each subzone similar in size. The third indicator is that the links between subzones should be as few as possible, which ensures that the street segments in each subzone are as closely connected as possible.

The 0-1 vectors $e_i = (e_{ij})_{j \in [1, 2, \dots, |E|]}$ for $i = 1, 2, \dots, N^w$ are the decision variables for the partitioning, indicating the edges belonging to subset E_i . $e_{ij} = 1$ indicates the j th edge belonging to the subset E_i ; otherwise, $e_{ij} = 0$. Also, we use vector $u_p = (u_{pi})_{i \in [1, 2, \dots, |E|]}$ to represent the real-time underpowered e-bike number on each edge at time period p and use $l = (l_i)_{i \in [1, 2, \dots, |E|]}$ to represent the length of each edge. Let g represent a partition solution, i.e., a realization of the decision variables $e_1, e_2, \dots, e_{N^w}$. At a new time period p , the following objective of the partition is designed to be optimized:

$$\min_{e_1, e_2, \dots, e_{N^w}} \text{TaskCut}(g) = \frac{\phi_1}{N^w} \sum_{i=1}^{N^w} \left(\frac{e_i^T u_p}{1^T u_p} - \frac{1}{N^w} \right)^2 + \frac{\phi_2}{N^w} \sum_{i=1}^{N^w} \left(\frac{e_i^T l}{1^T l} - \frac{1}{N^w} \right)^2 + \phi_3 \sum_{i=1}^{N^w} \frac{e_i^T M (1 - e_i)}{1^T M 1}, \quad (4)$$

$$\text{s.t. } \sum_{i=1}^{N^w} e_{ij} = 1, \quad \forall j \in [1, 2, \dots, |E|], \quad (5)$$

$$e_{ij} \in \{0, 1\}, \quad \forall i \in [1, 2, \dots, N^w], \forall j \in [1, 2, \dots, |E|], \quad (6)$$

$$\text{Connected}(M[e_i, e_i]) = 1, \quad \forall i \in [1, 2, \dots, N^w], \quad (7)$$

where $e_i^T u_p$ denotes the underpowered e-bike number in E_i at time period p , $e_i^T l$ denotes the total length of edges of E_i , $e_i^T M (1 - e_i)$ denotes the link number between E_i and the complement of E_i in E , and ϕ_1 , ϕ_2 , and ϕ_3 are the weight coefficients of the three indicators, respectively. Therefore, in equation (4), the first term represents the variance of the underpowered e-bike number (after normalization) in each subzone, the second term is the variance of the total length of edges (after normalization) in each subzone, the third term is

the ratio of the link number between subzones and the total link number in the road network.

Equation (5) ensures that each edge can only be divided into exactly one subzone, and equation (6) ensures the decision variables are 0-1 integers.

In equation (7), $M[e_i, e_i]$ is a submatrix of M which consists of the j th row and column of M where $e_{ij} = 1$, representing the adjacent matrix of edges in set E_i . The $\text{Connected}(M[e_i, e_i])$ function is used to determine whether

the subgraph induced by the adjacent matrix $M[e_i, e_i]$ is connected (e.g., according to the depth-first search algorithm). $\text{Connected}(M[e_i, e_i]) = 1$ indicates the subgraph induced by matrix $M[e_i, e_i]$ is connected; otherwise, $\text{Connected}(M[e_i, e_i]) = 0$.

Since the amount of tasks in each subzone depends on the real-time state of e-bikes and may change over time, it may not be appropriate to partition the road network in a fixed way throughout the day. However, the frequent changes to the road network partition may cause confusion to workers. Therefore, in the proposed spatiotemporal partitioning method, an initial partition solution is first generated at the beginning of the operating day and then will be adjusted dynamically in the large time scale.

4.2. Initial Partition Generation. When generating the initial road network partition, we first ignore the dynamical e-bike state and only consider the static topology of the road network to ensure that the initial partition is the same at the beginning of each day. We adopt the solution of the optimization problem (8) with constraints (5) and (6) as the initial road network partition.

$$\min_{e_1, e_2, \dots, e_{Nw}} \text{RatioCut}(g) = \frac{1}{2} \sum_{i=1}^{Nw} \frac{e_i^T M (1 - e_i)}{1^T e_i}. \quad (8)$$

The RatioCut problem in equation (8) is a classic graph-cutting problem and attempts to minimize the links between subzones while making the number of edges in each subzone as close as possible. The number of edges in a subzone is often correlated with the number of tasks and the total length of the edges in this subzone. Various algorithms to this problem have been proposed in the past and we adopt the solution based on spectral clustering algorithm, which can be referred in the literature [42, 43]. If there is a subgraph in the initial partition that is not connected, the nonmaximal connected components of this subgraph will be assigned to other adjacent subgraphs to ensure that all subgraphs in the initial partition are connected.

4.3. Partition Adjustment Algorithm. For a new time period p and an existing road network partition g of the previous time period (for period p_0 , the previous road network partition is the initial partition), we first determine whether the existing partition needs to be adjusted according to the real-time e-bike state, and if so, we adjust the existing partition through minimizing the objective in equation (4).

In the last two decades, various algorithms for community detection or graph partition have been proposed, including a large part of optimization-based methods, i.e., the modularity optimization method [44–47]. With reference to literature [46], a simulated annealing heuristic algorithm is adopted to adjust the road network partition. The simulated annealing algorithm is an effective heuristic algorithm that can effectively avoid falling into local minima and eventually converge to the global optimum. The process is shown in Algorithm 1.

In Algorithm 1, we use three operators to generate new solutions, i.e., the *task balance operator*, the *length balance operator*, and the *local search operator*. The *task balance operator* consists of the following steps as illustrated in Figure 5:

Step 1: for the existing partition g , the $\text{TaskCut}(g)$ is calculated according to equation (4), as shown in Figure 5(a).

Step 2: select two adjacent subzones randomly according to their difference in underpowered e-bike number. Here we ensure that the adjacent subzones with larger difference in underpowered e-bike number have a higher probability of being selected (e.g., through Boltzmann exploration strategy). Suppose the subzones E_2 and E_3 in Figure 5(b) were selected. As $e_2^T u_p > e_3^T u_p$, we select the edges in E_2 that are linked to the edges in E_3 as the removable edge set of E_2 , i.e., the edges (v_0, v_3) and (v_2, v_5) .

Step 3: we randomly select an edge from the removable edge set of E_2 and transfer it to E_3 , such as the edge (v_2, v_5) . It should be noted that the edge selected here cannot change the connectivity of E_2 after being transferred, i.e., satisfying the constraint in equation (7). Then, a new partition g' is created as shown in Figure 5(c).

The *length balance operator* and the *local search operator* are the same as the *task balance operator* except for step 2 above. In the *length balance operator*, we select two adjacent subzones for edge transfer randomly according to their difference in the sum of edge lengths. In the *local search operator*, we select two adjacent subzones for the edge transfer completely at random.

We define the $\text{TaskCut}_{\min}$ in Algorithm 1 as the minimum TaskCut value at the previous time period. This suggests that no change to the existing partition is needed as long as the variance of the underpowered e-bike number in each subzone at the current period is not greater than the previous time period. For period p_0 , we set $\text{TaskCut}_{\min} = 0$.

5. Lookahead Policy with MCTS Algorithm

After partitioning the road network into subzones, on the lower level of the solution framework, we model the routing decision process of each worker in their subzone as a mono-agent MDP and adopt a lookahead policy with MCTS algorithm; the lookahead policy based on the rollout heuristic is introduced in Section 5.1. A MCTS algorithm that can improve the simulation efficiency is proposed in Section 5.2. A heuristic pruning rule is added to the MCTS algorithm in Section 5.3.

5.1. Lookahead Policy. This paper takes an arc as the unit of the routing decision process as the two arcs of an edge generally cannot be served at the same time. We use $Q_k^*(s_k^m, x_k^m)$ to denote the optimal Q -value of the state-decision pair, representing the expected rewards obtained along the optimal policy after executing decision x_k^m under

Input: u_p at time period p , the edge length l , existing partition g , the initial temperature β_0 , $\text{MaxIter}^{\text{inner}}$, $\text{MaxIter}^{\text{outer}}$, $\alpha \in (0, 1)$, $\text{TaskCut}_{\min}$

Output: new road network partition g^{best}

If $\text{TaskCut}(g) \leq \text{TaskCut}_{\min}$:
 Keep the existing partition g as g^{best}

Else:
 Set temperature $\beta = \beta_0$
Repeat:
Repeat:
 Create a new partition g' from g using *task balance operator*
 Calculate $\Delta_{\text{task}} = \text{TaskCut}(g') - \text{TaskCut}(g)$
 Accept $g = g'$ with probability of $\min\{1, \exp(-\Delta_{\text{task}}/\beta)\}$
 Create a new partition g' from g using *length balance operator*
 Calculate $\Delta_{\text{task}} = \text{TaskCut}(g') - \text{TaskCut}(g)$ and accept $g = g'$ with probability of $\min\{1, \exp(-\Delta_{\text{task}}/\beta)\}$
Until iteration limit $\text{MaxIter}^{\text{inner}}$ reached
 Create a new partition g' from g using *local search operator*
 Accept $g = g'$ if $\text{TaskCut}(g') < \text{TaskCut}(g)$
 Set $\beta = \alpha\beta$
 Use g^{best} to record the best partition solution found
Until iteration limit $\text{MaxIter}^{\text{outer}}$ reached
 Output the new road network partition: g^{best}

ALGORITHM 1: Simulated annealing-based partition adjustment algorithm.

FIGURE 5: An illustration of the task balance operator. The first number on each edge denotes the number of underpowered e-bikes of the edge at time period p , and the second number on each edge denotes the length of the edge (km). Here, ϕ_1 , ϕ_2 , and ϕ_3 are set to 100, 100, and 1, respectively. (a) Existing partition. (b) Select subzones. (c) Transfer edge.

the state s_k^m . Thus, the optimal decision can be deduced by the following equation:

$$x_k^m = \operatorname{argmax}_{x \in X_{s_k^m}} Q_k^*(s_k^m, x). \quad (9)$$

However, it is difficult to obtain the optimal Q-value in our problem. The difficulty lies not only in the complexity of the problem but also in the stochastic and time-dependent user demands.

This paper adopts a lookahead policy for our problem. The rollout policy is one of the lookahead policies [18, 48] and will estimate the approximate value of a feasible decision from the simulated experience generated in a simulator. The simulator will generate the stochastic pickup and return demands, according to a priori knowledge or the prediction of the realistic environment. The rollout policy is suitable for problems where the state space is very large.

The process of the lookahead policy with rollout heuristic is shown in Figure 6. For the k th decision point of worker w_m , we first simulate from each feasible decision x for a predefined simulation run I_r and then use the Q-value calculated by equation (10) to approximate the optimal Q-value in equation (9).

$$Q_k(s_k^m, x) = \frac{1}{I_r} \sum_{j=1}^{I_r} \sum_{i=k}^{k_{\max}^j} \lambda^{i-k} \cdot r_{ij}^m \quad \forall x \in X_{s_k^m}, \quad (10)$$

where λ is a time discount factor to reduce the impact of rewards in future because the uncertainty gets higher further into the future. As there are a large number of time steps in the operating day, we truncate the rollout trajectories and set the simulation horizon to H time steps. H is set to 180 according to preliminary tests. $k_{\max}^j$ is the terminal decision point within the horizon H in the j th simulation trajectory. r_{ij}^m is the reward for the i th decision point of worker w_m in the j th simulation trajectory. For the decision point $i = k + 1, k + 2, \dots, k_{\max}^j$, the decision is made following the base heuristic polikers also follow the base heuristic policy when making decisions during the simulation.

5.2. MCTS Algorithm. The rollout policy spends the same simulation runs for each feasible decision, resulting in a lack of exploitation of the decisions that appear to be promising and much simulation for some decisions that may be clearly inferior to others [48]. Therefore, this paper adopts the MCTS sampling algorithm, a variant of the rollout policy, that can better the trade-off between exploration and exploitation when allocating simulation resources.

When we are making a routing decision, we are also attempting to find an arc belonging to the optimal route. Constructing a decision tree of all feasible decisions is equivalent to traversing all possible routes that a worker can take in future.

In the decision tree, a node represents an arc. The root node (denoted as node_1) represents the current arc where the worker is located at. The child nodes of a given node indicate the arcs that can be expanded from the given arc. All children of the root node form the feasible decision set. The nodes between the root node and a leaf node form a

possible route: $\mathcal{L} = (\text{node}_1, \text{node}_2, \dots)$. It is worth noting that in the simulation horizon, if the worker runs out of batteries, he will return to the depot immediately to reload the batteries and then go to his subzone to continue the operation. The arcs passed during this process will not be recorded in the decision tree.

On the basis of the rollout policy, the MCTS algorithm will maintain a decision tree during the simulation. Each node in the decision tree has two attributes: n_i denotes the number of times that node_i is visited and score_i denotes the value of node_i . The root node and its child nodes are initialized with $n_i = 0$, $\text{score}_i = 0$.

MCTS is an iterative algorithm, where the maximum iteration number is supposed to be set to $I_{\max}$. Each iteration contains the following four main parts: tree traversal, node expansion, random simulation, and backpropagation:

- (i) Tree traversal: tree traversal starts from the root node and selects the child nodes layer by layer until the current node is a leaf node. One criterion for selecting child node is the UCB algorithm [49]:

$$\text{UCB}(\text{node}_i) = \begin{cases} \text{score}_i + \gamma \sqrt{\frac{2 \ln N}{n_i}}, & \text{if } n_i > 0, \\ \infty, & \text{if } n_i = 0, \end{cases} \quad (11)$$

where node_i is a child node, N denotes the total visited times of the parent node, and γ is a weighting factor. The child node with the largest UCB value will be selected. The simulator will simulate the user demand and the operation of workers after a child node is selected. When the worker completes the task, a new child node will be selected until the current node is a leaf node.

- (ii) Node expansion: if the current leaf node has not been visited, then go to the next step; if the current leaf node has been visited, then expand its children to the decision tree and set the visited time and the value of these child nodes to 0. A child node is selected at random for the next step.
- (iii) Random simulation: random simulation is performed from the current leaf node until the simulation horizon H according to the base heuristic policy in Algorithm 2. Suppose the worker is w_m and the current decision point is k . Until the end of the simulation, a route $\mathcal{L}$ of w_m will be obtained. Suppose the terminal decision point in the simulation is $k_{\max}$. The $\text{Score}(\mathcal{L})$ in equation (12) is used to evaluate route $\mathcal{L}$:

$$\text{Score}(\mathcal{L}) = \sum_{i=k}^{k_{\max}} \lambda^{i-k} \cdot r_i^m. \quad (12)$$

- (iv) Backpropagation: after getting the $\text{Score}(\mathcal{L})$, the nodes of the decision tree that are also in route $\mathcal{L}$ will be updated according to equations (13) and (14):

Input: state $s_k^m = (t_k^m, \eta_k^m, \varphi_k^m, I_k^m, C_k^m)$, current road network partition g

Output: decision x_k^m and calculate the feasible decision set $X_{s_k^m}$ based on the current worker's location I_k^m and the road network partition g ;

- (1) Calculate the ratio of the underpowered e-bike number and available e-bike number $u \cdot dp(x)$ on each feasible arc $x \in X_{s_k^m}$ based on the e-bike state η_k^m, φ_k^m .
- (2) If the available e-bike number is 0, we replace it with a sufficiently small value.
- (3) Select a decision x_k^m with Boltzmann exploration $P(x_k^m | s_k^m) = (\exp(u \cdot dp(x_k^m))) / (\sum_{x \in X_{s_k^m}} \exp(u \cdot dp(x)))$.

ALGORITHM 2: Base heuristic policy

FIGURE 6: A lookahead policy with rollout heuristic.

$$n_i = n_i + 1 \quad \forall \text{node}_i \text{ in } L, \quad (13)$$

$$\text{score}_i = \frac{(n_i - 1) \cdot \text{score}_i + \text{Score}(\mathcal{L})}{n_i} \quad \forall \text{node}_i \text{ in } L. \quad (14)$$

After performing the backpropagation, the decision tree is updated. Then, it returns to the tree traversal in step (1) and continues iteration until the iteration number reaches $I_{\max}$. Finally, the child node of the root node with the highest value will be selected as the next arc to visit.

5.3. k -Cycle Pruning Rule. The rules for expanding the child nodes were introduced earlier, i.e., expanding the arcs that are adjacent to the parent node and in the same subzone. However, following this relatively loose rule may generate some apparently low-value routes, such as circular routes. Take Figure 3 for example; the circular route $(\langle v_7, v_4 \rangle, \langle v_4, v_7 \rangle, \langle v_7, v_4 \rangle, \langle v_4, v_7 \rangle, \dots)$ may yield low reward because of repeated visits to the same arcs during a short time. To avoid such situation, we propose a k -cycle pruning rule.

For $no \cdot de_i$ at the i th level of the decision tree (suppose the root node $node_1$ is the first level), we denote the route

from the root node to it by $\mathcal{L} = (\text{node}_1, \text{node}_2, \dots, \text{node}_{i-1}, \text{node}_i)$. If a child node of $node_i$ is in the set of $\{\text{node}_{i+1-\min(k,i)}, \text{node}_{i+2-\min(k,i)}, \dots, \text{node}_{i-1}, \text{node}_i\}$, this child node will not be expanded (unless there is no other expandable child node). k is an empirical constant and is tuned artificially. The routes with circles of length less than or equal to k will be pruned from the decision tree. For example, if k is set to 2, the decision tree from the arc v_7, v_4 in Figure 3 will be cut as shown in Figure 7. In the experiment, k is tuned to 6 according to preliminary tests.

6. System Simulator

Referring to literature [50], when there is no real e-bike trip data or no condition to test the proposed policy in reality, we turn our focus to simulation, which is an integral part of most studies of dynamic planning in complex systems. A system simulator is designed to capture the basic dynamics of the interactions of road network, e-bikes, workers, and users, which is shown in Figure 8.

In the “road network partition” module, an initial road network partition will be generated at the beginning of the operating day and will be adjusted every new period.

An e-bike in the EBSS has three attributes: location, battery power level, and usage state. It is assumed that users only pick up or return e-bikes in the virtual parking spots.

FIGURE 7: An example of the k -cycle pruning rule.

FIGURE 8: The simulation flowchart.

Therefore, e-bikes are only parked within the virtual parking spots or used on the road. The configuration at the beginning of the simulation includes the number of e-bikes in each virtual parking spot and the initial battery power level of each e-bike. In each time step, the attributes of the e-bike will

be changed by two factors: the worker's operation and the user's usage.

In the "worker's operation" module, the simulator will update the worker's operations one by one. Suppose a worker is operating on arc $\langle v_i, v_j \rangle$; the deadheading cost c_{ij}

is used as the expected travel time for the worker on arc $\langle v_i, v_j \rangle$. At every time step, if there is an underpowered e-bike on arc $\langle v_i, v_j \rangle$, the worker will spend τ^b minutes to replace the battery for this e-bike. Also, the battery power level of the e-bike will be updated to $\psi_{\max}$ accordingly. If there is no underpowered e-bike, the expected travel time will be reduced by one time step. If the worker's remaining travel time on the current arc becomes 0, a decision point occurs and the worker will select an adjacent arc to visit next according to the proposed policy.

In the “pickup demand” module, the simulator will generate a batch of new orders every time step. The order contains the following attributes: original virtual parking spot, destination, virtual parking spot, distance, and expected travel time.

The origins of the orders are first generated stochastically according to a certain distribution. Spatially, the virtual parking spots are aggregated based on the arc, i.e., virtual parking spots on the same arc have the same pickup demand distribution. Temporally, the pickup demand at each time step is assumed to conform to a certain Poisson distribution. Given the origin of an order, the destination is generated from a Rayleigh distribution: $f(x) = (x/\sigma^2)\exp(-(x^2/2\sigma^2))$, where x is the Manhattan distance from the destination to the origin and σ is a parameter that is set to 3 empirically. Given the origin and the destination, the shortest path between the origin and destination in the road network is calculated as the distance (e.g., calculated by the Dijkstra algorithm or Floyd algorithm). The user's speed is assumed to be generated from a normal distribution: $N(12 \text{ km/h}, 5 (\text{km/h})^2)$, empirically. Therefore, the expected travel time can be estimated and will be approximated as an integer number of time steps.

For each new order, the simulator will first try to dispatch it to an e-bike. If there is an available e-bike in the original virtual parking spot or neighboring virtual parking spots on the same arc, the dispatch will be successful and the order will be tracked by the simulator. The usage state of the dispatched e-bike will be changed to be in use; else, the dispatch is failed and the order will be deleted.

In the “return demand” module, at each time step, the simulator will subtract a time step from the expected travel time of the orders in progress. If the expected travel time of an order becomes zero, this order is finished and will be deleted from the simulator. The attributes of the e-bike dispatched to this order will be updated: the location will be changed to the destination, the battery power level (represented in range) will be subtracted by the distance of this order, and the state will be changed to be not in use.

Finally, if the time step reaches T , the simulation is ended and the simulation data will be analyzed.

7. Numerical Studies

In this section, the results of the numerical study on a test case are shown. In Section 7.1, the test case is presented; Section 7.2 introduces the benchmark policies; a comparison of the performance of different policies is

presented in Section 7.3; and more insights into the experiment are given in Sections 7.4–7.6. Some parameters of the model, as well as some tuned parameters in the algorithms, are given in Table 1.

7.1. Test Case. The simulated scenario of the test case study is shown in Figure 9, covering an area of $6 \times 6 = 36 \text{ km}^2$, with 81 vertexes and 272 arcs. Assume that vertex 41 is also the location of the depot (i.e., vertex v_0). To make the scenario as realistic as possible, we set three types of districts, indicated by red, blue, and green, respectively. Suppose the red arcs represent the central business district, the blue arcs represent the residential districts, and the green arcs represent the connective district.

Arcs in different colored districts are supposed to have different temporal trends of user demand. For example, in residential districts, the pickup demand level for e-bikes is higher in the morning rush than in the evening rush, while in central business districts, the pickup demand level for e-bikes is higher in the evening rush than in the morning rush. The distribution of user demand is assumed to be the same for the time steps during an hour. The relative pickup demand level for virtual parking spots in the three different districts during each hour is shown in Figure 10.

Given the relative pickup demand level and the total pickup demand number, the specific pickup demands can be generated for each virtual parking spot at each time step. The total pickup demand number in a day has three levels to verify the validity of the proposed policy: 2000 pickup demands/day, 4000 pickup demands/day, and 8000 pickup demands/day. Given the origin of an order, the destination, distance, and the expected travel time are generated from the rules in Section 6.

There are three lengths of arcs in the road network: 0.5 km, 1 km, and 2 km. Assume that there are 1–3 virtual parking spots randomly distributed in each arc, so the density of virtual parking spots is greater in the residential and central business districts than in the connecting districts because most arcs in the residential and central business districts are shorter than the arcs in the connecting districts. This is usually in line with the reality.

The total number of e-bikes in the system (i.e., N^{eb}) is fixed at 1000. In the initial configuration, the e-bikes are randomly distributed in each virtual parking spot and the battery level is uniformly distributed in the range $[0, \psi_{\max}]$, where $\psi_{\max}$ is set to 40 km.

Assume that there are 4 workers/vehicles and the battery capacity of each vehicle is 30. The speed of the worker when operating in his subzone is assumed to be constant at 10 km/h in the residential districts and central business districts and 12 km/h in the connective district. The speed of the worker when traveling between the depot and his subzone is assumed to be constant at 15 km/h. From this speed parameter, the deadheading cost of each arc can be calculated. The time cost to replace a battery is set to 1 minute, and the time cost to reload a battery at the depot is set to 0.5 minutes.

FIGURE 9: Road network of the test case study.

FIGURE 10: Relative pickup demand level for virtual parking spots in each district.

We simulated for 7 days to evaluate each policy. The working hours are from 7 am to 10 pm. Suppose it is possible to make the initial settings of the system generated from the same distribution for every day to test the effectiveness of the dynamic operations.

7.2. Benchmarks. To verify the effectiveness of the suggested policy, the following benchmark policies are designed.

Random policy: the random policy selects the feasible decisions with equal probability.

ϵ – greedy policy: without considering future information, ϵ – greedy policy selects an arc to visit next based on the ratio of the underpowered e-bike number and available e-bike number on each feasible arc (if the available e-bike number is 0, we replace it with a sufficiently small value). Suppose the number of feasible arcs is n . The feasible arc with the maximum ratio is selected with the probability of $(1 - \epsilon) + \epsilon/n$; any other feasible arc is selected with the probability of ϵ/n . Here ϵ is tuned to 0.3.

Rollout policy: the lookahead policy with rollout heuristic (denoted as the rollout policy) introduced in Section 5.1 is also used as a benchmark. The simulation horizons and the evaluation way of the rollout policy are consistent with the MCTS algorithm.

7.3. Results. We compare the suggested lookahead policy with MCTS algorithm (denoted as the MCTS policy) with the benchmarks at multiple demand levels.

The simulation run for each feasible decision I_r is tuned to 16 in the rollout policy. The maximum iteration I_{max} is set to the number of feasible decisions multiplied by 16 in the MCTS policy to ensure that the total simulation run at each decision point is the same for the rollout policy and the MCTS policy.

Three demand levels of 2000 pickup demand/day, 4000 pickup demands/day, and 8000 pickup demands/day are tested. The average score of each policy is calculated from the simulation trajectories by equation (1). The result is shown in Figure 11.

The result shows that the rollout policy and the MCTS policy outperformed the myopic random policy and ϵ – greedy policy significantly. Except at the demand level of 2000 pickup demands/day, the MCTS policy outperformed the rollout policy. Notably, at the demand level of 2000 pickup demands/day, the MCTS policy performed slightly worse than the rollout policy and the ϵ – greedy policy

performed slightly worse than the random policy. This phenomenon occurred because at a low demand level (with the same e-bike number and worker number), the amount of tasks for each worker is relatively low, resulting in a relatively abundant labor force for these tasks and a low sensitivity of the results to different policies under the assumption that temporary stops of vehicles are not allowed during operating.

The number of replaced batteries varies more while the successful pickup number varies less under different policies, resulting in a higher impact of the number of replaced batteries on the final score. In fact, we found that the impact on the success rate of user pickups through battery replacement operation alone is limited because the EBSS also has other problems that can affect the success of user pickups, such as the unbalanced distribution of e-bike supply and user demand, which could not be solved by battery replacement operation.

7.4. The Partitioning. For a given scenario, the road network partition way will be adjusted every time period. Take the test case with a demand level of 4000 pickup demands/day for example; the road network partition during some working hours of a day is shown in Figure 12.

In Figure 12, the edges with the same color belong to the same subzone. From the result, it could be seen that the spatiotemporal partitioning method is well able to partition the road network according to the topological characteristics and can adjust over the temporal workload characteristics. The results show that each subzone is relatively close in size and the partitioning results in each period are relatively close, which ensures that the change of the subzone does not cause too much confusion to the workers.

The average number of batteries replaced by each worker at different demand levels is shown in Table 2. In Table 2, the average number of initial underpowered e-bikes in each subzone at the beginning of the operating day is notated as “initial_underpowered,” the average number of new emerged underpowered e-bikes in each subzone during the operating day is notated as “new_underpowered,” and the average number of batteries replaced by each worker is notated as “battery_replaced.” The % completion_rate is defined as the ratio of the “battery_replaced” and the sum of the “initial_underpowered” and “new_underpowered” in the following equation:

$$\% \text{ completion_rate} = \frac{\text{battery_replaced}}{\text{initial_underpowered} + \text{new_underpowered}} * 100 \%. \quad (15)$$

As the number of user demands increases, the number of batteries replaced by each worker also increases, but the % completion_rate of each worker decreases. For a given demand level, the number of batteries replaced by each worker is relatively close, which proves the effectiveness of the partitioning algorithm. From the results, we can see that the batteries replaced by workers w_2 and

w_4 are relatively more than the workers w_1 and w_3 . In fact, this is because in the subzone where the road network is sparse and the e-bikes are more sparsely distributed (such as the subzone of w_1 and w_3 in Figure 12), the worker has to spend more time running on the road, resulting in a lower number of batteries replaced in the same operating time.

(a)

(b)

(c)

FIGURE 11: Evaluation of different policies at three demand levels. (a) Successful pickup number. (b) Number of replaced batteries. (c) Score.

FIGURE 12: An example of the road network partition result. The blue, yellow, red, and green arcs denote the subzone of workers w_1 , w_2 , w_3 , and w_4 , respectively. (a) 8 am. (b) 12 am. (c) 6 pm. (d) 10 pm.

From the results, we also can see that at each demand level, the % completion_rate of different workers is relatively close, which reflects that the partition is reasonable and allows the labor force to be appropriately distributed. Also, we can find that the worker number is sufficient at the demand level of 2000 pickup demands/day and the demand level of 4000 pickup demands/day because the batteries replaced are more than the new appeared underpowered batteries, but more workers are needed at the demand level of 8000 pickup demands/day as the batteries replaced by

each worker are all less than the new appeared under-powered batteries.

7.5. Dynamic and Static Operations. To verify the necessity of dynamic operation, we compared the lost user number under dynamic battery replacement operation and under static battery replacement operation. In the static case, no operation is applied to the e-bikes during the daytime, but the initial setting of a day is set to the same as the dynamic

TABLE 2: Results of the batteries replaced by each worker.

Worker	initial_underpowered	new_underpowered	battery_replaced	% completion_rate
(a) 2000 pickup demands/day				
w_1	13.8	40.6	47.4	87.1
w_2	25.6	45.4	62.4	87.9
w_3	13.4	34.8	41.8	86.7
w_4	20.0	49.4	59.2	85.3
(b) 4000 pickup demands/day				
w_1	14.8	86.8	83.6	82.3
w_2	25.2	94.6	97.2	81.1
w_3	14.2	63.8	63.8	81.8
w_4	22.6	98.4	94.4	78.0
(c) 8000 pickup demands/day				
w_1	14.0	144.2	120.0	75.9
w_2	29.0	168.8	145.0	73.3
w_3	14.0	127.0	106.0	75.2
w_4	20.4	195.6	155.4	71.9

case under the assumption that there is a static battery replacement operation at night.

The lost users are divided into two categories: lost users because there is no e-bike in the virtual parking spots, denoted as “unbalance lost,” and lost users because the e-bikes in the virtual parking spots are underpowered, denoted as “underpowered lost.”

The average number of lost users under static and dynamic operations at three demand levels is shown in Table 3. From the results, we can see that the dynamic operation could reduce the total lost users compared to the static case. In addition, the result shows that dynamic battery replacement mainly reduced the number of the “underpowered lost” users but may exacerbate the unbalance of the e-bike supply, resulting in more “unbalance lost” users. One reason for the high “unbalance lost” number in the dynamic operation case is that for an arc with underpowered e-bikes, in the static operation case, the subsequent failed pickups will be recorded as “underpowered lost”; however, in the dynamic operation case, the underpowered e-bikes may be replaced battery by the worker in time and will be picked up by users who arrive first, and the subsequent failed pickups on this arc will be recorded as “unbalance lost.” Therefore, both the battery replacement operation and rebalancing operation are essential for the EBSS.

The average number of lost users reduced by the dynamic operation (compared with the static operation) at three demand levels is 87.8, 314.0, and 1131.8, respectively, which suggests that at the low demand level, such as 2000 pickup demands/day, it may not be cost effective to perform dynamic operations or the number of workers can be reduced appropriately due to the low ratio of user demand over e-bike supply.

7.6. Simulation Runs. In the MCTS policy and rollout policy, the number of simulation runs is a key factor for estimating the value of feasible decisions. In general, the more the simulation runs, the more accurate the estimation, but it also requires more computational cost.

As the MCTS policy is able to allocate more simulation runs to more highly rewarding routes, it requires less

TABLE 3: Number of lost users under dynamic and static operation.

Demand level	Lost unbalanced	Lost underpowered	Lost total
(a) Static			
2000	203.4	233.8	437.2
4000	536.8	842.4	1379.2
8000	1117.8	3037.0	4154.8
(b) Dynamic (MCTS policy)			
2000	266.0	83.4	349.4
4000	759.6	305.6	1065.2
8000	1878.6	1144.4	3023.0

TABLE 4: Results under different simulation runs.

Simulation runs/decision	Pickup	Replace	Score
(a) Rollout			
4	4958.8	519.0	32839.7
8	4958.2	521.2	32919.4
16	4948.6	527.2	33115.0
32	4962.8	532.4	33346.5
(b) MCTS			
4	4961.4	512.8	32617.4
8	4969.2	526.4	33142.1
16	4972.6	530.0	33347.4
32	5021.0	541.2	33832.2

computational costs to achieve the same effect as the rollout policy. We compared the results of these two policies for different number of simulation runs at the demand level of 8000 pickup demand/day, as shown in Table 4.

Four numbers of simulation runs are tested: 4, 8, 16, and 32 simulation runs per feasible decision. Take the 8 simulation runs for example; if the number of feasible decisions of a decision point is 3, then the total simulation runs at that decision point is 24. The result shows that the more the simulation runs, the higher the final score in both the rollout policy and the MCTS policy. The results of experiments with 8, 16, and 32 simulation runs all show that the MCTS policy performs better than the rollout policy, indicating that the MCTS policy requires only half the simulation runs to obtain the same or even better results than the rollout policy. However, in the experiment with 4

simulation runs, the MCTS policy performs worse than the rollout policy. One reason for this result is that the simulation runs for some feasible decisions may be so inadequate as to cause great deviation in the MCTS policy when the total number of simulations runs is relatively low.

8. Conclusion

In this paper, we have modeled the battery replacement in an EBSS as a DCARP according to the distribution characteristics of the virtual parking spots. The objective is to maximize a weighted sum of the number of successful pickup demands and the number of batteries replaced. A “Partition-First, Route-Second” solution framework was proposed for the scenario with multiple workers. A spatiotemporal partitioning method was proposed to partition the road network into subzones on a large time scale, and the routing decision process of each worker was modeled as MDP on a small time scale. A lookahead policy with MCTS sampling algorithm was proposed to anticipate the potential future influence of the current decisions. The numerical study results show that the spatiotemporal partitioning method was able to avoid the conflict between workers’ operations and allocate the labor force reasonably. The MCTS policy proposed in this paper outperformed the myopic policies and could reduce the simulation runs significantly with no degradation in effectiveness compared to the rollout policy.

The limitations of the application scenarios of the suggested policy also need to be noted. Like the rollout policy and other lookahead policies, the MCTS policy relies on the forecast of future exogenous information, such as the user demand trend. As analyzed in literature [25], if the impact of noise of user demand becomes too significant, the anticipation for future demand is less possible, and a policy only based on the current state may yield better results than the lookahead policy.

Future research for the EBSS may focus on the case where battery replacement operation and the rebalancing operation are performed simultaneously in the system. The unbalance between supply and demand in bike sharing systems has been studied extensively and also exists in the EBSS. Our experiments show that only replacing the battery dynamically may have limited improvement in the success rate of user pickup. As the e-bike has not only location attributes but also battery power level attributes, the rebalancing problem of an EBSS may be more complicated and needs further study.

Data Availability

The data used to support the findings of this study are available at https://drive.google.com/file/d/12Dl_Y1HkTVHl3FNoRQgr3S2lKtYnvQOe/view?usp=sharing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China (71772195) and the Shenzhen Municipal Science and Technology Innovation Committee (JCYJ20170412172030008 and JCYJ20170818092931604).

References

- [1] H. A. Eiselt, M. Gendreau, and G. Laporte, “Arc routing problems, part I: the Chinese postman problem,” *Operations Research*, vol. 43, no. 2, pp. 231–242, 1995.
- [2] V. Pillac, M. Gendreau, C. Guret, and A. L. Medaglia, “A review of dynamic vehicle routing problems,” *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.
- [3] H. N. Psaraftis, M. Wen, and C. A. Kontovas, “Dynamic vehicle routing problems: three decades and counting,” *Networks*, vol. 67, no. 1, pp. 3–31, 2016.
- [4] P. Lacomme, C. Prins, and W. Ramdane-Chrif, “Evolutionary algorithms for multiperiod arc routing problems,” in *Proceedings of the 9th Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Annecy, France, 2002.
- [5] F. Chu, N. Labadi, and C. Prins, “A scatter search for the periodic capacitated arc routing problem,” *European Journal of Operational Research*, vol. 169, no. 2, pp. 586–605, 2006.
- [6] J.-P. Riquelme-Rodrguez, A. Langevin, and M. Gamache, “Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints,” *Networks*, vol. 64, no. 2, pp. 125–139, 2014.
- [7] G. V. Batista, “A new ant colony optimization algorithm to solve the periodic capacitated arc routing problem with continuous moves,” *Mathematical Problems in Engineering*, p. 2019, 2019.
- [8] C. H. Christiansen, J. Lysgaard, and S. Whlk, “A branch-and-price algorithm for the capacitated arc routing problem with stochastic demands,” *Operations Research Letters*, vol. 37, no. 6, pp. 392–398, 2009.
- [9] G. Laporte, R. Musmanno, and F. Vocaturo, “An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands,” *Transportation Science*, vol. 44, no. 1, pp. 125–135, 2010.
- [10] M. Tagmouti, M. Gendreau, and J.-Y. Potvin, “A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs,” *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 954–963, 2010.
- [11] M. Tagmouti, M. Gendreau, and J.-Y. Potvin, “A dynamic capacitated arc routing problem with time-dependent service costs,” *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 1, pp. 20–28, 2011.
- [12] Y. Mei, K. Tang, and X. Yao, “Evolutionary computation for dynamic capacitated arc routing problem,” *Evolutionary Computation for Dynamic Optimization Problems, Studies in Computational Intelligence*, Springer, Berlin, Germany, pp. 377–401, 2013.
- [13] E. Benavent, “The capacitated arc routing problem,” *A Heuristic Algorithm*, 1990.
- [14] A. Butsch, J. Kalcsics, and G. Laporte, “Districting for arc routing,” *INFORMS Journal on Computing*, vol. 26, no. 4, pp. 809–824, 2014.
- [15] Y. Zhang, “Divide-and-conquer large scale capacitated arc routing problems with route cutting off decomposition,” *Information Sciences*, vol. 553, no. 2021, pp. 208–224.

- [16] T. Hintsch, S. Irnich, and L. Küllerich, "Branch-price-and-cut for the soft-clustered capacitated arc-routing problem," *Transportation Science*, vol. 55, 2021.
- [17] Y. Zhang, "A route clustering and search heuristic for large-scale multidepot-capacitated arc routing problem," *IEEE Transactions on Cybernetics*, 2021.
- [18] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Vol. 703, John Wiley & Sons, Hoboken, NJ, USA, 2007.
- [19] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, USA, 2018.
- [20] E. Macioszek and A. Kurek, "P&R parking and bike-sharing system as solutions supporting transport accessibility of the city," *Transport Problems*, vol. 15, 2020.
- [21] S. A. Shaheen, S. Guzman, and H. Zhang, "Bikesharing in Europe, the americas, and asia," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2143, no. 1, pp. 159–167, 2010.
- [22] S. Sohrabi, R. Paleti, L. Balan, and M. Cetin, "Real-time prediction of public bike sharing system demand using generalized extreme value count model," *Transportation Research Part A: Policy and Practice*, vol. 133, pp. 325–336, 2020.
- [23] J. Brinkmann, M. W. Ulmer, and D. C. Mattfeld, "Short-term strategies for stochastic inventory routing in bike sharing systems," *Transportation Research Procedia*, vol. 10, pp. 364–373, 2015.
- [24] S. Ghosh, P. Varakantham, Y. Adulyasak, and P. Jaillet, "Dynamic repositioning to reduce lost demand in bike sharing systems," *Journal of Artificial Intelligence Research*, vol. 58, pp. 387–430, 2017.
- [25] J. Brinkmann, M. W. Ulmer, and D. C. Mattfeld, "Dynamic lookahead policies for stochastic-dynamic inventory routing in bike sharing systems," *Computers & Operations Research*, vol. 106, pp. 260–279, 2019.
- [26] 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*, London, UK, August 2018.
- [27] E. Macioszek, P. Świerk, and A. Kurek, "The bike-sharing system as an element of enhancing sustainable mobility-A case study based on a city in Poland," *Sustainability*, vol. 12, no. 8, p. 3285, 2020.
- [28] S. Chang, "Innovative bike-sharing in China: solving faulty bike-sharing recycling problem," *Journal of Advanced Transportation*, vol. 2018, Article ID 4941029, 10 pages, 2018.
- [29] T. Guo, "Emerging technologies and methods in shared mobility systems layout optimization of campus bike-sharing parking spots," *Journal of Advanced Transportation*, vol. 2020, Article ID 8894119, 10 pages, 2020.
- [30] W.-L. Pearn, A. Assad, B. L. Golden, and Golden, "Transforming arc routing into node routing problems," *Computers & Operations Research*, vol. 14, no. 4, pp. 285–288, 1987.
- [31] H. Longo, M. P. De Aragao, and E. Uchoa, "Solving capacitated arc routing problems using a transformation to the CVRP," *Computers & Operations Research*, vol. 33, no. 6, pp. 1823–1837, 2006.
- [32] R. Hirabayashi, Y. Saruwatari, and N. Nishida, "Tour construction algorithm for the capacitated arc routing-problems," *Asia Pacific Journal of Operational Research*, vol. 9, no. 2, pp. 155–175, 1992.
- [33] M. Kiuchi, "An exact algorithm for the capacitated arc routing problem using parallel branch and bound method," in *Proceedings of the Spring National Conference of the Operational Research Society of Japan*, Tokyo, Japan, 1995.
- [34] A. A. Assad, W.-L. Pearn, and B. L. Golden, "The capacitated Chinese postman problem: lower bounds and solvable cases," *American Journal of Mathematical and Management Sciences*, vol. 7, no. 1-2, pp. 63–88, 1987.
- [35] J. M. Belenguer and E. Benavent, "A cutting plane algorithm for the capacitated arc routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 705–728, 2003.
- [36] S. Wöhlk, "New lower bound for the capacitated arc routing problem," *Computers & Operations Research*, vol. 33, no. 12, pp. 3458–3472, 2006.
- [37] E. Bartolini, J.-F. Cordeau, and G. Laporte, "Improved lower bounds and exact algorithm for the capacitated arc routing problem," *Mathematical Programming*, vol. 137, pp. 409–452, 2013.
- [38] W. L. Pearn, "Approximate solutions for the capacitated arc routing problem," *Computers & Operations Research*, vol. 16, no. 6, pp. 589–600, 1989.
- [39] W. L. Pearn, "Augment-insert algorithms for the capacitated arc routing problem," *Computers & Operations Research*, vol. 18, no. 2, pp. 189–198, 1991.
- [40] A. Hertz, G. Laporte, and M. Mittaz, "A tabu search heuristic for the capacitated arc routing problem," *Operations Research*, vol. 48, no. 1, pp. 129–135, 2000.
- [41] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "A genetic algorithm for the capacitated arc routing problem and its extensions," in *Proceedings of the Workshops on Applications of Evolutionary Computation*, 2001.
- [42] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992.
- [43] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral k-way ratio-cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1096, 1994.
- [44] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, vol. 69, no. 6, Article ID 066133, 2004.
- [45] R. Guimerà, M. Sales-Pardo, and L. A. Amaral, "Modularity from fluctuations in random graphs and complex networks," *Physical review. E, Statistical, Nonlinear, And soft Matter Physics*, vol. 70, no. 2, Article ID 025101, 2004.
- [46] A. Medus, G. Acuña, and C. O. Dorso, "Detection of community structures in networks via global optimization," *Physica A: Statistical Mechanics and Its Applications*, vol. 358, no. 2–4, pp. 593–604, 2005.
- [47] J. Duch and A. Arenas, "Community detection in complex networks using extremal optimization," *Physical Review. E, Statistical, Nonlinear, and soft Matter Physics*, vol. 72, no. 2, Article ID 027104, 2005.
- [48] D. P. Bertsekas, *Reinforcement Learning and Optimal Control*, Athena Scientific, Belmont, MA, USA, 2019.
- [49] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Lecture Notes in Computer Science*, pp. 282–293, Springer, Berlin, Germany, 2006.
- [50] L. Caggiani, R. Camporeale, M. Ottomanelli, and W. Y. Szeto, "A modeling framework for the dynamic management of free-floating bike-sharing systems," *Transportation Research Part C: Emerging Technologies*, vol. 87, pp. 159–182, 2018.