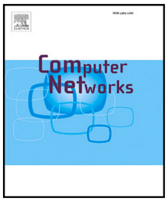
Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Congestion-aware Stackelberg pricing game in urban Internet-of-Things networks: A case study[☆]

Jiahui Jin [a,*], Zhendong Guo [b], Wenchao Bai [a], Biwei Wu [a], Xiang Liu [a], Weiwei Wu [a]

[a] *School of Computer Science and engineering, Southeast University, Nanjing, China*
[b] *College of Software Engineering, Southeast University, Nanjing, China*

ARTICLE INFO

Keywords:
Urban IoT network
Stackelberg pricing game
MPEC

ABSTRACT

We consider a one-leader multi-follower Stackelberg pricing game in urban Internet-of-Things networks. The leader manages multiple services and sets service prices to maximize profit, while followers seek the most cost-effective services, taking into account service prices, service congestion, and individual locations. Existing approaches often simplify the game process by neglecting followers' locations, but they are not applicable to real-world complex spatial distributions. In this paper, we focus on the congestion-aware Stackelberg pricing game in urban Internet-of-Things networks, using electric vehicle charging as a case study. The game is reformulated as a mathematical program with equilibrium constraints (MPEC) that considers both congestion effects and followers' spatial distributions. To solve the MPEC, we introduce the Segmentation-based Pricing with ITERative Optimization (SPITER) algorithm, which converges to a local maximum. Additionally, optimization techniques are developed to improve the performance of solving SPITER in urban environments. We evaluate the performance of SPITER using extensive experiments with two real-life urban datasets, demonstrating the advantages of our model and illustrating SPITER's effectiveness and convergence.

1. Introduction

We consider a one-leader multi-follower Stackelberg pricing game [1] that incorporates congestion effects and geolocations. In this game, the leader manages multiple services and sets service prices to maximize profit, while followers seek the most cost-effective services, taking into account service prices, service congestion, and individual locations. Such games find various applications in urban Internet-of-Things (IoT) networks, including spectrum leasing [2–8], cloud/fog/edge computing [9–15], and intelligent transportation [16–18]. Despite the significant progress made in previous research on addressing the Stackelberg pricing game [10,19–21], notable challenges persist when followers consider not only the price, but also how crowded nearby services are and their personal locations when making their choices.

To investigate the congestion-aware Stackelberg pricing game, we focus on electric vehicle (EV) charging, an urban IoT application, as a case study [22,23]. Fig. 1 shows an EV charging network, where the CS manager (leader) provides charging stations (CSs) to serve the EV users (followers). Nevertheless, the distribution of electric vehicles (EVs) and charging stations (CSs) in urban areas is not uniform, resulting in service congestion. This congestion occurs because EV users tend to prefer nearby CSs, causing congestion in urban regions with fewer CSs and degrading the charging experience. Conversely, regions with an abundance of CSs experience idle charging stations and reduced profit. In this situation, the objective of the CS manager is to maximize profit, while rational EV users aim to enhance their charging experience. Consequently, the leader (CS manager) and followers (EV users) have distinct objective functions. Therefore, in this paper, we establish a game model to determine CS pricing strategies aimed at enhancing both the charging experience and overall profit.

Nevertheless, optimizing profit is complex due to the interdependence between EV charging and CS pricing decisions. In our case study, the CS manager first sets the price for each CS, and then the EV users decide which CS to select based on the leader's price. Moreover, since EV users are rational, their decisions are motivated to maximize their own payoff (*i.e.,* charging experience), considering not only the price but also the service congestion and distance. Due to the congestion

[☆] This work is supported by National Natural Science Foundation of China under Grants No. 62072099 and No. 62232004, Natural Science Foundation of Jiangsu Province of China under Grant No. BK20230024, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No. BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No. 93K-9.

[*] Corresponding author.

E-mail addresses: jjin@seu.edu.cn (J. Jin), guozhendong@seu.edu.cn (Z. Guo), wbai@seu.edu.cn (W. Bai), beilwu@seu.edu.cn (B. Wu), xiangliu@seu.edu.cn (X. Liu), weiweiwu@seu.edu.cn (W. Wu).

https://doi.org/10.1016/j.comnet.2024.110405
Received 10 November 2023; Received in revised form 17 March 2024; Accepted 8 April 2024
Available online 10 April 2024

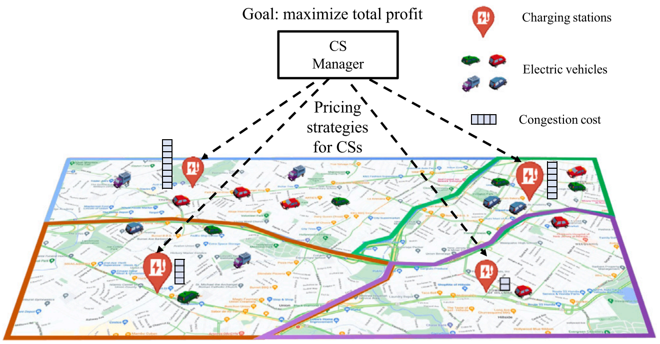


Fig. 1. Congestion-aware Stackelberg pricing game in urban Internet-of-Things networks: a case study of EV charging.

effects, the EV users' decisions are intertwined, introducing a non-cooperative game among them. Subsequently, the EV users' decisions affect the usage of each CS, and then the CS manager will set the CS price again to increase her profit. Overall, the interaction between the leader and the followers forms a two-level Stackelberg pricing game, with the upper level being the profit maximization problem of the CS manager, and the lower level being the EV users' non-cooperative game.

Solving the congestion-aware Stackelberg pricing game presents challenges for the following reasons: (i) In this game, the leader optimizes a quadratic objective function regarding the profit from each follower and numbers of followers in each CS, while the followers have individual optimization problems. This game involves a bilevel-optimization form [24], where the followers' optimization problems act as constraints within the leader's problem. Bilevel optimization is computationally expensive and NP-hard [18], making it difficult to find optimal solutions. Moreover, even evaluating solution optimality of a bilevel optimization problem is also an NP-hard task [25]. Current research relaxes constraints [23,26–29] or approximates them into convex problems [25,30–32], but such approaches do not guarantee the optimality, limiting their effectiveness. (ii) Incorporating congestion effects and spatial distance adds complexity. A follower usually prefers to the nearby CSs and his decision directly affects congestion, which impacts others' decisions due to the congestion effects. The rationality of followers creates a game-like relationship among the followers. Existing approaches simplify the game process by considering macroscopic equilibrium states like Stochastic User Equilibrium [33] or Wardrop Equilibrium [24]. However, these simplifications neglect the individual preferences of followers in the congestion-aware Stackelberg pricing game. For instance, in the EV charging market, it is crucial to consider the geographic distribution of EVs when determining pricing strategies.

In this paper, we address these issues by using a mathematical program with equilibrium constraints (MPEC) to model the congestion-aware Stackelberg pricing game. The constraints in our model stem from the equilibrium of the non-cooperative EV-EV game, which encompasses EV users' spatial distributions and congestion effects. To maximize the overall profit, we propose a (local) optimal algorithm called Segmentation-Based Pricing with Iterative Optimization (SPITER). Our approach begins by defining a subproblem that focuses on determining the optimal pricing strategy for a single CS, while holding the pricing strategies of other CSs constant. We employ a segmentation-based optimal method to solve this subproblem. This method partitions the pricing space into a series of subintervals and identifies the best price for each subinterval. By considering the entire range of pricing strategy, our approach ensures that the optimal solution is obtained for each subproblem. Through iterative resolution of these subproblems, SPITER converges toward a local maximum point of the original problem. Furthermore, we propose optimization techniques to enhance the computing performance of SPITER.

Our contributions are as follows.

1. (*Game model*) We propose a congestion-aware Stackelberg pricing game model that jointly consider the leader–follower and follower–follower interactions. Specially, we consider the followers' spatial distribution in urban environment. The existence and uniqueness of followers' equilibrium are analyzed. We also show that the problem is non-convex since it is constrained by the followers' equilibrium.
2. (*Effective algorithm*) We propose SPITER to solve the problem by using a novel segmentation method. Although the problem is non-convex, SPITER iteratively computes the optimal pricing strategy for each leader by fixing other leaders' strategies, and find a local maximum point of the pricing problem. We also prove that SPITER can converge to a local maximum point of the problem.
3. (*Optimization techniques*) We enhance the computing performance of SPITER by implementing optimization techniques to reduce the running time. These techniques involve fast initialization and distributed evaluation methods. By employing these approaches, we can expedite the computation process, thereby improving the overall computing performance of SPITER in urban environments.
4. (*Extensive evaluations*) We evaluate SPITER's performance with a case study on EV charging network using two real-life urban datasets. SPITER is compared with two pricing algorithms in terms of the CS manager's total profit under different experimental settings. Evaluation results show the advantages of our proposed model and illustrate SPITER's effectiveness and convergence.

The remaining of this paper is organized as follows. Related work is reviewed in Section 2. In Section 3, the system model and problem formulation are introduced. In Section 4, the Segmentation-Based Pricing with Iterative Optimization (SPITER) algorithm for the CSs' cooperative strategy is presented. In Section 5, an initial pricing strategy and dynamic scenario are presented. We present the simulation results and related analysis in Section 6, and conclude the work in Section 7.

2. Related work

2.1. Stackelberg pricing game with congestions

Stackelberg pricing games have been extensively studied in various computer network fields, including network communication [2–8,34], intelligent transportation [20], and cloud/fog/edge computing [9–15]. In these games, leaders aim to maximize their utility by setting prices for their resources, while followers pay for the resources they use. In real-world situations, limited resources often lead to congestion effects [35]. Thus, followers in such games consider not only the price but also the decisions of others. Since handling congestion effects can be challenging, existing approaches simplify the game by focusing on macroscope equilibrium states [20,24,36,37] like stochastic user equilibrium and the Wardrop equilibrium.

For example, Yuan et al. [20] studied the optimal charging strategy of EVs on a single road with congestion effects. EVs in the model follow a uniform distribution on the road. Boateng et al. [34] considered Stackelberg game in their problem. Their model involves resource trading among followers, rendering a three-stage Stackelberg game. Harks et al. [24] posited that the competition between followers results in the same costs of followers, and the Wardrop equilibrium can be reached. However, they overlooked the preferences of followers, that might preclude the realization of the Wardrop equilibrium. Nguyen et al. [36] considered the elastic demand of followers in their model. They assume that demand can be adjusted by the price of resource on large scale. The congestion between followers is not considered in the model.

However, these simplifications overlook the individual preferences of followers in congestion-aware Stackelberg pricing games. In our model, we considered the different preferences of followers that can be aspects like the resource demand in network communication, the weights of different costs in cloud/fog/edge computing, or the distance between CS and EV in the EV charging problem.

2.2. Solutions to Stackelberg game

There are several common methods for solving Stackelberg games, such as heuristic algorithms [19], backward induction [20,21], and KKT conditions [10]. When considering follower interactions, one typical approach is to include the lower-level equilibrium as constraints in the higher-level optimization. This transforms the Stackelberg game into a mathematical program with equilibrium constraints (MPEC). Techniques for solving MPEC problems involve approximation algorithms [25,30–32], decomposed methods [10,38], the Wardrop equilibrium [24], or changing constraints into convex or linear forms [23, 26–29].

For instance, Bohnlein et al. [25] utilized approximation algorithms to ascertain the minimal profit of leaders who set a uniform price for all resources. Wu et al. [27] introduced auxiliary integer variables to linearize the constraints of MPEC and subsequently solved the problem using a commercial software package. Chen et al. [10] employed a decomposition approach to partition their primary problem into several homogeneous sub-problems, and they used an iteration algorithm to achieve the equilibrium of each sub-problem. Luo et al. [38] also introduced an algorithm based on the decomposition method for MPEC problems. Different from [10], they divided the domain into several sub-domains, and solved the problem on each sub-domain.

However, these methods neglect the congestion of followers, and their MPEC formulations are also different from ours. Hence, their solutions cannot be utilized in our problem. To capture the congestion in the game, we do not use the methods such as the Wardrop equilibrium. Instead, we have introduced a segmentation-based method capable of achieving a local optimum.

2.3. Stackelberg game in EV charging

The Stackelberg game has been widely studied in the EV charging problem by a number of works [19,39–42]. In the EV charging problem, CSs set the price of charging first and then EVs make proper decisions according to the price. In EV charging problem, studies have discussed several factors of EV and CS such as travel cost [40], distance to a CS [40] and different charging methods [42]. However, many of these papers overlook the congestion effects in CSs caused by charging speed and their proposed EV distributions do not align with the requirements of specific settings, like urban and rural areas.

For example, Ma et al. [19] considered the effect of regional hotspot information in the charging decision-making process. They believe that hotspot information will affect the income of electric taxis. However, they overlook that the hotspot information also causes the congestion effects in CSs. Laha et al. [40] considered the time and energy consumption of travel in their model. However, they neglect congestion during charging. While some works mentioned congestion effects in their model, they simplified the problem to lower the level of complexity. Aujla et al. [41] considered a peak time in the EV charging problem. The peak time is determined based on the current energy consumption of EVs, but the number of EVs in CSs is not considered. Malandrino et al. [42] assumed that EVs are charged by replacing a fully-charged battery and applied congestion-averse utilities. Congestion is seldom considered in their charging model. However, the EV charging method in their model has not yet been realized in real life.

Using the congestion-aware Stackelberg pricing game, we conducted a case study on profit maximization in EV charging market. In our model, we considered the congestion effects in EV charging process and divided an area into different regions with different numbers of EVs to describe the EV distribution. The congestion-aware Stackelberg pricing game we studied is applied to this problem and the evaluation results show that our model and algorithm can achieve high profits for the CS charging market.

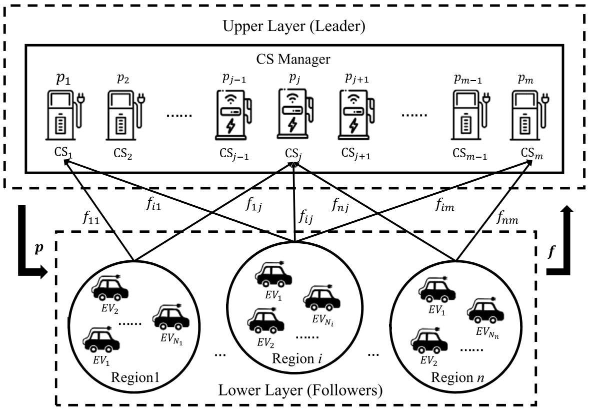


Fig. 2. Problem Model.

3. Preliminaries

3.1. Problem model

We divide the city to be analyzed into n regions in set $\mathcal{R} = \{1, 2, \dots, n\}$ according to the geographic and residential condition [43, 44]. Region i ($i \in \mathcal{R}$) contains N_i EVs to be charged. The city also contains m charging zones, each of which builds a charging station. For simplification, we use the center of a zone (region) to represent the zone's (region's) position. Let C be the set of charging zones, such that $C = \{1, 2, \dots, m\}$. We call the CS in zone j as CS j ($j \in C$), and denote its pricing strategy as p_j and its operating cost as ϵ_j. The average spatial distance from Region i to CS j is denoted as d_{ij}. To handle the citywide environment, we consider EV flows instead of individual EVs, because the market focuses more on the macro results of all the EVs' decisions in each region. We denote the EV flow sent from Region i to CS j as f_{ij} ($0 \le f_{ij} \le N_i$ and $\sum_{j=1}^{m} f_{ij} = N_i$), which is the (expected) number of EVs sent from Region i to CS j. A region determines the assignments of EV flows to generate its charging strategy. Table 1 shows frequently used notations. We then define EV flows' charging costs and the CS manager's profit (see Fig. 2).

EV Flows' Charging Costs. The charging cost of an EV flow C_{ij} considers CS j's price p_j, distance d_{ij}, and queue costs Q_j. Summing up the costs of all the flows, we define the charging cost function of EV flows in Region i as

$$C_i = \sum_{j=1}^{m} C_{ij} = \sum_{j=1}^{m} (\omega_p p_j + \omega_q Q_j + \omega_d d_{ij}) f_{ij}, \tag{1}$$

where a flow unit's cost is computed by $\omega_p p_j + \omega_q Q_j + \omega_d d_{ij}$. We measure the queuing cost and distance cost using monetary values for the waiting time and travel time [45]. Using these monetary values, the price cost, queuing cost, and distance cost can be combined using the weights ω_p, ω_q, and ω_d.

Congestion Effects. The queuing cost measures congestion effects, which the depends on the CS's capacity and number of EVs that come to the CS. We use a linear function to model the congestion effects [23,24,43]. Specifically, the workload of CS j is the total EV flows coming from all the regions, denoted by $\hat{f_j} = \sum_{i=1}^{n} f_{ij}$, and the capacity of CS j is c_j, so the queuing cost Q_j of CS j is as follows.

$$Q_j = \frac{\hat{f_j}}{c_j} \tag{2}$$

Table 1
Summary of notations.

Symbol	Description
$\mathcal{R}$	Set of regions
$\mathcal{C}$	Set of CSs
n	Number of regions
m	Number of CSs
N_i	Number of EVs in region i
c_j	Charging capacity of CS j
p_j	Charging price of CS j
p^{max}	Price ceiling of each CS
$\boldsymbol{p}$	Price vector of all CSs
d_{ij}	Distance from region i to CS j
ε_j	Operating cost of CS j
Q_j	Queuing cost of EV flow at CS j, *i.e.,* congestion
ω_p, ω_q, ω_d	Weights for p_j, Q_j, and d_{ij}
f_{ij}	EV flow from region i to CS j
$\widehat{f}_j$	Total EV flows coming to CS j from all regions
$\widetilde{\boldsymbol{f}}_i$	Charging strategy of EV flows in region i
$\widetilde{\boldsymbol{f}}_{-i}$	Charging strategies excluding EV flows in region i
$\mathcal{F}_i$	Set of all feasible strategies of EV flows in region i
$\widetilde{\boldsymbol{f}}^*$	The lower-layer equilibrium solution
C_{ij}	Charging costs of EV flow from Region i to CS j
C_i	Cost function of EV flows in region i
V_j	Profit function of CS j
V	Profit function of the CS manager
V_s	Profit function of the Subproblem
V_s^k	The kth subproblem segment
S_l	Set containing all $f_{ij} > 0$
S_e	Set containing all $f_{ij} = 0$

CS Market's Profit. The CS market's profit is a summation of all the CSs' profits, so the total profit function is defined as

$$V = \sum_{j=1}^{m} V_j = \sum_{j=1}^{m} (p_j - \varepsilon_j)\widehat{f}_j = \sum_{j=1}^{m} \left((p_j - \varepsilon_j) \sum_{i=1}^{n} f_{ij} \right), \tag{3}$$

where $V_j = (p_j - \varepsilon_j)\widehat{f}_j$ is the profit function of CS j.

According to Eqs. (1) and (3), the charging cost and market profit functions are highly coupled. The pricing strategies affect the assignments of EV flows, while the EV flows affect the queuing costs and total profit. This forms complex interactions among CSs and EV flows.

3.2. Problem formulation

We adopt the hierarchical Stackelberg game to model complex interactions, where the CS manager is the leader at the upper layer and the EV flows are followers at the lower-layer. In the hierarchical game, the CS manager takes actions first by adjusting the CSs' charging prices while considering interactions among CSs and EV flows. Upon receipt of all CSs' prices, EV flows determine their charging strategies. Following these discussions, we show the optimization problem for each participant in the hierarchical game.

3.2.1. Leader's optimization problem (upper layer)

The CS manager's goal is to determine a pricing strategy that maximizes the market profit. According to Eq. (3), the CS manager's optimization problem is as follows:

$$\max_{\boldsymbol{p}} \quad V(\boldsymbol{p}) \tag{4}$$

$$s.t. \quad \varepsilon_j < p_j \le p^{max}, \tag{4a}$$

where $\boldsymbol{p} = (p_1, p_2, \dots, p_m)$ is the optimization variable and p^{max} is the price ceiling of the market. The charging price p_j should be larger than the operating cost ε_j and no larger than the price ceiling, *i.e.,* $\varepsilon_j < p_j \le p^{max}$.

3.2.2. Follower 's optimization problem (lower layer)

The goal of Region i is to compute a charging strategy that minimizes its EV flows' charging costs. Given the pricing strategy p and other regions' strategies, Region i's optimization problem is to maximize Eq. (1).

$$\min_{\widetilde{\boldsymbol{f}}_i} \quad C_i\left(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p}\right), \tag{5}$$

$$s.t. \quad \sum_{j=1}^{m} f_{ij} = N_i, \tag{5a}$$

$$f_{ij} \ge 0, \forall j \in \mathcal{C}, \tag{5b}$$

where $\widetilde{\boldsymbol{f}}_i = [f_{i1}, f_{i2}, \dots, f_{im}]$ is the charging strategies of Region i's EV flows and $\widetilde{\boldsymbol{f}}_{-i} = \{\widetilde{\boldsymbol{f}}_1, \widetilde{\boldsymbol{f}}_2, \dots, \widetilde{\boldsymbol{f}}_{i-1}, \widetilde{\boldsymbol{f}}_{i+1}, \dots, \widetilde{\boldsymbol{f}}_n\}$ is the charging strategies of all the regions excluding Region i.

3.2.3. Lower-layer equilibrium condition

When solving Eq. (5) for each region, the regions minimize their own charging costs. This forms a non-cooperative game where regions' decisions are affected by each other due to the queuing costs. Therefore, the lower-layer problem's solution can be characterized by the Nash equilibrium. In an equilibrium state, no region can decrease its charging cost by unilaterally changing its charging strategy. Specifically, the lower-layer equilibrium among EV flows can be described as

$$C_i(\widetilde{\boldsymbol{f}}_i^*; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p}) \le C_i(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p}), \forall \widetilde{\boldsymbol{f}}_i \ne \widetilde{\boldsymbol{f}}_i^*, i \in \mathcal{R}, \tag{6}$$

where $\widetilde{\boldsymbol{f}}_i^*$ denotes the charging strategy of EV flows in region i in the equilibrium. Furthermore, the lower-layer equilibrium solution can be denoted as $\widetilde{\boldsymbol{f}}^* = (\widetilde{\boldsymbol{f}}_1^*, \widetilde{\boldsymbol{f}}_2^*, \dots, \widetilde{\boldsymbol{f}}_n^*)$.

3.2.4. Reformulating the game as MPEC

We have formulated the two-layer hierarchical game, which consists of Problem (4) at the leader's layer and Problem (5) at the followers' layer. The leader's optimization problem is constrained by the equilibrium of the followers' non-cooperative game. As the EV flows' equilibrium lacks closed-form expressions, the leader's optimization problem cannot be solved through the classical backward induction methods. To address this problem, we reformulate the CS manager's optimization problem as Problem (7).

$$\begin{aligned} \max_{\boldsymbol{p}} \quad & V(\boldsymbol{p}) = \sum_{j=1}^{m} \left((p_j - \varepsilon_j) \sum_{i=1}^{n} f_{ij}^* \right) \\ s.t. \quad & \begin{cases} \varepsilon_j < p_j \le p^{max}, \quad \forall j \in \mathcal{C}, \\ \widetilde{\boldsymbol{f}}_i^* = \arg\min C_i(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p}), \quad \forall i \in \mathcal{R}, \\ s.t. \begin{cases} \sum_{j=1}^{m} f_{ij}^* = N_i, \\ f_{ij} \ge 0, \forall j \in \mathcal{C}, \end{cases} \end{cases} \end{aligned} \tag{7}$$

where the lower-layer equilibrium condition in (6) is a constraint of the upper optimization problem in (4). Then f_{ij}^* is the EV flow from Region i to CS j at the lower-layer Nash equilibrium, which is a part of the charging strategy $\widetilde{\boldsymbol{f}}_i^*$. With Problem (7), we define the EV-charging pricing (EV-CP) problem as follows.

Problem Definition (EV-CP). *Given an EV charging market, our goal is to solve Problem (7) to optimize the market's total profit at the EV flows' Nash Equilibrium.*

The EV-CP problem corresponds to a mathematical program with equilibrium constraints (MPEC) [46]. Due to the inherent non-smooth constraints of MPEC, MPEC is a classic non-convex optimization problem, and it is usually intractable to analyze the optimality of the MPEC's solution. In the next section, we introduce how to solve the EV-CP and prove the convergence and optimality of our solution.

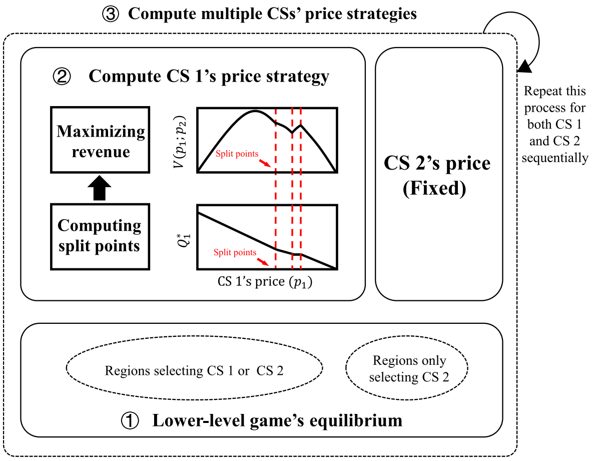

③ Compute multiple CSs' price strategies
② Compute CS 1's price strategy
Maximizing revenue
Computing split points
$V(p_1; p_2)$
Q_1^*
Split points
Split points
CS 1's price (p_1)
CS 2's price (Fixed)
Repeat this process for both CS 1 and CS 2 sequentially
Regions selecting CS 1 or CS 2
Regions only selecting CS 2
① Lower-level game's equilibrium

Fig. 3. Overview of SPITER.

4. Algorithm description

In this section, we begin by providing an overview of our solution. Next, we will analyze the properties of the lower-layer equilibrium. Using these properties as a foundation, we introduce a new segmentation-based algorithm called SPITER, which aims to solve the EV-CP problem.

4.1. Overview

SPITER is based on the block coordinate descent (BCD) method [47], which is an iterative algorithm for nonconvex optimization. It works by minimizing the objective function in each block coordinate, where in our case, a CS's pricing strategy is considered as a block coordinate. For example, in Fig. 3, we have a market with two CSs and multiple regions. This iteration's focus is on optimizing the price of CS 1 while keeping CS 2's price fixed. To determine the optimal price for CS 1, we need to understand how the queuing cost, denoted as Q_1^*, changes with the CS 1's price p_1. We observe that $Q_1^*(\cdot)$ behaves like a piece-wise linear function of p_1. As p_1 increases, $Q_1^*(\cdot)$ decreases linearly because some EV flows that originally chose CS 1 will switch to selecting CS 2. When the EV flows from a region to CS 1 decrease to 0, the slope of the linear function changes, creating a split point in $Q_j^*(\cdot)$. By identifying all the split points of $Q_j^*(\cdot)$, we can use quadratic optimization to calculate the highest achievable profit. Once we have determined the best profit for each segment of the piece-wise function, finding CS 1's optimal pricing strategy becomes straightforward. SPITER repeats this process sequentially for both CS 1 and CS 2 to optimize the overall profit.

Based on this idea, the framework of SPITER is as follows. We first rewrite the lower-layer equilibrium condition as nonlinear complementarity constraints to transform the EV-CP to a single-level optimization problem (Section 4.2). We then divide the rewritten EV-CP problem into a set of subproblems each of which corresponds to a block coordinate. Although the subproblems are also non-convex, SPITER computes their optimal solutions by using a segmentation method to determine all the split points of $Q_j^*(\cdot)$ (Section 4.3). By repeatedly solving the subproblems, SPITER finally converges to a local maximum point of the original EV-CP problem (Section 4.4).

4.2. Lower-layer game's equilibrium

To compute the lower-layer equilibrium solution, we use the following proposition.

Algorithm 1 Computing Lower-Layer Equilibrium

```
Input: The charging prices of all the CSs $\boldsymbol{p}$;
Output: The lower-layer equilibrium solution $\widetilde{\boldsymbol{f}}^*$;
Initialize the step $t \leftarrow 0$;
Initialize a feasible solution $\widetilde{\boldsymbol{f}}^0$;
repeat
    $i \leftarrow$ the next follower;
    $\widetilde{\boldsymbol{f}}_i^{t+1} \leftarrow \arg\min_{\widetilde{\boldsymbol{f}}_i \in \mathcal{F}_i} C_i(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}^t, \boldsymbol{p})$;
    $\widetilde{\boldsymbol{f}}^{t+1} \leftarrow \{\widetilde{\boldsymbol{f}}_{-i}^t, \widetilde{\boldsymbol{f}}_i^{t+1}\}$;
    $t \leftarrow t + 1$;
until Equilibrium condition (Eq. (6)) is satisfied or $t \leq I_0$
return $\widetilde{\boldsymbol{f}}^t$
```

Proposition 1 (*Lower-Layer Game's Equilibrium*)**.** *There always exists a unique equilibrium in the lower-layer game. Each follower's optimization problem can converge to a unique solution in the equilibrium state, regardless of the pricing strategy* $\boldsymbol{p}$ *at the upper layer.*

Proof. The objective function of Problem (5) is continuous, and the inequality and equality constraints are convex. Therefore, the feasible sets of (5) are closed, nonempty, and convex. The Hessian matrix of the cost function C_i is positive definite, which means that the second-order partial derivatives are greater than zero, *i.e.,* $\nabla^2 C_i \succ 0$ and the cost function C_i is strictly convex. Thus, the followers form a concave n-person game. By Rosen's uniqueness theorem [48], the lower-layer game always has a unique equilibrium, and each follower's optimization problem can converge to a unique solution in the equilibrium state, regardless of the pricing strategy $\boldsymbol{p}$ at the upper layer. □

Based on Rosen's convergence theorem [48], we propose an iterative algorithm (Algorithm 1) to compute the lower-layer equilibrium. Each iteration solves a follower's optimization problem, *i.e.,* $C_i(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p})$. Because $C_i(\cdot)$ is a classic convex quadratic programming problem, we can compute its optimal charging strategy, *i.e.,* the best-response strategy, using the off-the-shelf quadratic programming solvers. The algorithm iteratively solves each follower's optimization problem, until no one wants to change her charging strategy. Then, the equilibrium condition (Eq. (6)) is satisfied and the latest charging strategy is returned.

As each follower's best-response strategy is the optimal solution to a convex problem, we can apply the Karush–Kuhn–Tucker (KKT) condition [49] to equivalently present the equilibrium constraints in Problem (7). Let $L_i(\cdot, \cdot, \cdot)$ be the Lagrange function of follower i, such that

$$L_i(\widetilde{\boldsymbol{f}}_i, \boldsymbol{v}_i, \lambda_i) = C_i(\widetilde{\boldsymbol{f}}_i; \widetilde{\boldsymbol{f}}_{-i}, \boldsymbol{p}) - \widetilde{\boldsymbol{f}}_i \boldsymbol{v}_i + \lambda_i (N_i - \sum_{j=1}^{m} f_{ij}), \tag{8}$$

where $\widetilde{\boldsymbol{f}}_i = [f_{i1}, f_{i2}, \ldots, f_{im}]$ is the vector of charging strategies, and $\boldsymbol{v}_i = [v_{i1}, v_{i2}, \ldots, v_{im}]^T$ ($v_{ij} \geq 0, \forall j \in 1, \ldots, m$) and $\lambda_i \in R$ are the corresponding Lagrange multipliers that correspond to the inequality constraint functions (*i.e.,* $f_{ij} \geq 0$) and equality constraint function (*i.e.,* $N_i - \sum_{j=1}^{m} f_{ij} = 0$), respectively. The Lagrange function models the relationship between the gradient of the function and gradients of the constraints. Since each follower's best-response strategy is the minimum point of his cost function, the partial derivative $\nabla_{\widetilde{\boldsymbol{f}}_i} L_i$ of the Lagrangian $L_i(\cdot, \cdot, \cdot)$ with respect to each charging strategy f_{ij} equals to zero. Thus, we derive the KKT conditions for all the followers as Eq. (9), where the complementary slackness conditions $v_{ij} f_{ij} = 0$ reflect the fact that either the primal constraint (*i.e.,* $f_{ij} \geq 0$) is active at the optimal solution, or the Lagrange multiplier v_{ij} associated with the constraint is zero.

$$\text{KKT}\begin{cases} \nabla_{\tilde{f}_i} L_i = 0, & \forall i \in \mathcal{C} \\ \sum_{j=1}^{m} f_{ij} = N_i, & \forall i \in \mathcal{C}, \\ v_{ij} f_{ij} = 0, & \forall i \in \mathcal{C}, \forall j \in \mathcal{R}, \\ v_{ij} \geq 0, & \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \\ f_{ij} \geq 0, & \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \end{cases} \tag{9}$$

Then, the EV-CP can be transformed into a single-level nonlinear optimization problem, where the equilibrium constraints are replaced by the KKT conditions derived in Eq. (9).

$$\max_{\boldsymbol{p}} \quad V(\boldsymbol{p}) = \sum_{j=1}^{m} \left(p_j - \varepsilon_j\right) \hat{f}_j \tag{10}$$

$$\begin{aligned} s.t. \quad & \varepsilon_j < p_j \leq p^{max}, \\ & \text{KKT conditions as (9).} \end{aligned} \tag{11}$$

4.3. One CS's pricing strategy

To solve Problem (10), we first decompose the problem into a set of subproblems, each of which aims to analyze the effect of one considered CS's strategy contributing to the market profit, while the other CSs' pricing strategies are held fixed.

Subproblem. We denote the subproblem by V_s, and use $\boldsymbol{p}_{-j'}$ to denote the pricing strategies of all CSs excluding CS j'. The subproblem regarding CS j' is defined as

$$\begin{aligned} \max_{p_{j'}} \quad & V_s(p_{j'}; \boldsymbol{p}_{-j'}) \\ = & (p_{j'} - \varepsilon_{j'})\hat{f}_{j'} + \sum_{j \in \mathcal{R} \setminus \{j'\}} (p_j - \varepsilon_j)\hat{f}_j \\ s.t. \quad & \varepsilon_{j'} < p_{j'} \leq p^{max}, \\ & \text{KKT conditions as (9).} \end{aligned} \tag{12}$$

Comparing Problems (10) and (12), the only difference is that the variable of Problems (10) is an m-dimensional vector, *i.e.*, $\boldsymbol{p}$, while the variable of Problems (12) is the price of one considered CS $p_{j'}$ since other CSs' prices are fixed.

The goal of a subproblem is to maximize the market profit when changing CS $p_{j'}$'s price. With the BCD method, we can iteratively solve the subproblem for each CS. Once the solutions are optimal, the BCD method guarantees the algorithm's convergence to a local maximum of Problem (10). However, the subproblems are intractable to tackle, because they also are non-convex problems containing $m \times n$ complementary constraints *i.e.*, $v_{ij} f_{ij} = 0$.

4.3.1. Segmenting subproblem

We introduce our method for finding the optimal solution of each subproblem. Our method is based on a novel segmentation technique that decomposes the subproblem into a series of convex problems. We find the optimal solution after all the subproblem segments have been evaluated.

The basic idea of our method is to convert the complementary constraints into linear equality constraints. We observe that the complementary constraints correspond to two cases: (1) $v_{ij} = 0$ and $f_{ij} > 0$; (2) $v_{ij} > 0$ and $f_{ij} = 0$, $\forall i \in \mathcal{C}, \forall j \in \mathcal{R}$. When the price $p_{j'}$ changes from $\varepsilon_{j'}$ to p^{max}, parts of the complementary constraints may switch between the two cases. To distinguish between the two cases, we have the following two definitions,

$$S_l = \left\{ f_{ij} \middle| \ f_{ij} > 0, \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \right\}, \tag{13a}$$

$$S_e = \left\{ f_{ij} \middle| \ f_{ij} = 0, \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \right\}, \tag{13b}$$

where S_l is the set containing all f_{ij} larger than 0 and S_e is the set containing all f_{ij} equal to 0. Given fixed S_l and S_e, we can eliminate the multipliers v_{ij}, and thus, the complementary constraints are converted into linear equality constraints.

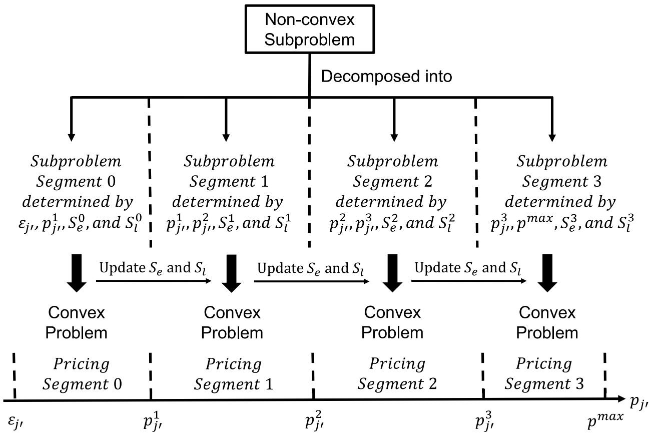

Non-convex Subproblem
Decomposed into
Subproblem Segment 0 determined by $\varepsilon_{j'}, p_{j'}^1, S_e^0$, and S_l^0
Subproblem Segment 1 determined by $p_{j'}^1, p_{j'}^2, S_e^1$, and S_l^1
Subproblem Segment 2 determined by $p_{j'}^2, p_{j'}^3, S_e^2$, and S_l^2
Subproblem Segment 3 determined by $p_{j'}^3, p^{max}, S_e^3$, and S_l^3
Update S_e and S_l
Update S_e and S_l
Update S_e and S_l
Convex Problem
Convex Problem
Convex Problem
Convex Problem
Pricing Segment 0
Pricing Segment 1
Pricing Segment 2
Pricing Segment 3
$\varepsilon_{j'}$
$p_{j'}^1$
$p_{j'}^2$
$p_{j'}^3$
p^{max}
$p_{j'}$

Fig. 4. Overview of segmentation technique. The technique decomposes the subproblem into a series of convex problems by finding the split points.

Incorporating this idea, Fig. 4 overviews the segmentation technique, where the feasible region of $p_{j'}$, $(\varepsilon_{j'}, p^{max}]$, is divided into multiple pricing segments. Within each pricing segment, all the complementary constraints do not switch between the two cases, so they can be transformed into multiple linear equality constraints containing Eq. (9) for $f_{ij} \in S^l$ and $f_{ij} = 0$ for $f_{ij} \in S^e$. By this transformation, the subproblem segment can be regarded as a convex problem. Additionally, as the feasible range of $p_{j'}$ is continuous and convex, the pricing segments are continuous and convex. Thus, the subproblem can be decomposed into a series of convex problems.

Subproblem Segments. Each convex problem corresponds to a subproblem segment. Formally, each subproblem segment is defined as

$$\begin{aligned} \max_{p_{j'}} \quad & V_s^k(p_{j'}; \boldsymbol{p}_{-j'}) \\ s.t. \quad & \begin{cases} p_{j'}^k < p_{j'} \leq p_{j'}^{k+1}, \\ f_{ij} = 0, \quad \forall f_{ij} \in S_e^k, \\ \nabla_{\tilde{f}_i} L_i = 0, \quad \forall f_{ij} \in S_l^k, \\ \sum_{j=1}^{m} f_{ij} = N_i, \quad \forall i \in \mathcal{C}, \\ f_{ij} \geq 0, \quad \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \end{cases} \end{aligned} \tag{14}$$

As shown in (14), we use a superscript k to differentiate the subproblem segments such that V_s^k denotes the kth subproblem segment lies on the kth pricing segment $(p_{j'}^k, p_{j'}^{k+1}]$. Moreover, S_l^k and S_e^k indicates the sets containing all f_{ij} larger than 0 and all f_{ij} equal to 0, respectively, in the kth subproblem segment.

Split Point. We refer to $p_{j'}^k$ as a split point of the subproblem. The split points decompose the feasible range of price into continuous pricing segments. In two successive pricing segments separated by $p_{j'}^k$, the sets S_l and S_e are changed such that $\exists f_{ij}, f_{ij} \in S_l^{k-1}$ but $f_{ij} \notin S_l^k$ (or $f_{ij} \notin S_l^{k-1}$ but $f_{ij} \in S_l^k$). On the other hand, in each pricing segment, the elements in the two sets remain unchanged.

4.3.2. Optimal algorithm for each subproblem

Our algorithm is based on the computation of a split points. Given two successive split points $p_{j'}^k$ and $p_{j'}^{k+1}$, we are able to determine the sets S_l^k, and S_e^k. Once we know S_l^k and S_e^k, the kth subproblem segment becomes a well-defined quadratic programming problem, which can be solved by standard optimization tools. Here we show how to compute $p_{j'}^k$, S_l^k, and S_e^k, and presents the optimal algorithm for each subproblem.

Computing $p_{j'}^0$, S_l^0, and S_e^0. Because $\varepsilon_{j'} \leq p_{j'} \leq p^{max}$, the first Split Point (*i.e.*, $p_{j'}^0$) equals to $\varepsilon_{j'}$. Combining the other charging stations' pricing strategies $\boldsymbol{p}_{-j'}$ and $p_{j'}^0$, we obtain the CS manager's pricing strategy $\boldsymbol{p}$. After that, the equilibrium strategy of EV flows in different regions can be computed by Algorithm 1. Then, we compute S_l^0 and S_e^0 by Eq. (13a) and (13b), respectively.

Computing $p_{j'}^{k+1}$, S_l^{k+1}, and S_e^{k+1}. We then introduce the procedure to compute $p_{j'}^{k+1}$, S_e^{k+1}, and S_l^{k+1}, given that $p_{j'}^k$, S_e^k, and S_l^k are known. Specifically, we define that $|S_l^k|$ is the number of f_{ij} larger than 0. Similarly, $|S_e^k|$ is the number of f_{ij} equal to 0, such that $|S_l^k| + |S_e^k|$ equals $m \times n$. Therefore, $v_{ij} f_{ij} = 0$ in (9) can be equivalently replaced by the linear equations and constraints based on S_l^k and S_e^k as follows:

$$\omega_p p_j + \omega_q Q_j + \omega_d d_{ij} + \omega_q \frac{f_{ij}}{c_j} = \lambda_i, \quad \forall f_{ij} \in S_l^k, \tag{15a}$$

$$\sum_{f_{ij} \in S_l^k} f_{ij} = N_i, \tag{15b}$$

$$\omega_p p_j + \omega_q Q_j + \omega_d d_{ij} - \lambda_i > 0, \quad \forall f_{ij} \in S_e^k, \tag{15c}$$

$$f_{ij} \geq 0, \quad \forall i \in \mathcal{R}, \forall j \in \mathcal{C}, \tag{15d}$$

where $\omega_p p_j + \omega_q Q_j + \omega_d d_{ij} + \omega_q \frac{f_{ij}}{c_j} = \lambda_i$ is derived from the constraint $\nabla_{\tilde{f}_i} L_i = 0$ of (9) and c_j is the capacity of CS j.

As the aforementioned linear equations and constraints show, those $f_{ij} \in S_l^k$ satisfy (15a) and (15b), and those $f_{ij} \in S_e^k$ satisfy (15c) and (15d) in the equilibrium state of the EV flows. As $p_{j'}$ changes from $p_{j'}^k$ to p^{max}, some $f_{ij} \in S_l^k$ may change values and become elements of S_e^k. Additionally, some $f_{ij} \in S_e^k$ may also change values and become elements of S_l^k. These two cases correspond to situations where Split Points may occur. Therefore, we must compute all the values of $p_{j'}$ that correspond to these two cases.

For all $f_{ij} \in S_l^k$, the linear equations in (15a) and (15b) contain f_{ij} and $p_{j'}$, and these equations are linearly independent. Thus, we obtain the linear relationship between f_{ij} and $p_{j'}$ by solving (15a) and (15b). Specifically, we present the linear relationships as

$$\begin{aligned} f_{ij} &= a_{ij}^k p_{j'} + b_{ij}^k, \forall f_{ij} \in S_l^k \\ \lambda_i &= a_{\lambda_i}^k p_{j'} + b_{\lambda_i}^k \end{aligned} \tag{16}$$

where a_{ij}^k, b_{ij}^k, $a_{\lambda_i}^k$, and $b_{\lambda_i}^k$ can be computed by solving the linear equations in (15a) and (15b). When $p_{j'}$ changes, some $f_{ij} \in S_l^k$ may become lower than 0, which will trigger split points. Let $p_{j'}^{ij}$ be a candidate split point corresponding to f_{ij} such that given the linear system, if $p_{j'} = p_{j'}^{ij}$, then $f_{ij} = 0$. Solving $a_{ij}^k p_{j'}^{ij} + b_{ij}^k = 0$, we get $p_{j'}^{ij} = \frac{-b_{ij}^k}{a_{ij}^k}$.

For all $f_{ij} \in S_e^k$, they correspond to constraint (15c). When $p_{j'}$ increases, some $f_{ij} \in S_e^k$ may violate the constraint (15c). Let $p_{j'}^{ij}$ be a candidate split point corresponding to f_{ij} such that given the linear system, if $p_{j'} \leq p_{j'}^{ij}$, then $f_{ij} = 0$ and if $p_{j'} > p_{j'}^{ij}$, then $f_{ij} > 0$. To compute $p_{j'}^{ij}$, we substitute λ_i with $a_{\lambda_i}^k p_{j'}^{ij} + b_{\lambda_i}^k$, so $p_{j'}^{ij}$ must satisfy $\omega_p p_{j'}^{ij} + \omega_q Q_{j'} + \omega_d d_{ij} - a_{\lambda_i}^k p_{j'}^{ij} + b_{\lambda_i}^k = 0$. Thus, we get $p_{j'}^{ij} = \frac{b_{\lambda_i}^k - \omega_q Q_{j'} - \omega_d d_{ij}}{\omega_p - a_{\lambda_i}^k}$.

After considering these two cases, we obtain a set of candidate split points, and the next split point $p_{j'}^{k+1}$ is the minimum one among the candidates, which we compute as

$$p_{j'}^{k+1} = \min\left\{ \min_{f_{ij} \in S_l^k} \frac{-b_{ij}^k}{a_{ij}^k}, \min_{f_{ij} \in S_e^k} \frac{b_{\lambda_i}^k - \omega_q Q_{j'} - \omega_d d_{ij}}{\omega_p - a_{\lambda_i}^k} \right\}. \tag{17}$$

At the split point $p_{j'}^{k+1}$, some elements in S_l^k and S_e^k will change their status. Specifically, those $f_{ij} \in S_l^k$ that satisfy $p_{j'}^{k+1} = \frac{-b_{ij}^k}{a_{ij}^k}$ will be moved to S_e^{k+1} and those $f_{ij} \in S_e^k$ that satisfy $p_{j'}^{k+1} = \frac{b_{\lambda_i}^k - \omega_q Q_{j'} - \omega_d d_{ij}}{\omega_p - a_{\lambda_i}^k}$ will be moved to S_l^{k+1}. Overall, we compute S_l^{k+1} and S_e^{k+1} as

$$\begin{aligned} S_l^{k+1} = &\left\{ f_{ij} \middle| a_{ij} p_{j'}^{k+1} + b_{ij} > 0, \forall f_{ij} \in S_l^k \right\} \\ &\bigcup \left\{ f_{ij} \middle| \omega_p p_{j'}^{k+1} + \omega_q Q_{j'} + \omega_d d_{ij} = \lambda_i, \forall f_{ij} \in S_e^k \right\}, \\ S_e^{k+1} = &\left\{ f_{ij} \middle| a_{ij} p_{j'}^{k+1} + b_{ij} = 0, \forall f_{ij} \in S_l^k \right\} \\ &\bigcup \left\{ f_{ij} \middle| \omega_p p_{j'}^{k+1} + \omega_q Q_{j'} + \omega_d d_{ij} < \lambda_i, \forall f_{ij} \in S_e^k \right\}. \end{aligned} \tag{18}$$

If $p_{j'}^{k+1} > p^{max}$, it means all split points have been computed, so we can set $p_{j'}^{k+1} = p^{max}$ as the last split point.

The following example illustrates the criteria used for segmentation and the process of the computing the optimal solution of the subproblem.

Example 4.1. We examine an urban IoT network consisting of two CSs, denoted as CS 1 and CS 2, and three regions, labeled as Regions 1, 2, and 3. We treat the values of weights w_q, w_p, and w_d and CS capacities c_j as one. The constant d_{ij} is the spatial distance from Region i to CS j. We set the distance d_{11} to zero, while all other distances are set to values greater than zero. The price for CS 2 is set at p^{max}, and our objective is to determine the optimal pricing strategy for CS 1. (i) Initially, we set CS 1's first price p_1^0 as ϵ_1, establishing an initial price vector $\boldsymbol{p}^0$ for all CSs as $\langle \epsilon_1, p^{max} \rangle$. This pricing strategy dictates the initial charging strategies for the regions upon reaching the lower-level equilibrium. In this case, all the EV flows of Region 1 select CS 1 due to the shorter distance (*i.e.*, $f_{11} = 1$ and $f_{12} = 0$), while the EV flows of Regions 2 and 3 select both CS 1 and CS 2, resulting in $f_{21}, f_{22}, f_{31}, f_{32} > 0$. Therefore, the set S_l^0 is $\{f_{11}, f_{21}, f_{22}, f_{31}, f_{32}\}$, and the set S_e^0 is $\{f_{12}\}$. (ii) Next, we proceed to determine the price segment $[p_1^0, p_1^1]$. According to Eqs. (15a) and (15b), the lower-level equilibrium corresponding to $[p_1^0, p_1^1]$ satisfies the following linear system (19), where p_1, f_{ij}, and λ_i are variables. As we increase p_1, the other variables f_{21}, f_{21}, f_{31}, f_{32}, λ_1, λ_2, and λ_3 change linearly, leading to an increase in the number of EVs moving from CS 1 to CS 2. We stop the increase in p_1 at a split point by evaluating Eq. (17). The split point indicates that either f_{21} or f_{31} drops to zero or λ_1 violates constraint (15c) (*i.e.*, f_{12} increasing beyond zero), resulting in the new sets S_l^1 and S_e^1, and a new linear system. (iii) We then determine the other price segments $(p_1^1, p_1^2]$, $(p_1^2, p_1^3]$, …, $(p_1^k, p^{max}]$ based on the specified criteria. Each price segment corresponds to a quadratic programming problem (14). We solve all the quadratic programming problems along p_1's domain to compute the optimal solution of the subproblem.

Proposition 2 (*Optimality of Algorithm 2*). *Algorithm 2 is optimal for the subproblem.*

Proof. Algorithm 2 is a one-dimensional line search method that divides the domain $(\epsilon, p^{max}]$ of the subproblem into successive segments. Each segment contains exactly one local maximum by solving the quadratic objective function, so Algorithm 2 is able to find all local maximums of the subproblem. Thus, the returned price strategy is an optimal solution. □

$$\left\{ \begin{aligned} & p_1 + \textstyle\sum_{i \in \{1,2,3\}} f_{i1} + d_{11} + f_{11} = \lambda_1 \\ & p_1 + \textstyle\sum_{i \in \{1,2,3\}} f_{i1} + d_{21} + f_{21} = \lambda_2 \\ & p^{max} + \textstyle\sum_{i \in \{1,2,3\}} f_{i2} + d_{22} + f_{22} = \lambda_2 \\ & p_1 + \textstyle\sum_{i \in \{1,2,3\}} f_{i1} + d_{31} + f_{31} = \lambda_3 \\ & p^{max} + \textstyle\sum_{i \in \{1,2,3\}} f_{i2} + d_{32} + f_{32} = \lambda_3 \\ & f_{11} = N_1 \\ & f_{12} = 0 \\ & f_{21} + f_{22} = N_2 \\ & f_{31} + f_{32} = N_3 \end{aligned} \right. \tag{19}$$

Algorithm 2 Optimal Algorithm for the Subproblem

```
Input: The charging prices of the other CSs $\boldsymbol{p}_{-j'}$;
Output: The best pricing strategy
 1: Initialize index $k = 0$ for $p_{j'}$;
 2: Initialize $p_{j'}^{k} = \epsilon_{j'}$;
 3: Initialize a pricing strategy $\boldsymbol{p}_{-j'}$;
 4: Compute the lower-layer equilibrium solution $\widetilde{\boldsymbol{f}}^{*}$ by Algorithm 1;
 5: while $p_{j'}^{k} < p^{max}$ do
 6:     Formulate the linear equations and constraints as (15a), (15b), (15c), and (15d);
 7:     Solve the linear equations to obtain the linear relationships in (16);
 8:     Compute the next split point $p_{j'}^{k+1}$ by (17);
 9:     Compute $S_l^{k+1}$ and $S_e^{k+1}$ by (18);
10:     $k \leftarrow k + 1$;
11: end while;
12: Obtain all subproblem segments based on the split points;
13: Compute the optimal pricing and corresponding profit for each subproblem segment;
14: return The best pricing strategy;
```

4.4. Multi-CSs' cooperative strategies

We show how SPITER computes CSs' cooperative strategies with the optimal solution to each subproblem. Basically, SPITER cyclically updates the CSs' pricing strategies, until all CSs cannot change their pricing to make the total profit higher or the number of iterations exceeds the predefined termination threshold I_1. This idea is similar to the block coordinate descent method [47] that the maximization of a multivariable function $V(\boldsymbol{p})$ can be achieved by maximizing it along one direction at a time, *i.e.,* solving univariate optimization problems in a loop.

As Algorithm 3 shows, SPITER starts with initial variable values $\boldsymbol{p}^0 = (p_1^0, \dots, p_n^0)$, round $t+1$ defines $\boldsymbol{p}^{t+1}$ from $\boldsymbol{p}^t$ by iteratively solving the single variable subproblems by Algorithm 2, such that

$$p_j^{t+1} = \underset{y \in (\epsilon_j, p^{max}]}{\arg\max}\ V_s(y; p_1^{t+1}, \dots, p_{j-1}^{t+1}, p_{j+1}^{t}, \dots, p_n^{t})$$

for each variable p_i of $\boldsymbol{p}$, for j from 1 to n. Thus, one begins with an initial guess $\boldsymbol{p}^0$ for a local maximum of $V()$, and gets a sequence $\boldsymbol{p}^0, \boldsymbol{p}^1, \boldsymbol{p}^2, \dots$ iteratively.

Proposition 3 (*Convergence and Optimality of* SPITER). SPITER *can converge to a local maximum of the EV-CP problem.*

Proof. We show that SPITER is convergent. Since $V_s(p_j^{t+1}; \boldsymbol{p}_{-j}^{t})$ is the global maximum of the subproblem, we have $V(\boldsymbol{p}^{t+1}) \geq V(\boldsymbol{p}^t)$. Thus, the market profit increases monotonically with the number of steps t, so that $V(\boldsymbol{p}^0) \leq V(\boldsymbol{p}^1) \leq V(\boldsymbol{p}^2) \leq \cdots$. Since the solution set of EV-CP is bounded, SPITER will eventually stop at a point where changing any coordinate will not improve the overall profit. This is how we prove the convergence of SPITER.

We prove that SPITER stops at a local maximum of EV-CP. This proof relies on the optimality property of the BCD method, which states that if the optimization problem is convex and differentiable, the BCD method converges to the global optimal solution of the problem. In our scenario, given the sets S_l and S_e, we can determine which EV flow f_{ij} is zero, thus eliminating the complementary constraint in the KKT condition (9). Suppose S_l and S_e determine the subdomain $\mathcal{X}$ of the problem (10), then our problem is convex and differentiable within $\mathcal{X}$. If SPITER stops at a point inside $\mathcal{X}$, it reaches the stationary point for that subset, and thus reaches a local maximum of EV-CP.

On the other hand, the BCD method can get stuck if the optimization problem is not smooth. Thus, if SPITER stops at the boundary of $\mathcal{X}$ (*i.e.,* at a split point $\boldsymbol{p}$), it is possible that the result is not a local maximum of EV-CP. We fix this issue by testing all subsets connecting the boundary. We assume that f_{ij} at the split point activates the constraints (15c) and (15d), *i.e.,* $\omega_p p_j + \omega_q Q_j + \omega_d d_{ij} - \lambda_i - \epsilon = 0$ (ϵ is an arbitrary small positive number) and $f_{ij} = 0$. We can construct a new pair of sets $S_l' = S_l \cup \{f_{ij}\}$ and $S_e' = S_e \setminus \{f_{ij}\}$, leading to a new subdomain $\mathcal{X}'$ of the problem (10), which connects the boundary of $\mathcal{X}$. Our problem inside $\mathcal{X}'$ is also convex and differentiable, so we can find an optimal point $\boldsymbol{p}'$ over $\mathcal{X}'$. We then compare $\boldsymbol{p}$ and $\boldsymbol{p}'$. Thus, if $\boldsymbol{p}$ is better than the optimal points of all subsets connecting the boundary, $\boldsymbol{p}$ is the local maximum; otherwise, we return the best $\boldsymbol{p}'$. If the best $\boldsymbol{p}'$ is on a different boundary, we repeat the above process until a local maximum is found. Since the number of different pairs of S_l and S_e is at most $O(2^{mn})$, we can guarantee that SPITER eventually converges to a local maximum of EV-CP. □

Algorithm 3 SPITER Algorithm for EV-CP

```
Input: The set of CSs and Regions in the game;
Output: The pricing strategy for all CSs
Initialize the step $t \leftarrow 0$;
Initialize a feasible pricing strategy $\boldsymbol{p}^0$ ;
repeat
    $j \leftarrow$ The next CS;
    $p_j^{t+1} \leftarrow \arg\max_{p_j \in (\epsilon_j, p^{max}]} V_s(p_j; \boldsymbol{p}_{-j}^{t})$
    $\boldsymbol{p}^{t+1} \leftarrow \{\boldsymbol{p}_{-j}^{t}, p_j^{t+1}\}$;
    $t \leftarrow t + 1$;
until No CS changes her pricing strategy or $t \leq I_1$;
return $\boldsymbol{p}^t$;
```

SPITER can be extended to the distributed environments without the centralized coordination. In such a case, each CS repeatedly receives other CSs' pricing strategies and solves the corresponding subproblem. Let $\boldsymbol{p}^*$ be the returned cooperative price strategy. Since no CS will change its strategy, $\left(p_j^*, V_s(p_j^*; \boldsymbol{p}_{-j}^*)\right)$ is the global maximum point for each CS j. Thus, the algorithm converges to a local maximum point of EV-CP in the distributed environment.

Time Complexity. We analyze the time complexity of SPITER. First, in Line 4 of Algorithm 2, we calculate the lower-layer equilibrium solution using Algorithm 1. This step takes $O(I_0 m^3)$ time, where I_0 is the number of iterations needed to find the Nash equilibrium, and $O(m^3)$ represents the time for determining the best response through quadratic programming. Next, in Lines 5–10 of Algorithm 2, we compute a set of split points. For each split point, we solve linear equation systems with $O(mn)$ variables, which requires $O((nm)^3)$ time. The number of split points depends on whether f_{ij} equals zero or not and can be $O(2^{mn})$. Thus, combining these steps, the time complexity of Algorithm 2 is approximately $O(I_0 m^3 + 2^{mn}(nm)^3)$. SPITER cyclically updates the CSs' pricing strategies by Algorithm 2, so the overall time complexity is approximately $O(I_1(I_0 m^3 + 2^{mn}(nm)^3))$ where I_1 is the number of iterations needed by SPITER. The values of both I_1 and I_0 are determined by the termination conditions set within the iterative algorithms.

5. Optimization techniques

We enhance the computational performance of SPITER by implementing optimization techniques. These techniques involve fast initialization with relaxation and distributed evaluation methods. These optimization techniques enhance the computational performance, increasing the adaptive capability of SPITER in practical environments. In this section, we provide a detailed discussion of these optimization techniques.

5.1. Relaxation method

We propose a relaxation method to accelerate the computation of SPITER's initial pricing strategy, *i.e.,* line 2 of Algorithm 3. An optimized initial pricing strategy can increase overall profit. While setting the initial price of each CS to p^{max} is a straightforward approach, it may lack flexibility for different problem instances. In contrast, our relaxation method selects initial pricing strategies for various problem instances by solving a sequence of smooth, regularized problems that progressively approximate the original electric vehicle charging problem (EV-CP).

The original EV-CP is difficult to solve due to the existence of non-smooth complementary constraints. The relaxation method perturbs the original EV-CP by smoothing the complementary constraints [50,51]. The perturbed problem with parameter μ is as follows.

$$\max_{\boldsymbol{p}} \quad V(\boldsymbol{p},\mu) = \sum_{j=1}^{m} \left(p_j - \varepsilon_j\right) \hat{f}_j \tag{20}$$

$$s.t. \quad \nabla_{\widetilde{f}_i} L_i = 0, \quad \forall i \in \mathcal{C} \tag{20a}$$

$$\sum_{j=1}^{m} f_{ij} = N_i, \quad \forall i \in \mathcal{C}, \tag{20b}$$

$$v_{ij} \geq 0, \quad \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \tag{20c}$$

$$f_{ij} \geq 0, \quad \forall i \in \mathcal{C}, \forall j \in \mathcal{R} \tag{20d}$$

$$f_{ij} - z_{ij} = 0, \quad \forall i \in \mathcal{C}, \ \forall j \in \mathcal{R}, \tag{20e}$$

$$\sqrt{\left(z_{ij} - v_{ij}\right)^2 + 4\mu^2} - \left(z_{ij} + v_{ij}\right) = 0, \tag{20f}$$

where $\boldsymbol{z} \in \mathbb{R}^{m \times n}$ is an auxiliary variable. In the perturbed problem, the complementary constraints of EV-CP (Problem (10)) are replaced by constraints (20e) and (20f), while the other constraints are the same. When $\mu = 0$, the last constraint (20f) may reduce to two cases: (1) $v_{ij} = 0$ and $\sqrt{z_{ij}^2} - z_{ij} = 0$, (2) $f_{ij} = 0$ and $\sqrt{(-v_{ij})^2} - v_{ij} = 0$. Since these two cases correspond to the complementary constraints in (10), the perturbed problem is equivalent to the original EV-CP when $\mu = 0$. When $\mu > 0$, the perturbed problem is a well-defined smooth problem, and it can be solved by standard optimization tools. Based on the observation, the relaxation method iteratively decreases μ ($\mu \to 0$) and solves a series of perturbed problems. When the computation converge, the method returns the last solution as an initial pricing strategy for SPITER.

5.2. Distributed SPITER

We propose a distributed version of SPITER to adapt to the practical urban environment, addressing the dynamic scenario where the charging demand changes over time and the global CS manager may not exist in the market. We first introduce the modeling of urban environment and then show how the distributed SPITER works in the urban environment.

We establish a discrete-event system to model the urban environment. The system divides the timeline into equal-sized intervals and processes discrete events sequentially. The first discrete event type in the system involves EV charging demands, which the system generates at each time interval. For a specific region i, we utilize the Poisson distribution with parameter θ_i to model EV charging demand events, where θ_i signifies the arrival rate, *i.e.,* the expected number of EVs generated in region i per time slice. The system randomly selects the location within the region for the generated EV. Once an EV charging demand is initiated, the EV selects a charging station to drive to, with the assumption that the EV will not change its decision. The second type of event involves EV arrival, where the arrival time of an EV is determined by its distance to each CS. Upon reaching the selected CS, the EV searches for an available charging pile where an EV can connect and charge its battery. If a charging pile is vacant, it starts charging, and the charging time remains constant. Otherwise, the EV joins a waiting queue. The third event is EV departure; after completing charging, the EV leaves the CS, making the charging piles available again. Subsequently, the EVs in the waiting queue begin charging. Additionally, the system contains strategy updating events that periodically update CS pricing and EV charging strategies.

Based on the urban environment, we propose the distributed SPITER to update CS pricing and EV charging strategies without the global CS manager. In the real-life scenarios, the urban environment is large and dynamic with tens of thousands of EVs. To handle such scenarios, the distributed SPITER operates at a coarse granularity, treating the region-level EV flow as followers instead of individual EVs. The distributed SPITER consists of the sets of CSs and Region Coordinators (RCs) for updating CS pricing and EV charging decisions, respectively. Each CS solves the subproblem (12) with Algorithm 2 to decide its pricing strategy, while each RC collects the EV charging demands in the region and optimizes the Eq. (5) to decide the number of EVs to send to each CS.

Here are the details of how the distributed SPITER works: We assign time steps to the CSs and RCs to generate strategy events in the discrete event system. Specifically, RC i updates its EV charging strategies at time T^{RC}_{km+i}, while CS j updates its pricing strategies at time T^{CS}_{kn+j}. In this context, the value of k is set to $0, 1, \ldots$; the values m and n represent the number of RCs and CSs, respectively; and the time $T_a < T_b$ if $a < b$. When updating the EV charging strategies, RC i collects the EV arrival rates, represented as θ_i, and determines the number of EVs to charge (*i.e.,* N_i) to be equal to θ_i. It then calculates the number of EVs, $\widetilde{\boldsymbol{f}}_i$, intended for each CS by optimizing Eq. (5). The optimized charging strategy, $\widetilde{\boldsymbol{f}}_i$ guides an EV in region i in selecting a CS to drive to, based on the probability ratio of $\frac{f_{ij}}{N_i}$. When updating the pricing strategies, CS j first receives the EV arrival rates from the RCs and the latest pricing strategies from other CSs. It then solves the subproblem (12) with the Algorithm 2 to decide its pricing strategy. Since we have assigned time steps to the CSs and RCs, the updates of the RCs and CSs are performed in a certain order. Consequently, as long as the EV arrival rates are unchanged, the distributed SPITER can converge to a local optimum according to the Propositions 1 and 3.

6. Performance evaluation

In this section, we show the effectiveness of the model and SPITER through simulations. The evaluation is performed on a 64-bit machine with 32-GB RAM and a six-core Intel i7-8770 3.20-GHz processor. We solve the optimization problem with Python 3.8, SciPy 1.5.2 and Pyomo 5.7.3.

6.1. Experimental settings

We first show how to set the parameters and introduce the algorithms evaluated in the simulations.

6.1.1. Urban data and parameters

We collect the EVs' and CSs' data of Nanjing, China (NJ)[1] and New York City, US (NYC).[2] As Fig. 5 shows, we divide Nanjing into 11 regions and New York City into 10 regions according to the official administration information. For simplicity, all CSs in one region are abstracted as one CS. Based on the assumption in [23], 5% of the EVs in a city need to be charged. Since there are about 100,000 EVs in NJ and 80,000 EVs in NYC, we suppose 5000 EVs and 4000 EVs need to be charged in NJ and NYC, respectively. We assume that

[1] http://www.cheyanjiu.com/info.php?CateId=19.

[2] https://www1.nyc.gov/html.

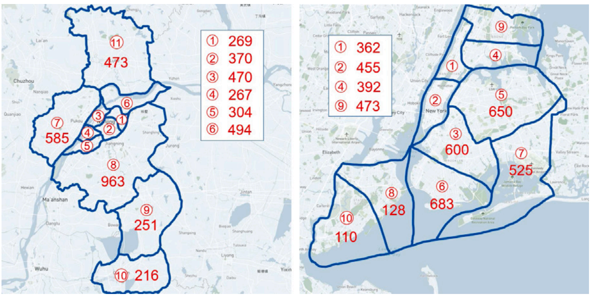


Fig. 5. EVs distribution in Nanjing (NJ, left) and New York City (NYC, right). The circled numbers indicate the regions' and CSs' indices.

Table 2
Experimental parameters.

Parameters	Description	Value
ω_p, ω_q, ω_d	Different weights	0.6, 0.1, 0.3
N_j^c	Charging capacity	4∼32
p^{max}	Ceiling of charging price	90
d_{ij}	Distance to CS	5∼80
n	Number of regions	11 (NJ), 10 (NYC)
ϵ_j	Operating cost	10∼70

the EVs' distribution follows the residential population distribution. Then, we obtain the EV distributions for the regions. Fig. 5 shows the EV distributions, and regions' and CSs' indices. In the following experiments, we select the regions (CSs) with indices 1 to k if the number of regions (CSs) is k.

Table 2 lists the experimental parameters. The distance from each region to each CS is obtained from Google Maps. According to the data on Telaidian (www.teld.cn) and chargepoint (www.chargepoint.com), the charging capacity of different CSs ranges from 4 to 32 and the estimated charging fee for a fully charged EV ranges from 40 to 90. Thus, we set p^{max} to 90. Considering different land prices in different regions, we set CSs' operating costs ϵ_j in regions with high land price (*i.e.,* Region 2 and 5 in NJ and NYC) to be a random value in $[60, 70]$ and set ϵ_j in other regions to be a random value in $[10, 30]$. By default, the weights in (1) are set as $\omega_p = 0.6$, $\omega_q = 0.1$, $\omega_d = 0.3$ by [23].

6.1.2. Pricing methods

We compare the methods SPITER, Smooth, ODCA, SPGCE, MUP, Static, Random and BF.

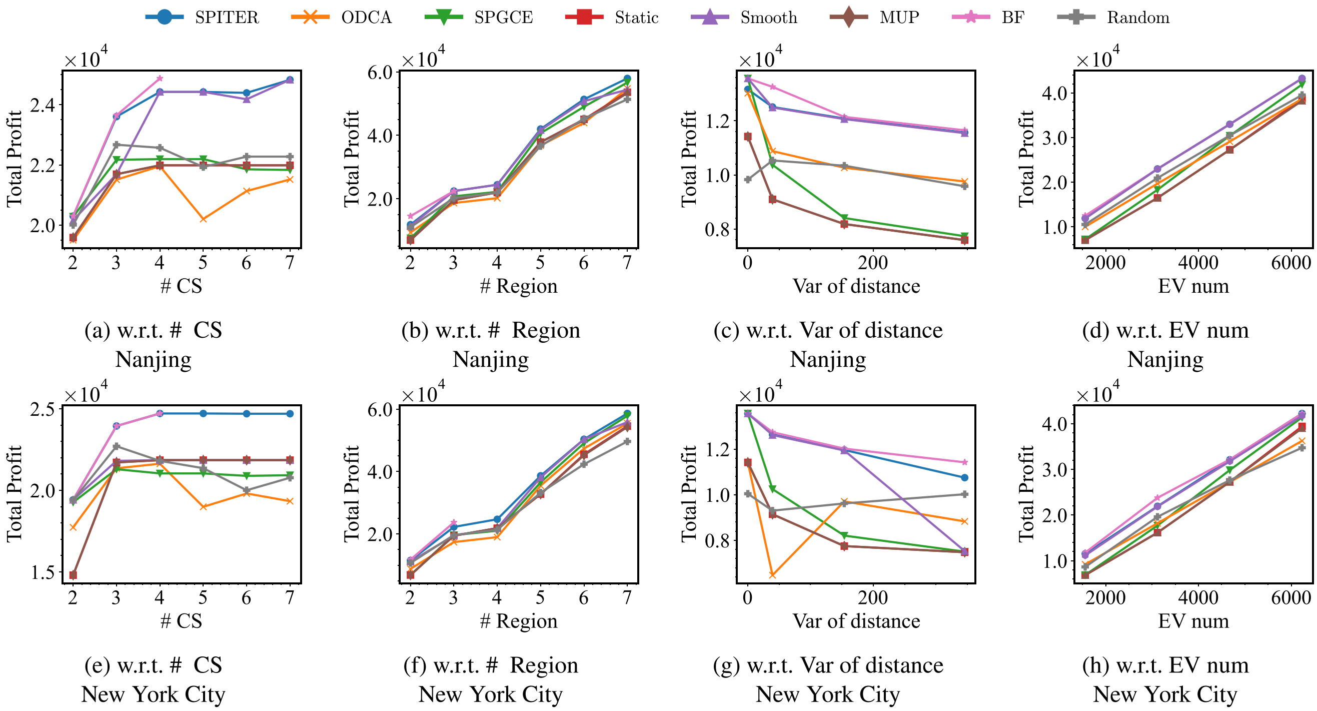
- SPITER computes the CSs' cooperative pricing strategies by Algorithm 3. It generates the initial pricing strategy using the Smooth algorithm. Prices can also be initialized using static methods, setting all prices to their maximum values. We refer to these two approaches as SPITER_smooth and SPITER_static respectively. Unless otherwise noted, the term SPITER throughout the following text represents the SPITER_smooth.
- Smooth [29] relaxes complementary constraints of Problem (10). It perturbs the original MPEC and considers a sequence of smooth and regular problems. Then, the relaxed problem can be solved by the off-the-shelf non-linear optimization techniques.
- ODCA (Optimal Demand Computation Algorithm) [10]. The method does not account for the interactions among followers. Thus, congestion effects are overlooked in the pricing strategy.
- SPGCE (Stackelberg Pricing Game with Congestion Effects) [24]. This approach calculates pricing strategies by considering the lower-level game's equilibrium as the Wardrop equilibrium. However, the distance factor is not considered in the method.
- MUP (MakeUp Pricing) [52] sets prices by the operating costs of CSs. Its pricing strategy is adding a markup on the operating costs of each CS. This pricing formula assigns prices to each CS in descending order based on cost, with a fixed price gap of 3 between each. The highest priced CS is set at a ceiling price of 90.
- Static is a method which sets all the CSs' prices to p^{max}. The static methods are commonly used in the real-life market.
- Random is a method that sets all the CSs' prices between p^{min} and p^{max} randomly. With the pricing strategy, it calculates the profit by solving the lower level equilibrium iteratively. Through 1000 iterations, it selects the strategy that yields the highest total profit.
- BF (Brutal Force) calculates the cooperative pricing strategies for CSs by transforming the lower-level game into KKT conditions. It enumerates and solve the problem (10) under every possible combination of the complementary constrains of KKT conditions. Best pricing strategy is selected from all possible solutions. It can obtain the optimal solution.

6.2. Total profit comparison

We compare the total profits of the methods SPITER, Smooth, ODCA, SPGCE, MUP, Static, Random and BF under different settings.
Changing Number of CSs. To compare the total profits of different methods under various numbers of CSs, we consider EVs in Regions 1 to 4 in NJ and gradually increase the number of CSs from 2 to 6. These CSs are sequentially placed at the center of each region, according to the region's numerical order. For instance, three CSs will be allocated successively to the centers of Region 1, Region 2, and Region 3. We conduct the experiment in NYC using same settings.

From Figs. 6(a) and 6(f), we can observe that SPITER always performs better than the other methods and performs within 0.1% of BF in Fig. 6(e). On average, in NJ, SPITER outperforms Smooth, ODCA, SPGCE, MUP, Static and Random by 2.7%, 15.2%, 10.0%, 10.9%, 10.9%, 8.9%, respectively. Similarly, in NYC, it surpasses these methods by 12.0%, 22.9%, 16.0%, 12.1%, 12.1%, 13.1%, respectively. When the number of CSs is 7 in NJ, SPITER still achieves the increase of profit, while other methods do not. This is because SPITER considers the distance of CS which is far from the Regions 1 to 4. As the number of CSs increases, SPITER can set lower prices for more remotely located charging stations to attract EV charging, thereby increasing total profit. This experiment demonstrates SPITER's effectiveness in leveraging additional CSs to enhance total profit.
Changing Number of Regions. To contrast the total profit of different methods across different number of regions, ranging from Region 1 to 5, we set a CS at the center of each region and consider the EVs in different number of regions from 2 to 7. In contrast to simply increasing the number of EVs in Regions, this experiment also introduces a more intricate distribution of EVs. The quantity of EVs within each region can be observed in Fig. 5. We conduct the experiment in NJ and NYC using the same settings.

From Figs. 6(b) and 6(f), we can see that the total profit of SPITER outperforms compared to the other methods and achieves an income within 0.001% of the optimal solution in Fig. 6(b). On average, in NJ, SPITER outperforms Smooth, ODCA, SPGCE, MUP, Static and Random by 3.2%, 11.0%, 3.6%, 10.5%, 10.2%, 13.1%, respectively. Similarly, in NYC, it surpasses these methods by 5.3%, 12.9%, 4.8%, 12.2%, 11.9%, 16.6%, respectively. The total profit of overall methods escalates in correlation with the number of regions. This trend is attributable to the inclusion of a larger number of EVs in the market. These simulations show that in scenarios with complex EV distributions and increasing EV quantities, SPITER consistently maintains a superior pricing strategy compared to other methods.
Changing Spatial Distribution. In order to explore the impact of different spatial distributions of CSs on total profit, we select Region 1 and 2 in NJ, and set 8 CSs around these regions. We adjust the positions of these CSs to alter the variance in distance between these regions and the CSs, without changing the mean distance. We conduct the experiment in NYC using same settings.



(a) w.r.t. # CS Nanjing

(b) w.r.t. # Region Nanjing

(c) w.r.t. Var of distance Nanjing

(d) w.r.t. EV num Nanjing

(e) w.r.t. # CS New York City

(f) w.r.t. # Region New York City

(g) w.r.t. Var of distance New York City

(h) w.r.t. EV num New York City

Fig. 6. Total Profit Comp. across Algo.

Fig. 6(c) shows the total profit of each method under different spatial distributions. We see that SPITER outperforms all methods and shows significant improvement. On average, in NJ, SPITER exceeds the performance of Smooth, ODCA, SPGCE, MUP, Static and Random by 8.7%, 43.9%, 29.8%, 47.2%, 47.2%, and 26.8%, respectively. Similarly, in NYC, it surpasses these methods by 3.5%, 15.9%, 28.8%, 46.0%, 46.0%, and 28.0%, respectively. These advancements are achieved because SPITER considers the spatial distribution of CSs. SPITER sets a lower charging price for the CSs that are far from Regions 1 and 2. This pricing strategy incentivizes EVs to opt for more distant CSs, thereby increasing total profit. This experiment demonstrates that SPITER achieves the highest total profit across different spatial distributions of CSs compared to other methods.

Changing Number of EVs. To explore the impact of the congestion effects caused by EVs on total profit, we set a CS at the center of Regions 1 to 6, labeled as CS1 to CS6, respectively, and consider the EVs in Regions 1 and 2 in NJ. We conduct the experiments in NYC using same settings.

In Fig. 6(d), it is evident that SPITER consistently outperforms other methods in both the NJ and NYC datasets. Specifically, in NJ, SPITER exceeds the performance of ODCA, SPGCE, MUP, Static and Random by 13.9%, 13.6%, 24.9%, 24.3%, and 9.5% respectively. In NYC, the margins are 18.5%, 12.5%, 20.7%, 20.2%, and 18.9%. The superior performance of SPITER is primarily due to its consideration of congestion effects, which is not accounted for in the compared methods. Additionally, an observed increase in total profits across all methods suggests that the integration of more EVs enhances market profitability. These findings underscore SPITER's effectiveness in achieving higher total profits through the strategic management of EVs, reaffirming its advantage over conventional methods by leveraging congestion insights.

6.3. Quality of price strategies

We also investigate the quality of price strategies, assessing the Price of Anarchy following equilibrium achieved by the SPITER in experiments and examining the costs of both CS manager and EVs.

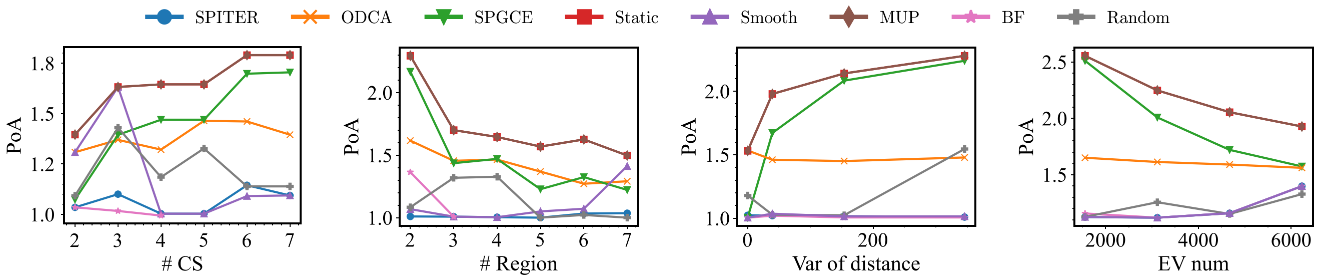
Price of Anarchy. We analyzed the social welfare of SPITER method compared to the social optimum with experiments under different settings. Common notion to measure the social efficiency is Price of Anarchy (PoA) [53], which is defined as the ratio of the social cost of equilibrium, calculated by pricing methods, and the social optimum which is defined as the minimum social cost achieved through collaboration between CSs and EVs.

Through the pricing strategies $\boldsymbol{p}$ calculated by methods, we can get the charging strategies of EVs $\widetilde{\boldsymbol{f}}$ in each region by which social cost can be calculated. Define the function of social cost $SC\left(\widetilde{\boldsymbol{f}}(\boldsymbol{p}), \boldsymbol{p}\right)$ as $\sum_{j=1}^{M} \epsilon_j \boldsymbol{f}_j + \sum_{j=1}^{m} (\omega_q Q_j + \omega_d d_{ij}) f_{ij}$ which is composed of CS cost and EV cost. SC measures operating cost of CSs and congestion and travel cost of EVs in regions. We show the result of the experiments by Fig. 7 which uses the same settings with Section 6.2.

Fig. 7(a) depicts the effects of CS number on PoA of each method in NJ. On average, SPITER attains a social cost that is within 6.3% of the optimal social cost in Fig. 7(a). On average, SPITER outperforms Smooth, ODCA, SPGCE, MUP, Static and Random by 10.5%, 23.3%, 27.5%, 35.5%, 35.5%, 12.7%, respectively.

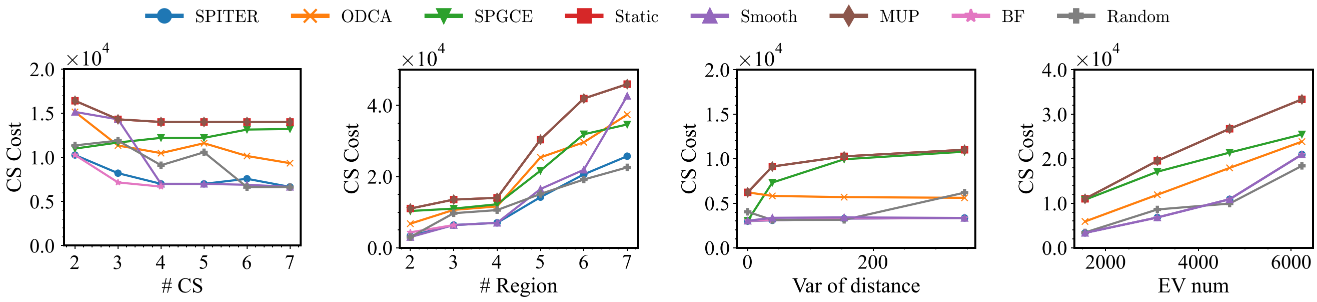
From Fig. 6(a), we see that the total profit of SPITER keeps growing with the number of CSs but the PoA of SPITER always fluctuates around 1. This is because SPITER considers the distance of CSs and congestion caused by EVs. SPITER strategically sets lower prices for CSs with lower operating costs that are farther away, thereby attracting electric vehicles (EVs) to these locations to alleviate congestion at high-operation-cost CSs in popular central areas. Through this approach, SPITER leverages the addition of new CSs, sustaining total profit growth while preserving a favorable PoA. This experiment demonstrates that SPITER can reduce CS costs through the addition of extra CSs and decrease congestion cost of EVs in core areas, thereby maintaining lower social costs.

CS Cost. We evaluate CS cost in the simulation to explore SPITER's ability of improving the benefits of different stakeholders in the urban IoT network. We also change the number of CSs, the number of regions, the spatial distribution of CSs and the number of EVs to explore the effects. The settings of the experiments is identical to the experiments



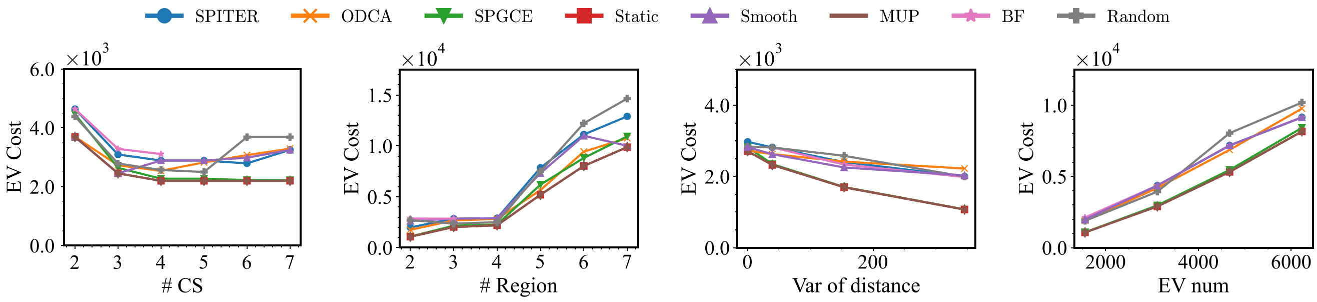
(a) w.r.t. # CS (b) w.r.t. # Region (c) w.r.t. # Var of distance (d) w.r.t. # EV num

Fig. 7. Price of Anarchy Comp. Across Algo. (Nanjing).



(a) w.r.t. # CS (b) w.r.t. # Region (c) w.r.t. Var of distance (d) w.r.t. # EV

Fig. 8. CS Cost Comp. (Nanjing).



(a) w.r.t. # CS (b) w.r.t. # Region (c) w.r.t. Var of distance (d) w.r.t. # EV

Fig. 9. EV Cost Comp. (Nanjing).

in Section 6.2. The results of the experiment are presented in Fig. 8.

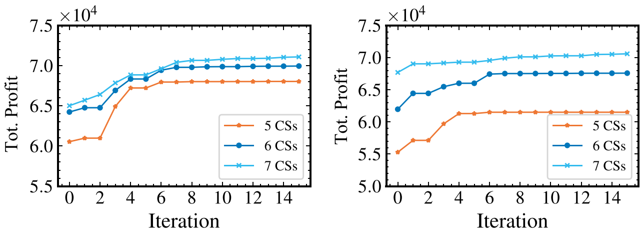
Fig. 7(b) shows the effects of the number of regions on the CS cost. The CS cost of SPITER maintains a social cost 1.5% within the optimal social cost. From Fig. 8(b), we see that SPITER achieves a reduction in CS cost by 20.7%, 28.1%, 31.2%, 41.1%, 41.1%, and 9.8% compared to the methods Smooth, ODCA, SPGCE, MUP, Static and Random, respectively, on average.

An increase in the number of regions implies a higher number of EVs requiring charging, leading to greater congestion. Additionally, the distribution of these EVs across various regions contributes to a more complex spatial distribution. Considering the distance and the congestion, SPITER addresses both issues by setting lower prices for CSs with lower operational costs, which are farther away, to attract EVs. Utilizing this strategy, SPITER successfully schedules EVs, consistently maintaining lower CS costs compared to other methods. The simulation shows the SPITER's ability of dealing with the increasing number of regions.

EV Cost. We also evaluate the EV cost in our experiments, which is composed of congestion cost and the distance cost. The experimental setup is consistent with those conducted in Section 6.2. The results of the experiments are shown in Fig. 9.

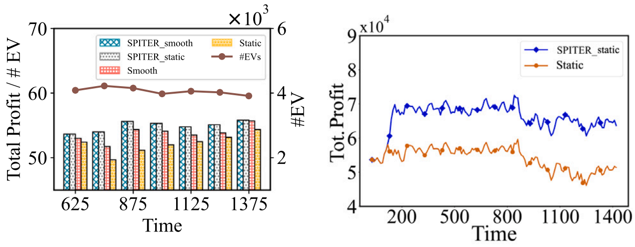
From Fig. 9(b), we observe a slight difference in the EV costs with SPITER compared to other methods. This divergence is attributed to SPITER's strategy of directing EVs to charge in more remote regions where CSs typically have lower operational costs, thereby generating increased profit. This approach reduces congestion costs but incurs higher travel costs. However, in the experimental setup, travel costs are weighted more heavily than congestion costs. Despite this, SPITER significantly reduces CS costs while keeping EV costs nearly unchanged. For instance, compared to the Smooth algorithm, SPITER's EV cost slightly increases by 0.48×10^4, but its CS cost significantly decreases by 3.36×10^4.

SPITER takes into account both congestion and distance factors, allowing EVs more charging options: EVs can choose to charge at more



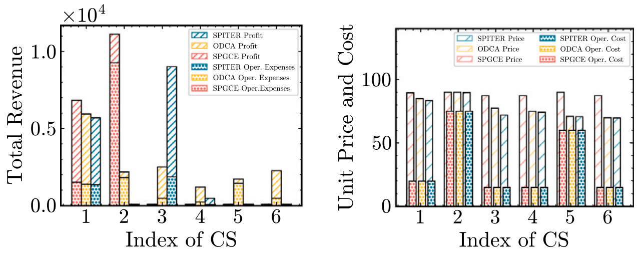
(a) Nanjing (b) New York City

Fig. 10. Algorithm Convergence.



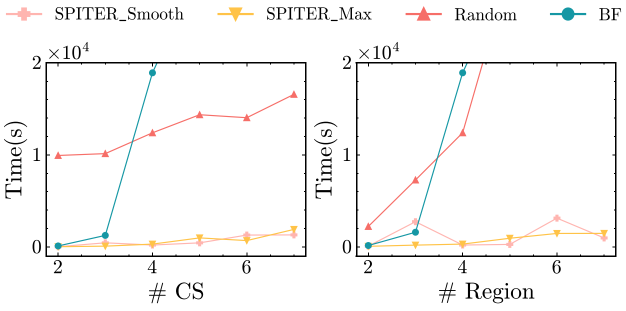
(a) Average profit (b) Convergence

Fig. 11. Performance on Dynamic Scenario.



(a) Profit and Oper. Expenses (b) Pricing Strategies

Fig. 12. Analysis on Individual CS (Nanjing).



(a) w.r.t. # CS (b) w.r.t. # Region

Fig. 13. Running Time.

distant CSs with lower prices, or opt for closer CSs with higher prices. In addition, when there are changes in the number or distribution of EVs, SPITER can schedule EVs by setting reasonable prices, thereby reducing congestion costs.

6.4. Analysis on individual CS

In the context of a business, revenue refers to the total amount of money generated from the sale of goods or services over a specific period of time. Thus, profits are the result of total revenue minus operating expenses. We explore how SPITER optimizes total profit and analyzes the impact of distance, congestion, and the operational costs of individual CSs on SPITER's pricing strategy. We further elaborate on our strategy by comparing ODCA and SPGCE.

We consider EVs in Regions 1 to 2 in NJ and set 6 CSs. These CSs are placed within 6 regions, with each CS located at the center of its respective region. We also carry out experiments focusing on EVs in Regions 1 and 2, adjusting the number of CSs from 2 to 6 with each CS located at the center of the region from 1 to 6. The experiment is designed to demonstrate that SPITER consistently achieves superior profits across various configurations.

The results of the experiments are shown in Fig. 12. Fig. 12(a) shows the total revenue for each CS, providing a clear view of the components of profits and operation expenses. Fig. 12(b) shows the price and operational cost of each CS in the experiment. The average distance to each CS can be calculated: $17.5, 17.5, 37.5, 42.5, 54.0, 52.5$.

Effects of Distance. CS 1 and CS 2 are closer to EVs compared to other CSs, making them more likely to be chosen. Fig. 12(b) illustrates that SPITER strategically lowers the charging prices for CS 3, 4, 5, 6, and 7, which are positioned further from Regions 1 and 2, making these prices significantly lower than those at CS 1 and CS 2. This pricing strategy ensures that EVs do not cluster at CSs 1 and 2, thus mitigating congestion. This effect is corroborated by Fig. 12(a). SPGCE overlooks distance as a contributing factor in its pricing assessment. The charging prices at CSs 3, 4, 5, and 6 are only 16.1% lower than those at CSs 1 and 2, which are deemed high.

Effects of Congestion. From Fig. 12(b), we can observe that SPITER sets the price for CS 3 lower than that of ODCA, resulting in the revenue from CS3 being 258.0% higher than that of ODCA. This is attributed to SPITER's consideration of congestion effects. By setting a lower price, it attracts more EVs, making it worthwhile to accommodate the congestion. ODCA, on the other hand, does not take congestion into account. Despite moderate pricing at CS 3, Fig. 12(a) shows that congestion still deters many EVs from choosing these stations for charging.

Effects of Unit Operating Costs. Except for the consideration of distance and congestion, operating cost of each CS also has effects on the pricing strategy of SPITER. We define unit price as the fee charged by a CS for providing electricity to an EV and unit cost as the cost incurred by a CS to supply electricity to an EV. Fig. 12(b) shows the operating costs of NJ and NYC, respectively. Due to the land prices, CS 1,3,4 and 6's operating costs are lower than CS 2 and CS 5. From Figs. 12(a) and 12(b), we observe that the price of CS 1 and CS 3 is low, which attracts more EVs. This strategic pricing, coupled with CS 3 and CS 1's low operating costs, significantly enhances the total profit.

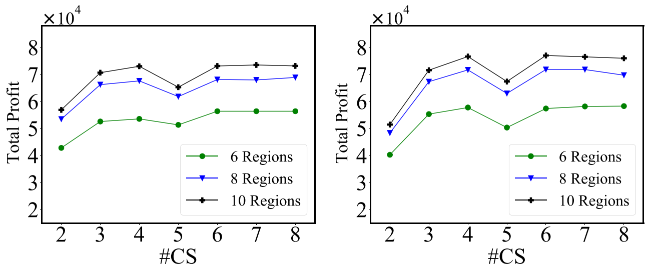
SPITER's proper pricing strategy stems from SPITER's integrated approach to considering both distance to CSs and congestion levels. The results show that SPITER can make pricing strategy according to the location of CSs and adjust the pricing strategy according to CS's operating cost and possible congestion to guide EV flows. The simulation demonstrates that SPITER absolute ability.

6.5. Computing performance

To assess the computing performance of SPITER, we conduct experiments on the convergence and the computation time of SPITER.

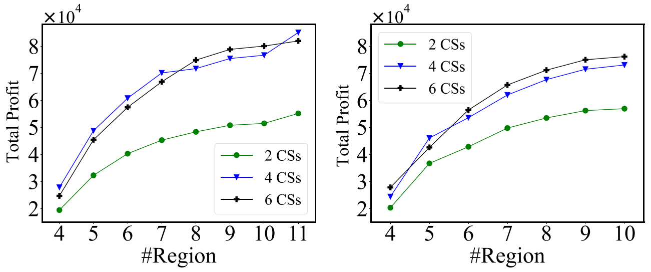
Convergence. We now evaluate the convergence of SPITER. Since the performance gap between SPITER_static and Static is more significant than that between SPITER_smooth and Smooth, we focus on the convergence of SPITER_static. In the experiments, we consider the EVs in Regions $1, 2, \ldots, 8$ in both cities. In each iteration, a CS updates its pricing strategy by solving the corresponding subproblem. In particular, at iteration k, CS k updates its strategy.

Figs. 10(a) and 10(b) show the results of algorithm convergence. We can observe that the total profit in NJ increases significantly at iterations 3 and 4. This is because CS 3 and CS 4 in NJ surround CS 2,



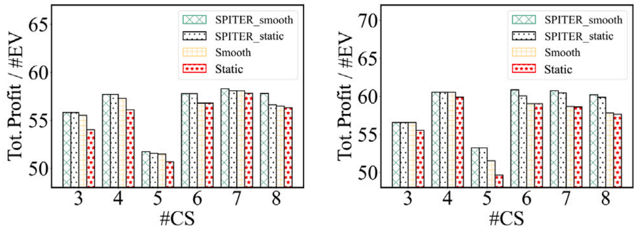
(a) Nanjing (b) New York City

Fig. 14. Effects of # CS.



(a) Nanjing (b) New York City

Fig. 15. Effects of # Region.



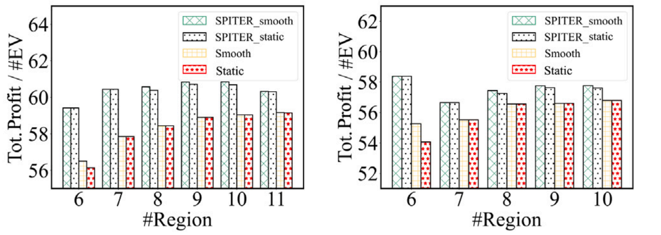
(a) Nanjing (b) New York City

Fig. 16. Initialization Methods Comp. w.r.t. # CS.

and their operating costs are significantly lower than CS 2's. When CS 3 and CS 4 update the pricing strategy, they decrease their prices, so EVs will prefer CS 3 and CS 4 rather than CS 2, which improves the total profit. Similar situations can be found in NYC, where the total profit increases significantly at iterations 1, 3, and 4. Additionally, when there are 5, 6, and 7 CSs, SPITER_static in NJ (NYC) converges after 6, 7, and 7 (4, 6, and 7) iterations, respectively. This implies that when all CSs have updated their pricing strategies, the algorithm will converge.

Running Time. We conduct a comparison of the computational times for the Best, Random, and SPITER methods, examining how the number of regions and CSs affects the computational efficiency of SPITER. The experiments set 4 CSs located in Region $1, 2, 3, 4$ and vary the number of regions from 2 to 7 and analyzed the impact on SPITER's performance. On the other side, the experiments also consider the EVs in regions 1 to 4 within NJ, modifying the number of CSs between 2 to 7 to understand the effects on running time.

Observations from Figs. 13(a) and 13(b) indicate that SPITER's running time does not significantly increase with either the number of CSs or regions. SPITER outperforms both the Random and Best methods in running time. This enhanced performance is attributed to SPITER's utilization of the Smooth method for initializing pricing strategies, which remains relatively unaffected by changes in the number of CSs and regions. The Best solution, determined through exhaustive search, becomes impractical for more than 4 CSs or 4 regions due to prohibitive computational demands. The Random method, on the other hand, iterates over random pricing strategies to find a lower equilibrium, leading to rapid increases in computational time with more regions. Fig. 13 provides a clear visualization of this characteristic. SPITER required minimal time yet achieved the total profit very close to the optimal solution. For example, Fig. 6(b) shows that SPITER can achieve the total profit within 0.001% of the optimal solution while only requiring 1% of the computational time on average, showcasing its exceptional computational efficiency.



(a) Nanjing (b) New York City

Fig. 17. Initialization Methods Comp. w.r.t. # Region.

6.6. Performance on dynamic scenario

We evaluate the effectiveness and convergence of SPITER in a distributed and dynamic scenario. The evaluation utilizes the NJ dataset, covering Regions 1 to 4, with two CSs located in Region 1 and Region 2, respectively. The parameters for SPITER are set as shown in Table 2. To simulate the dynamic scenario, a discrete-event system is developed, as introduced in Section 5.2. This system operates with an average of 4000 EVs. To emulate urban dynamics, the number of EVs varies over time. The CSs update their strategies at intervals of 120 time slices to adapt to the urban dynamics.

We use the average profit per EV to measure the effectiveness and stability of our game model in the dynamic scenario. Fig. 11(a) shows the results of SPITER's variants when charging demands change over time. The market profits of SPITER_smooth and SPITER_static are almost the same and higher than those of Smooth and Static. These results are consistent with the static scenarios. Moreover, we observe that the average profits generated by SPITER are stable, ranging from 53 to 55, while the average profits generated by Static vary from 49 to 54. This demonstrates the effectiveness of the dynamic pricing strategies employed by SPITER.

We demonstrate the convergence of the distributed SPITER in a dynamic scenario. We decrease the EV charging demand for each region at the 800th time slice to verify if the distributed SPITER can converge. Fig. 11(b) illustrates that the distributed SPITER initially converges at the 200th time slice. When the EV charging demand decreases at the 800th time slice, the profit drops immediately but converges again after another 200 time slices. In both cases, the market profit of the distributed SPITER is 20% larger than Static. These results showcase the effectiveness and convergence of the distributed SPITER.

6.7. Effects of parameter settings

We conduct simulations to evaluate the effects of the number of CSs and EVs and initialization methods on the total profit. Different from the experiments in which SPITER is compared to other methods, more CSs and Regions are selected in this simulation to explore the effects.

Number of CSs. We change the number of CS from 2 to 8 and evaluate its effects on the total profit. From Figs. 14(a) and 14(b), we observe that when the number of CSs increases, the market profit nearly remains stable when the number of regions is fixed. The stable profit stems from the fact that the market profit is determined by the number of EVs and CSs' profit per EV. Most CSs (*i.e.,* all CSs except CS 2 and CS 5) have similar charging price and operating cost, hence if the total number of EVs remains unchanged, the market profit will be stable. Regarding the two special cases, it is because the operating costs of CS 2 (*i.e.,* price 69 in NJ and price 67 in NYC) and CS 5 (*i.e.,* price 65 in NJ and price 64 in NYC) are higher than the average operating costs of other CSs, so the total profit decreases when CS 2 and CS 5 are added to the market.

Number of Regions. We change the number of regions from 4 to 11 and evaluate its effects on the total profit. From Figs. 15(a) and 15(b), we observe that when the number of regions increases, the market profit also increases, because more regions usually contain more EVs which improve the market profit. We also see from Figs. 15(a) and 15(b) that the number of CSs affects the market profit. Because the CSs' operating costs are different, EV flows' decisions affect the market profit. For example, considering the market in NJ, the profit improves significantly when CS 3 and CS 4 are added to the market due to their high profits compared. In this way EVs that are close to CS 3 or CS 4 are more likely to choose CS 3 or CS 4 rather than CS 2, which improves market profit.

Initialization Methods. We then compare SPITER with other pricing methods in terms of the EVs' average profit (*i.e.,* total profit/#EVs). We use EVs' average profit because it is a easier way to compare these three method. We first change the number of charging stations and select Regions $1, 2, \ldots, 10$. From Figs. 16(a) and 16(b), we see that SPITER_smooth always performs the best. On average, in NJ (NYC) SPITER_smooth is 0.05%, 1.1%, and 2.3% (0.03%, 1.29%, and 2.5%) better than SPITER_static, Smooth, and Static, respectively. Moreover, in NYC, SPITER_smooth is 6.4% better than Static when CSs 1 to 5 are selected. Since Smooth is better than Static, SPITER_smooth converges to better local maximum points than SPITER_static. This demonstrates that the better initial pricing strategies may lead to better final strategies (see Fig. 16 and Fig. 17).

7. Conclusion

In this paper, we proposed a congestion-aware Stackelberg pricing game in urban IoT networks and analyzed the existence and uniqueness of the followers' equilibrium with the followers' spatial distribution in urban environment. To solve the problem, we reformulated the hierarchical game as an MPEC. Then, we proposed the SPITER algorithm to sovle the MPEC, and proved SPITER's convergence. To enhance the efficiency of solving SPITER in urban environments, we also developed optimization techniques. In the end, we evaluated SPITER's performance with a case study on EV charging network using two real-life urban datasets. In the future, we will investigate the equilibrium among different leaders, assuming that each leader can adjust pricing strategies.

CRediT authorship contribution statement

Jiahui Jin: Conceptualization, Methodology, Writing – original draft. **Zhendong Guo:** Conceptualization, Methodology, Writing – original draft. **Wenchao Bai:** Conceptualization, Methodology, Writing – original draft. **Biwei Wu:** Conceptualization, Methodology, Writing – original draft. **Xiang Liu:** Conceptualization, Writing – original draft. **Weiwei Wu:** Conceptualization, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Patrick Briest, Martin Hoefer, Piotr Krysta, Stackelberg network pricing games, Algorithmica 62 (3–4) (2012) 733–753.

[2] Xin Kang, Rui Zhang, Mehul Motani, Price-based resource allocation for spectrum-sharing femtocell networks: a Stackelberg game approach, IEEE J. Sel. Areas Commun. 30 (3) (2012) 538–549.

[3] Yuan Wu, Tengyi Zhang, Danny H.K. Tsang, Joint pricing and power allocation for dynamic spectrum access networks with Stackelberg game model, IEEE Trans. Wireless Commun. 10 (1) (2011) 12–19.

[4] Nicole Sawyer, David B. Smith, Flexible resource allocation in device-to-device communications using Stackelberg game theory, IEEE Trans. Commun. 67 (1) (2019) 653–667.

[5] Beibei Wang, Zhu Han, K.J. Ray Liu, Distributed relay selection and power control for multiuser cooperative communication networks using Stackelberg game, IEEE Trans. Mob. Comput. 8 (7) (2009) 975–990.

[6] Shuan He, Wei Wang, Context-aware QoE-price equilibrium for wireless multimedia relay communications using Stackelberg game, in: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, 2017, pp. 506–511.

[7] Sungwook Kim, Multi-leader multi-follower Stackelberg model for cognitive radio spectrum sharing scheme, Comput. Netw. 56 (17) (2012) 3682–3692.

[8] Leng-Gan Yi, Yi-Min Lu, Utility-driven relay for hybrid access femtocells based on cognitive radio spectrum auction, Comput. Netw. 80 (2015) 155–166.

[9] Valerio Di Valerio, Valeria Cardellini, Francesco Lo Presti, Optimal pricing and service provisioning strategies in cloud systems: A Stackelberg game approach, in: 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, 2013, pp. 115–122.

[10] Yifan Chen, Zhiyong Li, Bo Yang, Ke Nai, Keqin Li, A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing, Future Gener. Comput. Syst. 108 (2020) 273–287.

[11] Abhishek Hazra, Mainak Adhikari, Tarachand Amgoth, Satish Narayana Srirama, Stackelberg game for service deployment of IoT-enabled applications in 6G-aware fog networks, IEEE Internet Things J. 8 (7) (2021) 5185–5193.

[12] Chaogang Tang, Huaming Wu, Optimal computational resource pricing in vehicular edge computing: A stackelberg game approach, J. Syst. Archit. 121 (2021) 102331.

[13] Kaiyuan Zhang, Xiaolin Gui, Dewang Ren, Tianjiao Du, Xin He, Optimal pricing-based computation offloading and resource allocation for blockchain-enabled beyond 5G networks, Comput. Netw. 203 (2022) 108674.

[14] Neetu Raveendran, Huaqing Zhang, Lingyang Song, Li-Chun Wang, Choong Seon Hong, Zhu Han, Pricing and resource allocation optimization for IoT fog computing and NFV: An EPEC and matching based perspective, IEEE Trans. Mob. Comput. 21 (4) (2022) 1349–1361.

[15] F. Busacca, L. Galluccio, S. Palazzo, A marketplace model for drone-assisted edge computing in 5G scenarios, Comput. Netw. 219 (2022) 109453.

[16] Zhiyuan Liu, Qiang Meng, Shuaian Wang, Speed-based toll design for cordon-based congestion pricing scheme, Transp. Res. C 31 (2013) 83–98.

[17] Byung-Wook Wie, Dynamic Stackelberg equilibrium congestion pricing, Transp. Res. C 15 (3) (2007) 154–174.

[18] Zhaoyang Lu, Huijun Sun, Jianjun Wu, Road pricing under mixed equilibrium behaviors on urban congested networks, in: 2011 Fourth International Joint Conference on Computational Sciences and Optimization, Kunming and Lijiang City, China, 2011, pp. 1095–1099.

[19] Kai Ma, Xiaoyan Hu, Jie Yang, Zhiyue Yue, Bo Yang, Zhixin Liu, Xinping Guan, Electric taxi charging strategy based on Stackelberg game considering hotspot information, IEEE Trans. Veh. Technol. 71 (3) (2022) 2427–2436.

[20] Wei Yuan, Jianwei Huang, Ying Jun Zhang, Competitive charging station pricing for plug-in electric vehicles, IEEE Trans. Smart Grid (2015) 1–13.

[21] Tengteng Ma, Yong Zhang, Siyu Yuan, Zhenjie Cheng, Cognitive RAN slicing resource allocation based on Stackelberg game, China Commun. 19 (5) (2022) 12–23.

[22] Shuying Lai, Jing Qiu, Yuechuan Tao, Junhua Zhao, Pricing for electric vehicle charging stations based on the responsiveness of demand, IEEE Trans. Smart Grid 14 (1) (2023) 530–544.

[23] Yanhai Xiong, Jiarui Gan, Bo An, Chunyan Miao, Yeng Chai Soh, Optimal pricing for efficient electric vehicle charging station management, in: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, AAMAS '16, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2016, pp. 749–757.

[24] Tobias Harks, Anja Schedel, Stackelberg pricing games with congestion effects, Math. Program. (2021).

[25] Toni Böhnlein, Stefan Kratsch, Oliver Schaudt, Revenue maximization in Stackelberg pricing games: Beyond the combinatorial setting, Math. Program. 187 (1–2) (2021) 653–695.

[26] Yan Cui, Zechun Hu, Haocheng Luo, Optimal day-ahead charging and frequency reserve scheduling of electric vehicles considering the regulation signal uncertainty, IEEE Trans. Ind. Appl. 56 (5) (2020) 5824–5835.

[27] Yiwei Wu, Masoud Barati, Gino J. Lim, A pool strategy of microgrid in power distribution electricity market, IEEE Trans. Power Syst. 35 (1) (2020) 3–12.

[28] Hossein Fotouhi, Elise Miller-Hooks, Optimal time-differentiated pricing for a competitive mixed traditional and crowdsourced event parking market, Transp. Res. C 132 (2021) 103409.

[29] Zhe Liu, Fan Wu, Sajal K. Das (Eds.), Wireless Algorithms, Systems, and Applications: 16th International Conference, WASA 2021, Nanjing, China, June 25–27, 2021, Proceedings, Part II, Lecture Notes in Computer Science, vol. 12938, Springer International Publishing, 2021.

[30] Rui Li, Wei Wei, Shengwei Mei, Qinran Hu, Qiuwei Wu, Participation of an energy hub in electricity and heat distribution markets: An MPEC approach, IEEE Trans. Smart Grid 10 (4) (2019) 3641–3653.

[31] Joe Naoum-Sawaya, Samir Elhedhli, Controlled predatory pricing in a multiperiod Stackelberg game: An MPEC approach, J. Global Optim. 50 (2) (2011) 345–362.

[32] Patrick Briest, Martin Hoefer, Piotr Krysta, Stackelberg network pricing games, 2008.

[33] Joseph N. Prashker, Shlomo Bekhor, Route choice models used in the stochastic user equilibrium problem: A review, Transp. Rev. 24 (4) (2004) 437–463.

[34] Gordon Owusu Boateng, Guolin Sun, Daniel Ayepah Mensah, Daniel Mawunyo Doe, Ruijie Ou, Guisong Liu, Consortium blockchain-based spectrum trading for network slicing in 5G RAN: A multi-agent deep reinforcement learning approach, IEEE Trans. Mob. Comput. 22 (10) (2023-10-01) 5801–5815.

[35] Chaitanya Swamy, The effectiveness of stackelberg strategies and tolls for network congestion games, ACM Trans. Algorithms 8 (4) (2012) 1–19.

[36] Duong Tung Nguyen, Hieu Trung Nguyen, Long Bao Le, Dynamic pricing design for demand response integration in power distribution networks, IEEE Trans. Power Syst. 31 (5) (2016) 3457–3472.

[37] Woongsup Lee, Robert Schober, Vincent W.S. Wong, An analysis of price competition in heterogeneous electric vehicle charging stations, IEEE Trans. Smart Grid 10 (4) (2019) 3990–4002.

[38] Zhi-Quan Luo, Jong-Shi Pang, Daniel Ralph, Piecewise sequential quadratic programming for mathematical programs with nonlinear complementarity constraints, in: Athanasios Migdalas, Panos M. Pardalos, Peter Värbrand (Eds.), Multilevel Optimization: Algorithms and Applications, Springer US, Boston, MA, 1998, pp. 209–229.

[39] Sung-Guk Yoon, Young-June Choi, Jong-Keun Park, Saewoong Bahk, Stackelberg-game-based demand response for at-home electric vehicle charging, IEEE Trans. Veh. Technol. 65 (6) (2016) 4172–4184.

[40] Aurobinda Laha, Bo Yin, Yu Cheng, Lin X. Cai, Yu Wang, Game theory based charging solution for networked electric vehicles: a location-aware approach, IEEE Trans. Veh. Technol. 68 (7) (2019) 6352–6364.

[41] Gagangeet Singh Aujla, Neeraj Kumar, Mukesh Singh, Albert Y. Zomaya, Energy trading with dynamic pricing for electric vehicles in a smart city environment, J. Parallel Distrib. Comput. 127 (2019) 169–183.

[42] Francesco Malandrino, Claudio Casetti, Carla-Fabiana Chiasserini, Massimo Reineri, A game-theory analysis of charging stations selection by EV drivers, Perform. Eval. 83–84 (2015) 16–31.

[43] Yanhai Xiong, Bo An, Sarit Kraus, Electric vehicle charging strategy study and the application on charging station placement, Auton. Agents Multi-Agent Syst. 35 (1) (2021) 3.

[44] Wayes Tushar, Walid Saad, H. Vincent Poor, David B. Smith, Economics of electric vehicle charging: A game theoretic approach, IEEE Trans. Smart Grid 3 (4) (2012) 1767–1778.

[45] Shuying Lai, Jing Qiu, Yuechuan Tao, Junhua Zhao, Pricing for electric vehicle charging stations based on the responsiveness of demand, IEEE Trans. Smart Grid 14 (1) (2023-01) 530–544.

[46] Z.-Q. Luo, J.-S. Pang, D. Ralph, Mathematical Programs with Equilibrium Constraints, Cambridge Univ. Press, Cambridge, U.K., 1996.

[47] Paul Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, J. Optim. Theory Appl. 109 (3) (2001) 475–494.

[48] J.B. Rosen, Existence and uniqueness of equilibrium points for concave n-person games, Econometrica 33 (1964) 520–534.

[49] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge Univ. Press, 2004.

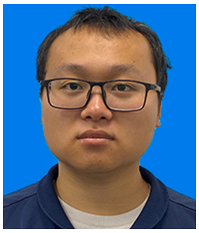
[50] Biwei Wu, Xiaoxuan Zhu, Xiang Liu, Jiahui Jin, Runqun Xiong, Weiwei Wu, Revenue maximization of electric vehicle charging services with hierarchical game, in: Wireless Algorithms, Systems, and Applications - 16th International Conference, Nanjing, China, June 25-27, 2021, 2021, pp. 417–429.

[51] F. Facchinei, H. Jiang, L. Qi, A smoothing method for mathematical programs with equilibrium constraints, Math. Program. 85 (1) (1999) 107–134.

[52] Tanya Sammut-Bonnici, Derek F. Channon, Pricing strategy, in: Cary L. Cooper (Ed.), Wiley Encyclopedia of Management, first ed., Wiley, 2015-01-22, pp. 1–3.

[53] Tobias Harks, Stackelberg strategies and collusion in network games with splittable flow, Theory Comput. Syst. 48 (4) (2011) 781–802.

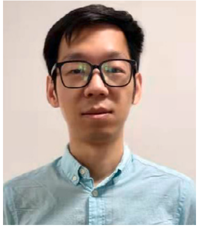
Jiahui Jin is an associate professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. He received his Ph.D. degree in computer science from Southeast University in 2015. He had been a visiting Ph.D. student at University of Massachusetts, Amherst, U.S., during August 2012 to August 2014. His current research interests include IoT network and urban computing.



Zhendong Guo received the B.S degree in Software College of Southeast University, Nanjing, China, in 2021. Currently, he is working toward the M.S. degree in software engineering at Southeast University, Nanjing, China. His current research interests include urban IoT networks and game theory.

Wenchao Bai is working toward the Ph.D. degree in computer science and engineering at Southeast University, Nanjing, China. His current research interests include urban computing and graph computing system.

Biwei Wu received the M.S degree with the School of Computer Science and Engineering, Southeast University. His current research interests include urban computing.



Xiang Liu received the Ph.D. degree with the School of Computer Science and Engineering, Southeast University. He has published several articles in refereed international conference proceedings, such as International Joint Conference on Artificial Intelligence, International Conference on Autonomous Agents and Multi-agent Systems. His main research interests include algorithmic game theory, mechanism design, and reinforcement learning.

Weiwei Wu received the B.Sc. degree from the South China University of Technology and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong (CityU), and University of Science and Technology of China (USTC) in 2011. He is currently a Professor with the School of Computer Science and Engineering, Southeast University, China. He went to Nanyang Technological University (NTU, Mathematical Division, Singapore, for post-doctoral research in 2012. He has published over 50 peer-reviewed papers in international conferences/journals, and serves as TPCs and reviewers for several top international journals and conferences. His research interests include optimizations and algorithm analysis, wireless communications, crowdsourcing, cloud computing, reinforcement learning, game theory, and network economics.