Update Scheduling for ADMM-based Energy Sharing in Virtual Power Plants Considering Massive Prosumer Access

Cheng Feng, *Student Member, IEEE,* Kedi Zheng, *Member, IEEE,* Yangze Zhou, *Student Member, IEEE,* Peter Palensky, *Senior Member, IEEE,* Qixin Chen, *Senior Member, IEEE*

***Abstract*—With the proliferation of distributed energy resources (DERs), electricity consumers in virtual power plants (VPPs) are transitioning into prosumers and are encouraged to share surplus energy with peers. Nevertheless, large-scale energy sharing among thousands of prosumers may encounter communication-related challenges. Communication network congestion may result in a significant increase in the negotiation waiting time to reach a sharing agreement, and potentially risks exceeding the deadline of negotiation before the market gate closes, rendering energy sharing ineffective. This paper proposes an online partial-update algorithm for the alternating direction method of multipliers (ADMM)-based energy sharing. By restricting the update connection between the VPP and the prosumers, the algorithm selects a subset of the prosumers participating in ADMM updates each round, hence eliminating the excessively long waiting time caused by communication congestion. Considering the delay induced by massive prosumer communication access requests, a method for determining the optimal number of prosumers participating in updates is provided. To fully utilize the limited update opportunities, a fair and efficient prosumer update scheduling policy is designed. The VPP schedules the participation of prosumers in updates such that the convergence-critical prosumers receive higher priority, yet every prosumer is granted sufficient update opportunities. Additionally, the extra computation and communication overheads brought by the prosumer scheduling are minimized, allowing the whole algorithm to be executed in real time. Numerical studies are conducted to validate the effectiveness of the algorithm and its performance in reducing the overall convergence time.**

***Index Terms*—Energy Sharing, Virtual Power Plant, ADMM, Machine type Communications, Access Delay, Update Scheduling, Distributed Energy Resource.**

I. Introduction

With the distributed energy resources (DERs) continuously integrated into the distribution power system, massive traditional electricity consumers are transforming into proactive prosumers [1]. Due to their small scales, it is anticipated that the prosumers would be aggregated by the virtual power plant (VPP), which is immediately seen by the transmission network operators [2]. Traditionally, prosumers are only permitted to sell their surplus power to the VPP at a low feed-in tariff, which inhibits the full utilization of DERs' energy. Nowadays, prosumers are encouraged to share their excess energy with their peers. With a coordinated energy sharing mechanism, the prosumers and the VPP can all achieve higher profitability [3].

Prosumers are different stakeholders who make individual decisions and possess private information [4]. It is difficult for the VPP to obtain personal data (such as DERs' capabilities) to perform centralized energy sharing. Consequently, energy sharing is often implemented in a distributed scheme with the assistance of a pool-based platform. Under a distributed sharing scheme, the prosumers and the VPP can only exchange iterative updates about the quantities and prices of sharing energy until an agreement is established on the sharing plan [5]. This approach is equivalently treated as the distributed solution procedure of a welfare optimization problem. Among the distributed solution methods, the alternating direction method of multipliers (ADMM) algorithm is the most popular method due to its robust convergence performance [6].

Nevertheless, with thousands of prosumers engaged in energy sharing, communication-related issues may arise. For the standard ADMM algorithm, the VPP must wait until it receives update data from *all the prosumers*. Due to the congestion of the wireless access communication network, the waiting time may be extremely long. It may result in a significant increase in the convergence time and potentially risk exceeding the deadline of the negotiation/coordination time before the algorithm reaches a convergence, rendering energy sharing ineffective [7]. Thereby, it is necessary to develop a novel ADMM algorithm, where the connection between the VPP and the prosumers is limited, and only a subset of prosumers is involved in updates each round. As a coordinator, the VPP should evaluate how to fully utilize the limited update opportunities by scheduling updates for prosumers. The prosumers, who are important to accelerating the convergence rate, should be scheduled more frequently, yet every prosumer should be granted sufficient opportunities to conduct updates. In this manner, the partial update ADMM algorithm can achieve efficiency and fairness simultaneously.

Recent works have recognized the critical role of the VPP in energy systems. Relevant research focused on VPP bidding [8], VPP dispatch [9], VPP formation [10], and

Manuscript received 23 September 2022; revised 24 December 2022; accepted 2 February 2023. This work is supported by the National Key R&D Program of China under Grant 2021YFB2401200, the National Natural and Science Foundation of China under Grant U2066205, and the International (NSFC-NWO) Joint Research Project of National Natural Science Foundation of China under Grant 52161135201. (*Corresponding author: Qixin Chen*, E-mail: qxchen@tsinghua.edu.cn).

C. Feng, K. Zheng and Q. Chen are with the State Key Lab of Power Systems, the Department of Electrical Engineering, Tsinghua University, Beijing 100084, China. Y. Zhou is with the Energy Engineering College, Zhejiang University, Hangzhou 310027, Zhejiang, China. P. Palensky is with the Intelligent Electrical Power Grids (IEPG) Group, Electrical Sustainable Energy (ESE), Electrical Engineering, Mathematics & Computer Science (EEMCS), Delft University of Technology, 2600 GA Delft, Netherlands.

VPP dis-aggregation approaches [11]. Among them, the VPP dispatch problem is the closest topic, and several attempts have been made to address communication-related issues in VPP dispatch. Ref. [12] designed a distributed economic dispatch method in sparse communication networks. Ref. [13] presented an attack-robust distributed economic dispatch strategy for possible cyber-attacks. Ref. [14] dealt with missing or delayed information by predicting them based on the autoregressive integrated moving average model. Conventional VPP dispatch strategies organize the DERs in a VPP-centric manner rather than prosumer-centric energy sharing.

The energy sharing problem has been widely studied along with peer-to-peer (P2P) trading or transactive energy, which is jointly referred as energy sharing in this paper [6]. The primary focus was to design efficient sharing algorithms for different schemes, including leader-follower, distributed, fully decentralized, hybrid, and grand coalition schemes. Ref. [15] treated the energy sharing mechanism as the leader-follower game where prosumers respond to the aggregator's price signal. Ref. [5] designed a sharing mechanism where prosumers play a distributed generalized Nash game in the pool-based platform. Ref. [16] proposed energy sharing among prosumers in the form of fully decentralized bilateral trading. Ref. [17] investigated that energy sharing is achieved when both individual prosumers and the prosumer coalition coexist. Ref. [18] treated prosumers as a grand coalition and designed the profit benefit allocation scheme using the cooperation game theory. Compared with pure P2P sharing among only prosumers, energy sharing in the VPP should also incorporate the VPP's benefit. Ref. [6] systematically reviewed methods to take the community's or the aggregator's benefit into consideration and presented a unified math model. Ref. [3] included the benefit of the system coordinator in energy sharing through Nash bargaining. Ref. [19] further incorporated the benefit of the distribution network operator and the physical network constraints into energy sharing. Most of the research mentioned above used the ADMM algorithm.

Communication-related issues in energy sharing have become a growing concern. Ref. [20] summarized the information-related issues in energy sharing, including the need for asynchronous computing and privacy-preserving algorithms. Ref. [21] borrowed the idea of node coloring to reduce the communication of P2P trading. Ref. [22] conducted experiments on the effects of communication connectivity sparsity on convergence optimality. Ref. [23] conducted experiments on how P2P trading algorithms respond to a larger number of prosumers and the presence of asynchronicity. It highlighted the need for the sparsification of negotiation processes. Ref. [7] designed an asynchronous ADMM algorithm for fully decentralized energy sharing, where prosumers could freely trade without waiting for idle or inactive neighboring agents. Most research focused on fully decentralized P2P energy sharing without considering the benefit of the coordinator. Besides, it is also critical for the VPP to actively schedule prosumers to participate in updates rather than passively waiting for them.

To summarize, this paper makes the following contributions:

1) Develop a partial-update ADMM algorithm for energy sharing in the VPP, which only requires a subset of the prosumers to involve in distributed iterations every round. The number of prosumers participating in updates is deliberately restricted to avoid communication congestion caused by massive update requests, so the overall convergence time is minimized.
2) Design a fair and efficient scheduling policy to determine which prosumers will participate in updates given limited update opportunities. The policy identifies convergence-critical prosumers who are important to accelerate the convergence rate and offers them a higher update priority. Meanwhile, each prosumer is granted enough opportunities to conduct updates to promote fairness.
3) Investigate techniques to enhance the online performance of the complete partial-update ADMM algorithm with scheduling. The extra complexity induced by prosumer scheduling, including the additional computation and communication time, is reduced as much as possible to eliminate its impact on the convergence time.

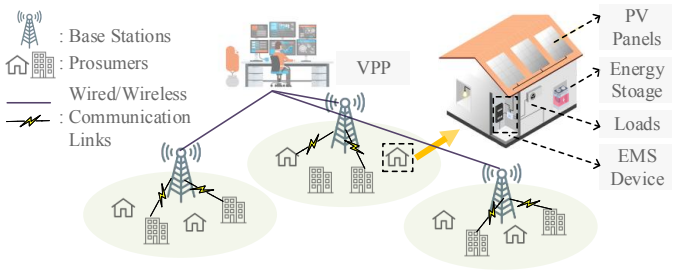


Fig. 1. The system structure of partial-update energy sharing in the VPP.

The remaining sections are organized as follows: Section II describes the setting of energy sharing in the VPP and formulates the social welfare maximization problem in energy sharing. Section III presents the standard ADMM algorithm, and proposes the partial-update ADMM algorithm to overcome the shortcomings of the standard ADMM algorithm. Section IV provides the method to determine the update set size of prosumers, formulates the scheduling policy, and completes the online partial update ADMM with the online and fair scheduling policy. Section V conducts case studies to verify the effectiveness of the proposed method. Section VI draws conclusions and gives future research directions.

Notations: Bold italic $\boldsymbol{x}$ denote vectors (column vectors). $\boldsymbol{x}^\top$ denotes $\boldsymbol{x}$'s transpose. Bold verticals $\mathbf{X}$ denote matrices. $[x]^+$ and $[x]^-$ denote $\max(0,x)$ and $\min(0,x)$, respectively. $[x]_b^a$ denotes $\min(a,\max(b,x))$ where $a>b$. $\lceil a \rceil$ denotes the smallest integer larger than a. $\|\boldsymbol{x}\|$ denotes the L2-norm. $|\mathcal{A}|$ is the cardinality of the set $\mathcal{A}$. $\mathcal{A}^c$ is $\mathcal{A}$'s complement set. $\emptyset$ is the empty set. ∇ is the gradient operator. ∂ is the subgradient operator. ∇^2_{xy} denotes the second-order partial derivatives with respect to variables x and y.

II. Energy Sharing Problem Formulation

The system consists of the prosumers $i = 1, ..., I$ and the VPP coordinator (VPP, for short), as shown in Fig.1. Each prosumer i has DERs, including load equipment, rooftop PV panels, and energy storage. They are equipped with an energy management system (EMS) for monitoring, trading,

and billing. This study considers the situation where energy sharing is arranged in the pool-based form in the day-ahead market, where $t = 1, ..., T$ trading periods are considered. For communications, cellular-based machine-type communications (MTCs) are utilized to connect the prosumers' EMS to the VPP through base stations [24], as shown in Fig.1. The VPP participates in the transmission-level market as a price-taker. Nash bargaining is used to reach a mutually beneficial energy-sharing plan [3], [16], [19]. At the first step of Nash bargaining, prosumers communicate with the VPP, determining the optimal energy sharing plan. It requires the VPP and the prosumers to cooperatively solve the social welfare maximization problem, which aims to generate the entire social welfare surplus as much as possible so everyone can benefit from energy sharing. The imbalance caused by power loss during sharing can induces an exogenous cost for the VPP to purchase more power from the transmission-level market. The cost can be incorporated in the ex-post settlement between the VPP and the prosumers after the sharing plan is implemented. The method to allocate the exogenous cost caused by the power loss is also a complex problem and beyond the scope of this paper.

For prosumer i, its decision variables $\boldsymbol{x}_i$ includes $\boldsymbol{x}_i = [(\boldsymbol{P}_i^{\mathrm{EX}})^\top, (\boldsymbol{P}_i^{\mathrm{SH}})^\top, (\boldsymbol{P}_i^{\mathrm{L}})^\top, (\boldsymbol{P}_i^{\mathrm{CH}})^\top, (\boldsymbol{P}_i^{\mathrm{DIS}})^\top, \boldsymbol{S}_i^\top]^\top$, respectively indicating i's exchanging power with the VPP $\boldsymbol{P}_i^{\mathrm{EX}}$, sharing power with prosumer peers $\boldsymbol{P}_i^{\mathrm{SH}}$, load power $\boldsymbol{P}_i^{\mathrm{L}}$, storage's charging power $\boldsymbol{P}_i^{\mathrm{CH}}$, storage's discharging power $\boldsymbol{P}_i^{\mathrm{DIS}}$, and storage's state of charge $\boldsymbol{S}_i$. In this paper, vectors $\boldsymbol{P}_i^{\mathrm{EX}}, \boldsymbol{P}_i^{\mathrm{SH}}, \boldsymbol{P}_i^{\mathrm{L}}, \boldsymbol{P}_i^{\mathrm{CH}}, \boldsymbol{P}_i^{\mathrm{DIS}}, \boldsymbol{S}_i$ with the subscript i are in the dimension of $T \times 1$ and are stacked by prosumer i's single-period variables. For example, $\boldsymbol{P}_i^{\mathrm{EX}} = [P_{i,1}^{\mathrm{EX}}, ..., P_{i,t}^{\mathrm{EX}}, ..., P_{i,T}^{\mathrm{EX}}]^\top$. Matrix without subscript i are in the dimension of $T \times I$ and are stacked by all the prosumers' variables. For example, $\mathbf{P}^{\mathrm{EX}} = [\boldsymbol{P}_1^{\mathrm{EX}}, ..., \boldsymbol{P}_i^{\mathrm{EX}}, ..., , \boldsymbol{P}_I^{\mathrm{EX}}]$. The complete social welfare maximization problem is shown as follows:

$$\begin{aligned} \max_{\boldsymbol{x}_i} \quad U = & \underbrace{-\lambda_t^{\mathrm{b}} \left[\sum\nolimits_i P_{i,t}^{\mathrm{EX}}\right]^+ - \lambda_t^{\mathrm{s}} \left[\sum\nolimits_i P_{i,t}^{\mathrm{EX}}\right]^-}_{=U_{\mathrm{VPP}}(\mathbf{P}^{\mathrm{EX}})} + \\ & \sum\nolimits_i \underbrace{V_i\left(\boldsymbol{P}_i^{\mathrm{L}}\right) - C_i\left(\boldsymbol{P}_i^{\mathrm{CH}}, \boldsymbol{P}_i^{\mathrm{DIS}}\right)}_{=U_i(\boldsymbol{x}_i)} \\ \text{s.t.} \quad & \boldsymbol{x}_i \in \Omega_i, \sum\nolimits_i \boldsymbol{P}_i^{\mathrm{SH}} = \mathbf{0} \end{aligned} \tag{1}$$

where $U_{\mathrm{VPP}}(\mathbf{P}^{\mathrm{EX}})$ and $U_i(\boldsymbol{x}_i)$ are the equivalent social utilities for the VPP and the prosumers; $\boldsymbol{x}_i \in \Omega_i$ stands for prosumer i's individual constraints and Ω_i is a convex set, which is regarded as i's privacy. $U_{\mathrm{VPP}}(\mathbf{P}^{\mathrm{EX}})$ and $U_i(\boldsymbol{x}_i)$ take into account the utilities from $t = 1$ to $t = T$. The concrete constraints of Ω_i are listed in the appendix. $\sum_i \boldsymbol{P}_i^{\mathrm{SH}} = \mathbf{0}$ indicates the sharing power should be balanced within the prosumers for every t.

In $U_{\mathrm{VPP}}(\mathbf{P}^{\mathrm{EX}})$, λ_t^{b} and λ_t^{s} are the purchasing price and selling price of the transmission-level market, which are different $\lambda_t^{\mathrm{b}} > \lambda_t^{\mathrm{s}}$ due to transmission service charges and tax; $\lambda_t^{\mathrm{b}}[\sum_i P_{i,t}^{\mathrm{EX}}]^+$ and $\lambda_t^{\mathrm{s}}[\sum_i P_{i,t}^{\mathrm{EX}}]^-$ are the money for purchasing/selling power in the transmission-level network. In $U_i(\boldsymbol{x}_i)$, $V_i(\boldsymbol{P}_i^{\mathrm{L}})$ is the satisfaction for prosumer to use load $\boldsymbol{P}_i^{\mathrm{L}}$, which should be concave and monotonically increasing for $P_{i,t}^{\mathrm{L}}, \forall t$; $C_i(\boldsymbol{P}_i^{\mathrm{CH}}, \boldsymbol{P}_i^{\mathrm{DIS}})$ is the storage's aging cost, which is convex and monotonically increasing for $P_{i,t}^{\mathrm{CH}}, P_{i,t}^{\mathrm{DIS}}, \forall t$. The retailing electricity payments and the sharing rewards are treated as internal exchanges among the VPP and prosumers, which do not affect overall social welfare.

Algorithm 1: Standard ADMM

Input: Initial values $\boldsymbol{\alpha}_{i(0)}, \boldsymbol{P}_{i(0)}, \boldsymbol{E}_{i(0)}$; stopping criterion $\varepsilon_1, \varepsilon_2$; $k = 1$

while $\|\Delta\boldsymbol{\alpha}_{i(k,k-1)}\| \geqslant \varepsilon_1$ ***or*** $\|\Delta\boldsymbol{x}_{i(k,k-1)}\| \geqslant \varepsilon_2$ **do**

  Step 1: The VPP solves (5) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{P}_{i(k)}$, and transmits variables $\boldsymbol{E}_{i(k+1)}$ to every prosumer i;

  Step 2: The prosumer i receive $\boldsymbol{E}_{i(k+1)}$ and solve (6) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k+1)}$;

  Step 3: The prosumer i update $\boldsymbol{\alpha}_{i(k+1)}$ as:

$$\boldsymbol{\alpha}_{i(k+1)} = \boldsymbol{\alpha}_{i(k)} + \rho\left(\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k+1)}\right) \tag{2}$$

  And send updates $\boldsymbol{P}_{i(k+1)}, \boldsymbol{\alpha}_{i(k+1)}$ to the VPP.

  Step 4: The VPP waits until it receives all the prosumers' updates and sets $k = k + 1$.

end

Result: $\boldsymbol{x}_{i(k)}, \boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k)}$

III. Partial-update ADMM Algorithm

A. Problem and Algorithm Formulation

The welfare maximization problem (1) contains massive DERs' private information. ADMM is frequently utilized to solve the problem. Auxiliary variables $\mathbf{E}^{\mathrm{EX}}, \mathbf{E}^{\mathrm{SH}}$ are introduced to simulate the negotiation process for the VPP and prosumers to reach a consensus. The optimization is reformulated as Eq.(3):

$$\begin{aligned} \max_{\boldsymbol{x}_i, \mathbf{E}^{\mathrm{EX}}, \mathbf{E}^{\mathrm{SH}}} \quad & U = U_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum\nolimits_i U_i(\boldsymbol{x}_i) \\ \text{s.t.} \quad & \boldsymbol{x}_i \in \Omega_i, \sum\nolimits_i \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0} \\ & \boldsymbol{E}_i^{\mathrm{EX}} = \boldsymbol{P}_i^{\mathrm{EX}}, \boldsymbol{E}_i^{\mathrm{SH}} = \boldsymbol{P}_i^{\mathrm{SH}} \end{aligned} \tag{3}$$

The corresponding augmented Lagrangian function is:

$$\begin{aligned} \mathcal{L} = & -U + \sum\nolimits_i \boldsymbol{w}_i^\top(\boldsymbol{E}_i^{\mathrm{EX}} - \boldsymbol{P}_i^{\mathrm{EX}}) + \boldsymbol{v}_i^\top(\boldsymbol{E}_i^{\mathrm{SH}} - \boldsymbol{P}_i^{\mathrm{SH}}) \\ & + \frac{\rho}{2}\|\boldsymbol{E}_i^{\mathrm{EX}} - \boldsymbol{P}_i^{\mathrm{EX}}\|^2 + \frac{\rho}{2}\|\boldsymbol{E}_i^{\mathrm{SH}} - \boldsymbol{P}_i^{\mathrm{SH}}\|^2 \end{aligned} \tag{4}$$

where $\boldsymbol{w}_i, \boldsymbol{v}_i$ are Lagrangian multipliers for consensus constraints; ρ is the penalty factor. For brevity, the multipliers are denoted by $\boldsymbol{\alpha}_i = [\boldsymbol{w}_i^\top, \boldsymbol{v}_i^\top]^\top$; the prosumer i's and the VPP's variables are unified by $\boldsymbol{P}_i = [(\boldsymbol{P}_i^{\mathrm{EX}})^\top, (\boldsymbol{P}_i^{\mathrm{SH}})^\top]^\top$, $\boldsymbol{E}_i = [(\boldsymbol{E}_i^{\mathrm{EX}})^\top, (\boldsymbol{E}_i^{\mathrm{SH}})^\top]^\top$. The dimension of $\boldsymbol{\alpha}_i, \boldsymbol{P}_i$, and $\boldsymbol{E}_i$ is $2T \times 1$. Through decomposition, the VPP's sub-problem is:

$$\begin{aligned} \min_{\mathbf{E}^{\mathrm{EX}}, \mathbf{E}^{\mathrm{SH}}} \quad & \sum\nolimits_t -U_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum\nolimits_i \boldsymbol{\alpha}_i^\top \boldsymbol{E}_i + \frac{\rho}{2}\|\boldsymbol{E}_i - \boldsymbol{P}_i\|^2 \\ \text{s.t.} \quad & \sum\nolimits_i \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0} \end{aligned} \tag{5}$$

The prosumer i's sub-problem is:

$$\min_{\boldsymbol{x}_i \in \Omega_i} \quad -U_i(\boldsymbol{x}_i) - \boldsymbol{\alpha}_i^\top \boldsymbol{P}_i + \frac{\rho}{2}\|\boldsymbol{P}_i - \boldsymbol{E}_i\|^2 \tag{6}$$

Algorithm 2: Partial-Update ADMM

Input: Initial values $\boldsymbol{\alpha}_{i(0)}, \boldsymbol{P}_{i(0)}, \boldsymbol{E}_{i(0)}$; stopping criterion $\varepsilon_1, \varepsilon_2, \varepsilon_3$; $k=1$; $\overline{k}_i = 1$

while $\|\Delta\boldsymbol{\alpha}_{i(\overline{k}_i+1,\overline{k}_i)}\| \geqslant \varepsilon_1$ ***or*** $\|\Delta\boldsymbol{x}_{i(\overline{k}_i+1,\overline{k}_i)}\| \geqslant \varepsilon_2$ ***or*** $\|\boldsymbol{E}_{i(k)} - \boldsymbol{P}_{i(k)}\| \geqslant \varepsilon_3$ **do**

- **Step 1-a:** The VPP solves (5) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{P}_{i(k)}$;
- **Step 1-b:** The VPP selects the prosumers $i \in \mathcal{A}_k$ to participate in updates, sets $\overline{k}_i = k$ as the latest round for $i \in \mathcal{A}_k$ to get updated, and transmits $\boldsymbol{E}_{i(k+1)}$ to $i \in \mathcal{A}_k$;
- **Step 2:** The prosumers $i \in \mathcal{A}_k$ receive $\boldsymbol{E}_{i(k+1)}$ and solve (6) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k+1)}$;
- **Step 3:** The prosumers $i \in \mathcal{A}_k$ update $\boldsymbol{\alpha}_{i(k+1)}$ as Eq.(2), and uploads $\boldsymbol{P}_{i(k+1)}, \boldsymbol{\alpha}_{i(k+1)}$ to the VPP;
- **Step 4:** The VPP waits for τ_{tol} time to receive $i \in \mathcal{A}_k$'s updates and sets $k = k+1$.

end

Result: $\boldsymbol{x}_{i(k)}, \boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k)}$

The complete standard ADMM algorithm is shown as Algorithm 1. For convenience, the changes between the variables' values at round k_1 and k_2 will be denoted by $\Delta\boldsymbol{P}_{i(k_1,k_2)}, \Delta\boldsymbol{\alpha}_{i(k_1,k_2)}, \Delta\boldsymbol{E}_{i(k_1,k_2)}$. The VPP and *each* prosumer repeatedly solve sub-problems until the variables' changes are within the convergence tolerance $\varepsilon_1, \varepsilon_2$. When massive (for example, $> 10^4$) prosumers try to communicate with the VPP via establishing connections with base stations, the communication radio access network will inevitably become congested [25]. The iterations of the distributed ADMM algorithm become much slower due to the communication network congestion. It is more reasonable for the VPP to schedule a subset of the prosumers, rather than all prosumers, to participate in updates, as shown in Fig.1. Thereby, this paper proposes the partial-update ADMM algorithm, as shown in Algorithm 2. The major advantages of the partial-update ADMM are:

- *Partial update*: The VPP selects the subset of prosumers $i \in \mathcal{A}_k \neq \varnothing, \forall k$ to participate in updates (Step 1-b). The rest prosumers $i \in \mathcal{A}_k^{\text{c}}$ stay silent. It prevents the situation where massive prosumers transmit their updates simultaneously and the communication network becomes congested.
- *Waiting Deadline*: The VPP sets a deadline to wait for prosumers. The VPP is allowed to proceed when the waiting time exceeds the threshold τ_{tol}. It prevents the situation where the VPP waits a long time for updates.

Because of the partial update mechanism, there will be asynchrony between silent prosumers and the VPP, and thus the stopping condition should be revised. In Algorithm 2, the latest update round for the prosumer i is recorded as $\overline{k}_i$. For the prosumers $i \in \mathcal{A}_{k-1}^{\text{c}}$, their variables will not be updated $\Delta\boldsymbol{P}_{i(k,k-1)} = \boldsymbol{0}$, $\Delta\boldsymbol{\alpha}_{i(k,k-1)} = \boldsymbol{0}$, so the changes are recorded as their latest changes between round $\overline{k}_i + 1$ and $\overline{k}_i$. Besides, since $\boldsymbol{E}_{i(k)}$ keeps updating for all prosumers, the history changes of $\Delta\boldsymbol{\alpha}_{(\overline{k}_i+1,\overline{k}_i)}$ can not reflect the current consensus errors in round k. The VPP should check the consensus errors $\|\boldsymbol{E}_{i(k)} - \boldsymbol{P}_{i(k)}\|$ to ensure the convergence dissensus is bounded. The partial-update ADMM algorithm equals to the standard ADMM algorithm when the update set covers all of the prosumers $|\mathcal{A}_k| = I$.

For further analysis of convergence, Algorithm 2 is converted to the sequential updates from the perspective of the VPP after it computes $\boldsymbol{E}_{i(k+1)}$ and determines $i \in \mathcal{A}_k$:

$$\boldsymbol{P}_{i(k+1)} = \begin{cases} \arg\min\ (6), & i \in \mathcal{A}_k \\ \boldsymbol{P}_{i(k)}, & i \in \mathcal{A}_k^{\text{c}} \end{cases} \tag{7}$$

$$\boldsymbol{\alpha}_{i(k+1)} = \begin{cases} (2), & i \in \mathcal{A}_k \\ \boldsymbol{\alpha}_{i(k)}, & i \in \mathcal{A}_k^{\text{c}} \end{cases} \tag{8}$$

$$\boldsymbol{E}_{i(k+2)} = \arg\min\ (5) \tag{9}$$

Then the VPP will select the scheduled prosumers $i \in \mathcal{A}_{k+1}$, and repeat (7)-(9) until the algorithm converges.

B. Convergence and Optimality

The partial-update ADMM algorithm can be treated as a special variety of random multi-block ADMM algorithms [26]. The variables of I prosumers are treated as the 1st to Ith block, and the variables of the VPP are treated as the $I+1$ block. The update of prosumers is treated as the update of different blocks of variables. It is shown that the general multi-block ADMM lacks a convergence guarantee [27], while some empirical studies indicate that the multi-block ADMM is still very effective in solving real problems [28]–[30]. Besides, the partial-update algorithm can be treated as a kind of asynchronous ADMM algorithms [31], [32]. The proof of convergence for a general asynchronous ADMM algorithm for constrained optimization problems is still an open problem. It may raise concerns that the partial-update ADMM algorithm may deviate to a different solution as the standard ADMM algorithm and the optimality of energy sharing may be deteriorated. Thereby, it is proved that, under mild assumptions, the fixed points of the partial-update ADMM algorithm is the KKT point of the energy-sharing problem, which is the same as the standard ADMM algorithm. In addition, it is proved that when the given stopping condition of the partial-update ADMM algorithm is reached, the obtained result must be close to the optimal value of the energy sharing problem.

Assumption 1. (*Slater Condition and Strong Duality*) The energy sharing optimization problem satisfies the Slater condition: it is strictly feasible. Because the problem is a convex optimization problem, it also indicates the strong duality holds for the energy sharing problem.

Assumption 2. (*Update Set*) At every round, the update set size is not empty $\mathcal{A}_k \neq \emptyset$.

Assumption 3. (*Update Chance*) Every prosumer i will get at least one update chance every $T_{\max}$ round. It means every prosumer will obtain infinite update chances.

Theorem 1. (*Optimality Equivalence* [33]) Let $f(\boldsymbol{x})$ and $h(\boldsymbol{x})$ be two convex functions from $\boldsymbol{x}$ to $\mathbb{R}$, and let $h(\boldsymbol{x})$ be differentiable. Also, let Ω be a closed convex set. Then,

$$\begin{aligned} &\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}\in\Omega}\ f(\boldsymbol{x}) + h(\boldsymbol{x}) \ \Leftrightarrow \\ &\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}\in\Omega}\ f(\boldsymbol{x}) + \nabla h(\boldsymbol{x}^\star)^\top (\boldsymbol{x} - \boldsymbol{x}^\star) \end{aligned} \tag{10}$$

Let $(\boldsymbol{E}_i^\star, \boldsymbol{x}_i^\star)$ denote any optimal solution of the energy sharing problem and $\boldsymbol{\alpha}_i^\star$ denote any dual optimal solution.

Lemma 1. Suppose the fixed point of the algorithm is $\boldsymbol{\alpha}_{i(k+1)} = \boldsymbol{\alpha}_{i(k)} = \hat{\boldsymbol{\alpha}}_i, \boldsymbol{x}_{i(k+1)} = \boldsymbol{x}_{i(k)} = \hat{\boldsymbol{x}}_i, \boldsymbol{E}_{i(k+1)} = \boldsymbol{E}_{i(k)} = \hat{\boldsymbol{E}}_i$. Under assumptions 1-3, the fixed point $(\hat{\boldsymbol{E}}_i, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{\alpha}}_i)$ is the KKT point $(\boldsymbol{E}_i^\star, \boldsymbol{x}_i^\star, \boldsymbol{\alpha}_i^\star)$ of the energy sharing problem.

Proof. Please see the appendix. □

Lemma 2. After each round of update, the gap between the objective and the optimal value can be bounded as follows:

$$\begin{aligned} &-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}_{(k+2)}^{\mathrm{EX}}) + \sum\nolimits_i -\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right) - \\ &\left(-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}\star}) + \sum\nolimits_i -\tilde{U}_i\left(\boldsymbol{x}_i^\star\right)\right) \leqslant \\ &\sum\nolimits_i \|\boldsymbol{\alpha}_{i(k+1)}\| \|\boldsymbol{P}_{i(k+1)} - \boldsymbol{E}_{i(k+2)}\| + \\ &\rho \|\boldsymbol{E}_{i(k+2)} - \boldsymbol{P}_{i(k+1)}\| \|\boldsymbol{E}_i^\star - \boldsymbol{E}_{i(k+2)}\| \end{aligned} \tag{11}$$

Proof. Please see the appendix. □

When the algorithm is terminated by satisfying the consensus requirement, $\|\boldsymbol{P}_{i(k+1)} - \boldsymbol{E}_{i(k+2)}\|$ is small while $\|\boldsymbol{\alpha}_{i(k+1)}\|$ and $\|\boldsymbol{E}_i^\star - \boldsymbol{E}_{i(k+2)}\|$ is bounded. Thereby, the obtained result when the algorithm stops must be close to the optimal value.

IV. Prosumer Update Scheduling Policy

The key step in Algorithm 2 is the update scheduling step 1-b. The VPP should determine the number of prosumers participating in updates, namely the update set size $|\mathcal{A}_k|$. Besides, it also should decide which prosumers are included in $|\mathcal{A}_k|$. Different update set sizes and scheduling policies may significantly affect the convergence rate and optimality. This section deals with the two problems above.

A. Optimal Update Set Size Determination

$|\mathcal{A}_k|$ can be time-varying for different rounds k. For simplicity, the update set size $\mathcal{A}_k$ is set as the same value $|\mathcal{A}| = |\mathcal{A}_1| = ... = |\mathcal{A}_k|$ for every round k. Under this setting, the parameter complexity of the algorithm is reduced. Because Algorithm 2 only involves part of the prosumers to participate in updates, the number of convergence iterations is expected to increase. The VPP should select $|\mathcal{A}|$ by trading off the single-round update delay (decreases with a lower $|\mathcal{A}|$) and the number of convergence iterations (decreases with a larger $|\mathcal{A}|$) to minimize the overall convergence time.

From the perspective of the VPP, the latency in each round is mainly composed of 4 parts:

- The downlink transmission delay τ_1: the delay to transmit $\boldsymbol{E}_{i(k+1)}$ to $i \in \mathcal{A}_k$. The data size is usually around 10 kbits, and the transmission rate of typical cellular-based MTC (LTE-M) is around 1 Mbits/s [24]. $\tau_1 \approx 10$ ms.
- The prosumer's computing delay τ_2: the delay for $i \in \mathcal{A}_k$ to solve sub-problems and update multipliers for 0.5-1.5 s.
- The prosumer's access delay τ_3: the delay for $i \in \mathcal{A}_k$ to access base stations before transmitting uplink data. The more prosumers $|\mathcal{A}|$ attempt to establish access, the larger τ_3 will be. τ_3 varies from 20 ms to several seconds.
- The uplink transmission delay τ_4: the delay for $i \in \mathcal{A}_k$ to upload $\boldsymbol{P}_{i(k+1)}, \boldsymbol{\alpha}_{i(k+1)}$ to the VPP. Similar to τ_1, the uplink transmission delay $\tau_4 \approx 10$ ms.

After that, the VPP will use the collected updates to update $\boldsymbol{E}_{i(k+2)}$. The computing delay τ_0 to solve the VPP's optimization problem is 0.5-1.5 s.

The access delay τ_3 is the major bottleneck in the massive MTC. It originates from the random access (RA) process in cellular communications [34]. In 3GPP standards, at a fixed interval (the time slot), MTC devices (including EMS devices) can contend for the uplink access opportunities. Each MTC device randomly selects one of the sequences (preambles) in the pool (composed of M preambles). If multiple MTC devices transmit the same preamble at the same time, then all of them fail due to preamble collision. The access request can be successfully sent only when there is no concurrent transmission of the same preamble at the same time slot. The successful requests will be granted with uplink transmission resources to upload data, while the failed MTCs will keep trying to establish access to the base station continuously. Under limited opportunities, massive concurrent access can drastically increase the collision probability and thus will increase the access latency [35].

$|\mathcal{A}|$ is selected to minimize the overall convergence time:

$$\begin{aligned} &\min\nolimits_{|\mathcal{A}|} \ \tau K \\ &\quad \text{s.t. } \tau = \tau_0 + \min\left(\tau_1 + \tau_2 + \tau_3 + \tau_4, \tau_{\mathrm{tol}}\right) \\ &\qquad \tau_3 = \mathcal{F}_{\mathrm{Access}}\left(|\mathcal{A}|\right), K = \mathcal{F}_{\mathrm{Iteration}}\left(|\mathcal{A}|\right) \end{aligned} \tag{12}$$

where τ is the summed single-round iteration time; K is the total convergence iterations. The single-round iteration time τ will not exceed $\tau_0 + \tau_{\mathrm{tol}}$ because the VPP will not wait after the time hits the waiting deadline. Neither the access delay τ_3 nor the number of convergence iterations K can be expressed analytically. The single-round access attempts are determined by a complex access contest process among prosumers. The number of convergence iterations varies with the specific prosumer update policy and detail parameters. Thereby, the optimal set size of prosumers can only be determined numerically through simulations. Through simulations, the VPP knows how the access delay $\mathcal{F}_{\mathrm{Access}}(|\mathcal{A}|)$ and the total number of iterations $\mathcal{F}_{\mathrm{Iteration}}(|\mathcal{A}|)$ vary with the update set size. The VPP can numerically solve the problem (12) by enumerating the objective value τK under different $|\mathcal{A}|$ offline. Then it can approximately compute the set size when the minimum convergence time is reached.

B. Fair and Efficient Scheduling Policy

An efficient scheduling policy for the ADMM algorithm should help it reach the saddle point of the augmented Lagrangian function (4) as soon as possible so that convergence iterations are minimized. After one round of updates via (7)-(9), the changes of the augmented Lagrangian function can be decomposed as (13):

$$\begin{aligned} &\Delta\mathcal{L} = \mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+2)}, \boldsymbol{\alpha}_{(k+1)}) - \mathcal{L}(\mathbf{x}_{(k)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)}) = \\ &\underbrace{\mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)}) - \mathcal{L}(\mathbf{x}_{(k)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)})}_{=\Delta} + \end{aligned} \tag{13}$$

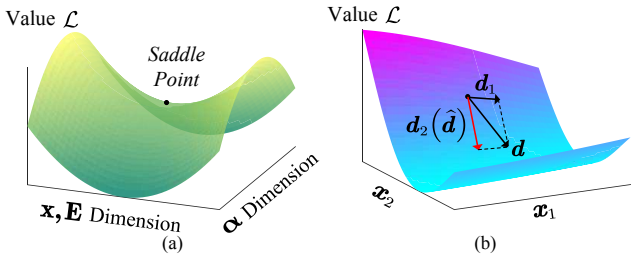


Fig. 2. (a) The ADMM is to find the saddle point of the Lagrangian function; (b) the illustration the intuition behind the efficient update policy.

$$\underbrace{\mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k+1)}) - \mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)})}_{=\Delta\mathcal{L}_{\text{MUL}}} + \underbrace{\mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+2)}, \boldsymbol{\alpha}_{(k+1)}) - \mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k+1)})}_{=\Delta\mathcal{L}_{\text{VPP}}}$$

where $\Delta\mathcal{L}_{\text{PRO}}, \Delta\mathcal{L}_{\text{MUL}}, \Delta\mathcal{L}_{\text{VPP}}$ are the changes brought by updates of (7)-(9), respectively. $\Delta\mathcal{L}_{\text{PRO}}, \Delta\mathcal{L}_{\text{VPP}}$ are brought by primal variables' changes and to find the *minimum* in $\boldsymbol{x}$'s and $\boldsymbol{E}$'s dimensions; $\Delta\mathcal{L}_{\text{MUL}}$ is brought by Lagrangian multipliers' changes and to find the *maximum* in $\boldsymbol{\alpha}$'s dimensions, shown as Fig.2(a).

In the partial-update ADMM algorithm, on $\boldsymbol{x}$'s dimensions, only a subset of $\boldsymbol{x}_{i\in\mathcal{A}_k}$ of $\boldsymbol{x}$ can be selected. A toy example of a two-dimension (two-prosumer) case is shown in Fig. 2(b) to illustrate the intuition behind the efficient policy. The descent direction is set as $\boldsymbol{d}$ when all prosumers are selected. If it is limited to selecting one dimension (prosumer) to go descent, then the dimension with more impact on the direction $\boldsymbol{d}$ should be selected as $\hat{\boldsymbol{d}}$. In that way, the descent direction is the steepest, and then it can reach the minimum as fast as the full dimension direction does. Similar results can be obtained for the multipliers $\boldsymbol{\alpha}$, whereas the direction is the ascent direction.

Directly analyzing the impacts of different prosumers on the descent value of $\Delta\mathcal{L}_{\text{PRO}}, \Delta\mathcal{L}_{\text{VPP}}$ and the ascent value $\Delta\mathcal{L}_{\text{MUL}}$ is difficult. Thereby, by using the convexity of the sub-problems, $\Delta\mathcal{L}_{\text{PRO}}, \Delta\mathcal{L}_{\text{MUL}}, \Delta\mathcal{L}_{\text{VPP}}$ are substituted by their bounds as follows:

$$\Delta\mathcal{L}_{\text{PRO}} \leqslant -\frac{\rho}{2}\sum\nolimits_{i\in\mathcal{A}_k} \|\boldsymbol{P}_{i(k+1)} - \boldsymbol{P}_{i(k)}\|^2 \tag{14}$$

$$\Delta\mathcal{L}_{\text{MUL}} = \frac{1}{\rho}\sum\nolimits_{i\in\mathcal{A}_k} \|\boldsymbol{\alpha}_{i(k+1)} - \boldsymbol{\alpha}_{i(k)}\|^2 \tag{15}$$

$$\Delta\mathcal{L}_{\text{VPP}} \leqslant -\frac{\rho}{2}\sum\nolimits_{i} \|\boldsymbol{E}_{i(k+2)} - \boldsymbol{E}_{i(k+1)}\|^2 \tag{16}$$

In (14)-(16), the effects of prosumers on $\Delta\mathcal{L}$ are separated, making it easy to find the steepest update direction. The proof for Eqs.(14)-(16) is in the appendix.

Notice that the update of $\Delta\boldsymbol{E}_{i(k+2,k+1)}$ in (16) depends on $\Delta\boldsymbol{\alpha}_{i(k+1,k)}$ and $\Delta\boldsymbol{P}_{i(k+1,k)}$ through (9). Thereby, the bound of $\Delta\mathcal{L}_{\text{VPP}}$ can be further scaled as (17):

$$\begin{aligned}\Delta\mathcal{L}_{\text{VPP}} \leqslant\ & \frac{\rho}{2I}\|\sum\nolimits_i \Delta\boldsymbol{P}_{i(k+1,k)} - \rho^{-1}\Delta\boldsymbol{\alpha}_{i(k+1,k)}\|^2 \\ & - \frac{\rho}{2}\sum\nolimits_{i\in\mathcal{A}_k} \|\Delta\boldsymbol{P}_{i(k+1,k)} - \rho^{-1}\Delta\boldsymbol{\alpha}_{i(k+1,k)}\|^2\end{aligned} \tag{17}$$

The proof for Eq.(17) is in the appendix.

To find the prosumers to compromise the steepest direction, the VPP selects the prosumers as follows:

$$\begin{aligned}\min_{a_i}\ & -\frac{\rho}{2}\sum\nolimits_i a_i\|\Delta\boldsymbol{P}_{i(k+1,k)}\|^2 - \frac{1}{\rho}\sum\nolimits_i a_i\|\Delta\boldsymbol{\alpha}_{i(k+1,k)}\|^2 \\ & - \frac{\rho}{2}\sum\nolimits_i a_i\|\Delta\boldsymbol{P}_{i(k+1,k)} - \rho^{-1}\Delta\boldsymbol{\alpha}_{i(k+1,k)}\|^2 \\ & + \frac{\rho}{2I}\|\sum\nolimits_i a_i(\Delta\boldsymbol{P}_{i(k+1,k)} - \rho^{-1}\Delta\boldsymbol{\alpha}_{i(k+1,k)})\|^2 \\ \text{s.t.}\ & \sum\nolimits_i a_i = |\mathcal{A}|\,, a_i \in \{0,1\}\end{aligned} \tag{18}$$

where a_i is the binary variable to decide whether the prosumer i should be selected at this round. (18) is a pure integer programming problem with I binary variables. It aims to find the smallest combination of $|\mathcal{A}|$ in all I prosumers, whose complexity is $O(\frac{I!}{|\mathcal{A}|!(I-|\mathcal{A}|)!})$.

Nevertheless, the pure efficient scheduling via Eq.(18) is biased. Some prosumers may hardly be given any update chance and may raise concerns about fairness. The fully fair scheduling policy is the round-robin policy, where prosumers are scheduled one by one. To improve fairness, this VPP should spare part rounds for every prosumer to participate in updates. Therefore, the round-robin and efficient policies should be switched in different rounds, compromising the whole scheduling policy. In that way, the combined scheduling policy is fair and efficient. For example, from round k_1+1 to round k_2, prosumers are scheduled by the round-robin policy to promote fairness; during round k_2+1 to round k_3, prosumers are scheduled by the efficient policy Eq.(18) to improve efficiency. The frequency of switching policies are contracted by the VPP and the prosumers. It is recommended that the round-robin policy and the efficient policy are switched every $\lceil I/|\mathcal{A}|\rceil$ rounds. Under this setting, every prosumer can get updated at least once during the $\lceil I/|\mathcal{A}|\rceil$ of round-robin policy rounds. Besides, the maximum update interval for a prosumer is $2\times\lceil I/|\mathcal{A}|\rceil$ when it does not involve in any update during the $\lceil I/|\mathcal{A}|\rceil$ of efficient policy rounds. The interval is not too long such that the fairness and the optimality of the algorithm is guaranteed.

C. The Complete Online Partial-update ADMM Algorithm with the Fair and Efficient Policy

Directly using the fair and efficient scheduling policy by solving Eq.(18) may incur several problems as follows:

- *Causality:* The problem takes the $\Delta\boldsymbol{\alpha}_{i(k+1,k)}, \Delta\boldsymbol{P}_{i(k+1,k)}$ as parameters. However, the VPP can not know $\boldsymbol{\alpha}_{i(k+1)}$ and $\boldsymbol{P}_{i(k+1)}$ before receiving updates from the prosumer i. Thereby, the VPP needs a method to estimate the values $\Delta\boldsymbol{\alpha}_{i(k+1,k)}, \Delta\boldsymbol{P}_{i(k+1,k)}$ before scheduling i.
- *Complexity:* Eq.(18) is a mixed integer problem, which is NP-hard and time-consuming. If the scheduling policy adds too much additional time in each round, then the benefit brought by efficient scheduling is overridden and the online performance of the whole algorithm will degrade.

The problems above are tackled one by one as follows:

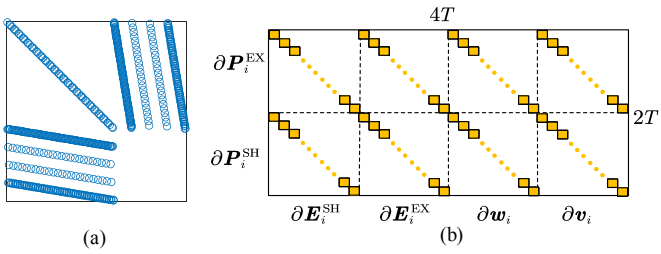


Fig. 3. (a) An example of the sparse structure of $\mathbf{M}_i$; (b) an illustration of the sparsification process of $\mathbf{G}_i$. (Non-zero elements are marked in colors.)

1) Gradient-based Estimation Method: The gradient-based method is used to estimate $\Delta\boldsymbol{\alpha}_{i(k+1,k)}, \Delta\boldsymbol{P}_{i(k+1,k)}$. The latest (round $\bar{k}_i$) problem solved by the prosumer i is:

$$\min_{\boldsymbol{x}_i \in \Omega_i} \ -U_i\left(\boldsymbol{x}_i\right) - \boldsymbol{\alpha}_{i\left(\bar{k}_i\right)}^{\top} \boldsymbol{P}_i + \frac{\rho}{2} \left\| \boldsymbol{P}_i - \boldsymbol{E}_{i\left(\bar{k}_i+1\right)} \right\|^2 \tag{19}$$

If the prosumer i participates in the update of round k, the optimization problem to be solved by i will be changed. In the objective function, $\boldsymbol{\alpha}_{i(\bar{k}_i)}$ is changed to $\boldsymbol{\alpha}_{i(k)}$; $\boldsymbol{E}_{i(\bar{k}_i+1)}$ is changed to $\boldsymbol{E}_{i(k+1)}$. $\Delta\boldsymbol{P}_{i(k+1,k)}$ is brought by these two changes. If the VPP can estimate the sensitivity of $\boldsymbol{P}_{i(k+1)}$ with respect to the changes of parameters, namely the gradient $\mathbf{G}_i$, then it can estimate $\Delta\boldsymbol{P}_{i(k+1,k)}$ as follows:

$$\begin{aligned} \Delta\boldsymbol{P}_{i(k+1,k)} &\approx \begin{bmatrix} \nabla_{\boldsymbol{E}_i} \boldsymbol{P}_i \\ \nabla_{\boldsymbol{\alpha}_i} \boldsymbol{P}_i \end{bmatrix}^{\top} \begin{bmatrix} \boldsymbol{E}_{i(k+1)} - \boldsymbol{E}_{i(\bar{k}_i+1)} \\ \boldsymbol{\alpha}_{i(k)} - \boldsymbol{\alpha}_{i\left(\bar{k}_i\right)} \end{bmatrix} \\ &= \mathbf{G}_i^{\top} \Delta\boldsymbol{\theta}_i \end{aligned} \tag{20}$$

where $\Delta\boldsymbol{\theta}_i = [\Delta\boldsymbol{E}_{i(k+1,\bar{k}_i+1)}, \Delta\boldsymbol{\alpha}_{i(k,\bar{k}_i)}]$ represents parameter changes. $\mathbf{G}_i$ can be obtained through the general sensitivity analysis method [36] for the convex optimization problem (19). Suppose that the equality constraints and the active inequality constraints are $\boldsymbol{h}(\boldsymbol{x}_i)$ in Ω_i, and the corresponding dual multipliers are $\boldsymbol{y}_i$. The Lagrangian function of Eq.(19) is

$$L_i = -U_i(\boldsymbol{x}_i) - \boldsymbol{\alpha}_{i(\bar{k}_i)}^{\top} \boldsymbol{P}_i + \frac{\rho}{2} \|\boldsymbol{P}_i - \boldsymbol{E}_{i(\bar{k}_i+1)}\|^2 + \boldsymbol{y}_i^{\top} \boldsymbol{h}\left(\boldsymbol{x}_i\right) \tag{21}$$

which is named as the individual Lagrangian function to distinguish (21) from (4). The optimality condition of the prosumer i is $\nabla_{\boldsymbol{x}_i} L_i = \mathbf{0}$. When $\boldsymbol{E}$ changes, $\nabla_{\boldsymbol{x}_i} L_i = \mathbf{0}$ still approximately holds. Thereby, the sensitivity can be obtained by computing the derivative of $\nabla_{\boldsymbol{x}_i} L_i = \mathbf{0}$ as:

$$\underbrace{\begin{bmatrix} \nabla_{\boldsymbol{x}_i \boldsymbol{x}_i} L_i & \nabla_{\boldsymbol{x}_i} \boldsymbol{h}(\boldsymbol{x}_i) \\ (\nabla_{\boldsymbol{x}_i} \boldsymbol{h}(\boldsymbol{x}_i))^{\top} & 0 \end{bmatrix}}_{=\mathbf{M}_i} \begin{bmatrix} \nabla_{\boldsymbol{\theta}_i} \boldsymbol{x}_i \\ \nabla_{\boldsymbol{\theta}_i} \boldsymbol{y}_i \end{bmatrix} = \underbrace{\begin{bmatrix} -\nabla_{\boldsymbol{\theta}_i \boldsymbol{x}_i} L_i \\ 0 \end{bmatrix}}_{=\mathbf{N}_i} \tag{22}$$

$\mathbf{G}_i$ is extracted from the relevant elements of $\nabla_{\boldsymbol{\theta}_i} \boldsymbol{x}_i$.

After $\Delta\boldsymbol{P}_{i(k+1,k)}$ is estimated, $\Delta\boldsymbol{\alpha}_{i(k+1,k)}$ can be obtained from (2). To use the estimation method, the prosumers need to transmit $\mathbf{G}_i$ to the VPP to inform the VPP about their willingness to adjust their sharing plans. Transmitting gradients is common in distributed algorithms (such as federated learning [37]) and will not violate prosumers' privacy.

2) Complexity Reduction: First, prosumer i is required to compute its gradient matrix $\mathbf{G}_i$ through sensitivity analysis in addition to updating its variables. The extra computation in this step is to solve the set of linear equations $\mathbf{M}_i \left[(\nabla_{\boldsymbol{\theta}_i} \boldsymbol{x}_i)^{\top}, (\nabla_{\boldsymbol{\theta}_i} \boldsymbol{y}_i)^{\top}\right]^{\top} = \mathbf{N}_i$. The dimension of the prosumer i's primal variable $\boldsymbol{x}_i$ is $6T \times 1$, and thus the dimension of the matrix $\mathbf{M}_i$ is above 144 when $T = 24$ considering there must be some active constraints. Generally, the complexity to solve this linear equation is $O(\frac{2}{3}D^3)$ where D is the dimension of the square matrix $\mathbf{M}_i$. However, the matrix $\mathbf{M}_i$ is highly sparse, as shown in Fig.3(a). Most of the time, there are only 1-5 non-zero elements in one row of $\mathbf{M}_i$. Thereby, the inverse of the matrix $\mathbf{M}_i$ can be efficiently computed through the sparse lower–upper (LU) decomposition, whose theoretic computation cost equal to the same floating-point operations with the number of non-zero elements (around $O(D)$) and far less than $O(\frac{2}{3}D^3)$. In case studies, we will show the time to compute $\mathbf{M}_i^{-1}$ is extremely short compared with the solution time of the optimization problem. Thereby, the additional time to compute matrix $\mathbf{G}_i$ can be ignored.

Next, the prosumer i will transmit the gradient matrix $\mathbf{G}_i$ with the update variables $\boldsymbol{P}_i, \boldsymbol{\alpha}_i$ to the VPP. The dimensions of $\boldsymbol{P}_i$, $\boldsymbol{\alpha}_i$, and $\mathbf{G}_i$ are $2T \times 1$, $2T \times 1$, $2T \times 4T$, respectively. Thereby, transmitting the full matrix $\mathbf{G}_i$ greatly increases the communication payloads. To reduce the communication complexity, the elements across different time slots are dropped: $\frac{\partial P_{i,t_1}}{\partial E_{i,t_2}} = 0$, $\frac{\partial P_{i,t_1}}{\partial \alpha_{i,t_2}} = 0$, $\forall t_1 \neq t_2$. The element dropout process is also illustrated in Fig. 3(b), where only the 'diagonal' elements of $\mathbf{G}_i$ are transmitted. The matrix after the dropout is the sparse gradient matrix $\tilde{\mathbf{G}}_i$, where the coupling between different time slots is ignored. The number of non-zero elements $\tilde{\mathbf{G}}_i$ is $2T \times 8$, which is far less than $\mathbf{G}_i$'s dimension $2T \times 4T$. Considering $\boldsymbol{P}_i$, $\boldsymbol{\alpha}_i$, $\tilde{\mathbf{G}}_i$ altogether, the total communication payloads are now $2T \times 10$ floating-point numbers ($2T \times 10 \times 32$ bits). If $T = 24$, the total data to be updated is 15360 bits or 15 kbits. Thereby, the size of data to be transmitted is still around 10-kbit level.

With the data transmitted by the prosumers, the VPP can estimate $\Delta\boldsymbol{P}_{i(k+1,k)}, \Delta\boldsymbol{\alpha}_{i(k+1,k)}$ and treat them as parameters to solve the problem Eq.(18). The problem Eq.(18) is simplified by ignoring the coupled term $\frac{\rho}{2I} \| \sum_i a_i (\Delta\boldsymbol{P}_{i(k+1,k)} - \rho^{-1} \Delta\boldsymbol{\alpha}_{i(k+1,k)}) \|^2$ among different prosumers. In that way, the scheduling optimization problem is simplified, Eq.(18) is reformulated as the following problem:

$$\begin{aligned} \min_{a_i} \sum\nolimits_i a_i \Big(& \underbrace{-\frac{\rho}{2} \|\Delta\boldsymbol{P}_{i(k+1,k)}\|^2 - \frac{\rho}{2} \|\Delta\boldsymbol{P}_{i(k+1,k)}}_{\cdots} \\ & \underbrace{- \frac{\Delta\boldsymbol{\alpha}_{i(k+1,k)}}{\rho} \|^2 - \frac{1}{\rho} \left\| \Delta\boldsymbol{\alpha}_{i(k+1,k)} \right\|^2 \Big)}_{=\Lambda_i} \\ \text{s.t.} \ & \sum\nolimits_i a_i = |\mathcal{A}|, a_i \in \{0,1\} \end{aligned} \tag{23}$$

The simplified problem (23) is a top-$|\mathcal{A}|$ problem, which equivalently finds the $|\mathcal{A}|$th maximum of the value $-\Lambda_i$ among prosumers. The complexity of this problem is $O(I \times |\mathcal{A}|)$ by using the classical bubble sorting method. The complexity satisfies $O(I \times |\mathcal{A}|) \ll O(\frac{I!}{|\mathcal{A}|!(I-|\mathcal{A}|)!})$ when $1 < |\mathcal{A}| \ll I$.

Algorithm 3: Online Partial Update ADMM with the Fair and Efficient Scheduling Policy

Input: Initial values, stopping criterion, $k = 1$; $\overline{k}_i = 0$

Select $|\mathcal{A}|$ via solving the problem Eq.(12);
while *stopping condition is not met* **do**
  Step1-a: The VPP solves (5) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{P}_{i(k)}$;
  Step1-b:
  if k *is the fair round* **then**
    Use the round-robin policy;
  else if k *is the efficient round* **then**
    Use the efficient policy;
  Step2: The prosumers $i \in \mathcal{A}_k$ receive $\boldsymbol{E}_{i(k+1)}$ and solve (19) with $\boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k+1)}$;
  Step3: The prosumers $i \in \mathcal{A}_k$ update multipliers as Eq.(2);
  Step4: The prosumers $i \in \mathcal{A}_k$ conducts sensitivity analysis of the problem (19) to obtain the gradient matrix $\mathbf{G}_i$, and send updates $\boldsymbol{P}_{i(k)}$, $\boldsymbol{\alpha}_{i(k+1)}$, and the sparse gradient matrix $\tilde{\mathbf{G}}_i$ to the VPP;
  Step5: The VPP waits for τ_{tol} time to receive $i \in \mathcal{A}_k$'s updates and sets $k = k + 1$.
end
Function `Round-robin Policy` **:**
  $j = \max(\mathcal{A}_{k-1})$;
  return $\mathcal{A}_k = (j + 1 : j + |\mathcal{A}|) \mod I$;
End;
Function `Effcient Policy` **:**
  Compute $\Delta \boldsymbol{P}_{i(k+1,k)}, \Delta \boldsymbol{\alpha}_{i(k+1,k)}$ with $\tilde{\mathbf{G}}_i$ for all i through Eq.(20) and Eq.(2), and then compute Λ_i for all i shown in Eq.(23);
  Find the top-$|\mathcal{A}|$ elements of $-\Lambda_i$ as $\mathcal{A}_k$;
  return $\mathcal{A}_k$;
End;
Result: $\boldsymbol{x}_{i(k)}, \boldsymbol{\alpha}_{i(k)}, \boldsymbol{E}_{i(k)}$

In summary, the complete online partial-update ADMM with the fair and efficient update scheduling policy is shown as Algorithm 3. The details of the round-robin scheduling policy and the efficient policy are shown in the function form at the bottom. Notice that before the VPP uses the efficient policy, it should receive $\tilde{\mathbf{G}}_i$ from each prosumer i at least once. Thereby, the algorithm should start with the round-robin policy to collect enough information about $\tilde{\mathbf{G}}_i$.

V. Case Studies

This section implements numerical experiments to verify the effectiveness of the proposed method. All the experiments are implemented using Matlab software's parallel workers on the servers with an AMD EPYC 7H12 @ 2.60GHz CPU and 384.0 GB of RAM at the Beijing Super Cloud Computing Center. The optimization solver is the Gurobi software.

The energy sharing is among the VPP and $I = 10^4$ prosumers for the next $T = 24$ hours. The prosumers are evenly distributed under 10 base stations (10^3 prosumers for each base station). The prosumers' utility function is set to be a quadratic function $V_i(\boldsymbol{P}_i^{\text{L}}) = \sum_t \xi_{i,t}(P_{i,t}^{\text{L}})^2 + \varrho_{i,t} P_{i,t}^{\text{L}}$. $I = 10^4$

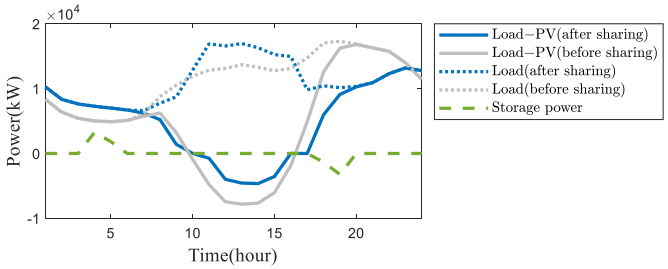


Fig. 4. Loads, loads-PV generations before and after energy sharing, the storage power after sharing.

randomly selected load profiles in the Ireland CER project [38] and the London LCL project [39] are utilized to systematically generate prosumers' utility functions. The maximum load $\overline{P_{i,t}^{\text{L}}}$ is set to be 3 times of the recorded load profile; $\underline{P_{i,t}^{\text{L}}}$ is set to be half of the recorded load profile. $\varrho_{i,t}$ is randomly selected between $[10, 20]$ ¢/kW. To ensure the utility function is an increasing function, the quadratic coefficient $\xi_{i,t}$ is set as $-\varrho_{i,t}/2\overline{P_{i,t}^{\text{L}}}$ ¢/(kW)^2. The degrading function of energy storage is set as $C_i(\boldsymbol{P}_i^{\text{CH}} + \boldsymbol{P}_i^{\text{DIS}}) = \sum_t c_i \times (P_{i,t}^{\text{CH}} + P_{i,t}^{\text{DIS}})$ where $c_i \in [2, 4]$ ¢/kW. The maximum capacity of energy storage is set as 4 times of the average daily load in recorded profiles. The SOC is limited between $[0.1, 1]$ of its capacity. $S_{i,0} = S_{i,T}$ is set as 0.55 times of the storage capacity. The charging and discharging efficiencies are all set as 95%. The price data are the daily average nodal price in the PJM market from July, 2021 to July, 2022. λ_t^{b} is set as twice of the nodal price and λ_t^{s} is 1.5 times. For the ADMM algorithm, the penalty factor is set as $\rho = 2$. All the stopping condition is set as $\epsilon = 0.1$. All the initial values of variables are set as 0.

The welfare maximization problem is first directly solved, and the solution is set as the optimal point $\boldsymbol{x}^\star$. The effect of energy sharing is presented in Fig.4. Under the setting of this case, energy sharing can make full use of the PV generations, relieve the negative effect of the duck curve, and encourage prosumers to utilize their storage to arbitrage properly.

For comparisons, the case settings are as follows:

- *Round-robin case*: In Algorithm 3, the VPP only uses the round-robin policy to schedule prosumers for updates.
- *Scheduling case*: In Algorithm 3, the round-robin policy and the efficient policy are switched every $\lceil I/|\mathcal{A}| \rceil$ rounds.

The update set size $|\mathcal{A}|$ varies in $\{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 10\} \times 10^3$. Through computing, some statistics are recorded as follows: the average sparsity of matrix $\mathbf{M}_i$ is 1.05%; the time to solve the prosumers' sub-problem is between $[0.17, 0.48]$ seconds; the time to compute $\mathbf{M}_i$'s inverse is between $[0.0073, 0.0078]$ seconds. The additional computation time is ignorable compared with the sub-problem solution problem.

A. Convergence Iteration Numbers

The convergence iterations with different $|\mathcal{A}|$ are shown in Fig.5. Compared with the pure round-robin case, the scheduling case takes fewer iterations to reach the convergence. Beyond that, compared with the scheduling with full

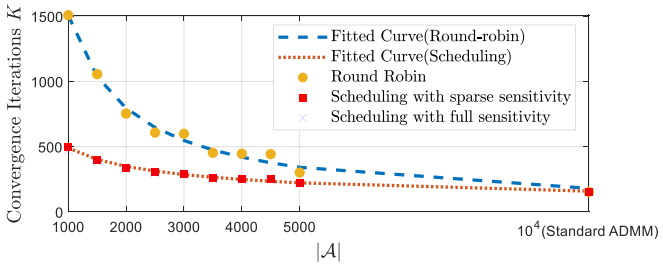


Fig. 5. The convergence iterations under different $|\mathcal{A}|$ under three cases and the fitting results of the convergence iterations K against different $|\mathcal{A}|$.

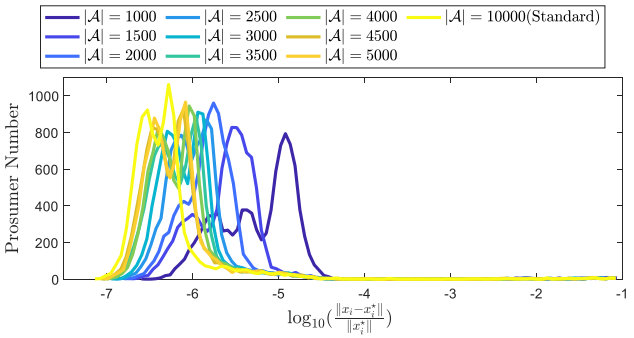


Fig. 6. The distribution for the relative optimality gap for prosumers $\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_i^\star\|}{\|\boldsymbol{x}_i^\star\|}$ under the scheduling case.

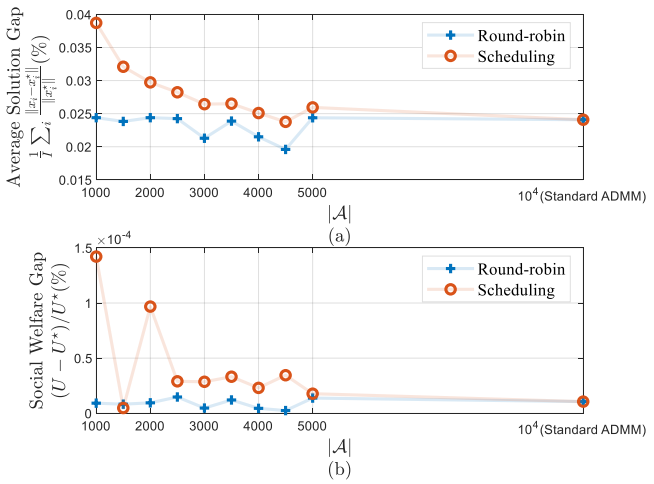


Fig. 7. (a) The average solution gap $\frac{1}{I}\sum_i \frac{\|\boldsymbol{x}_i - \boldsymbol{x}_i^\star\|}{\|\boldsymbol{x}_i^\star\|}$ for different update set sizes; (b) The social welfare gap $\frac{U - U^\star}{U^\star}$ for different update set sizes.

sensitivity case, scheduling with sparse sensitivity achieves a similar convergence rate. It shows that the sparse gradient matrix $\tilde{\mathbf{G}}_i$ is a close estimate to the full gradient matrix $\mathbf{G}_i$. We will mainly focus on the case when scheduling with sparse sensitivity matrix below.

The convergence iterations against different $|\mathcal{A}|$ are fitted by the power function. It shows that $K = \mathcal{F}_{\text{Iteration}}(|\mathcal{A}|) = 8.77 \times 10^5 \times |\mathcal{A}|^{-0.97}$ for the round-robin case and $K = \mathcal{F}_{\text{Iteration}}(|\mathcal{A}|) = 1.44 \times 10^4 \times |\mathcal{A}|^{-0.49}$ for the scheduling case. Thus, the convergence iterations are approximately inverse to the update set size $|\mathcal{A}|$. The result indicates that enlarging the update set size to the standard ADMM ($|\mathcal{A}| = 10^4$) may not be beneficial as expected. When $|\mathcal{A}|$ increases from 5×10^3 to 10^4, the total convergence iterations decrease from 215 to 153. The VPP needs to wait $2\times$ many as prosumers to upload their updates but only saves 25% of the convergence iterations, which ends in increasing the total convergence time.

B. Solution Optimality and Accuracy

The relative gap between the convergence result $\boldsymbol{x}_i$ and the optimal point $\boldsymbol{x}_i^\star$ is computed to check the optimality. The distribution of relative optimality gaps $\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_i^\star\|}{\|\boldsymbol{x}_i^\star\|}$ for prosumers under the scheduling case are exhibited in Fig.6. From $|\mathcal{A}| = 10^3$ to $|\mathcal{A}| = 10^4$, the optimality gaps decrease, showing that a larger update size can bring down the overall gap between the obtained solution and the optimal point. The average solution gap is further illustrated in Fig.7(a), where the gaps between the round-robin case and the optimal point are all exhibited. It shows that the gaps are all in the 10^{-4} level. The partial-update ADMM algorithm when $|\mathcal{A}| \in \{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\} \times 10^3$ reaches a similar average average gap with the standard ADMM algorithm. The social welfare utility gaps $\frac{U - U^\star}{U^\star}$ for different update set sizes are shown in Fig.7(b). the social welfare utility gaps $\frac{U - U^\star}{U^\star}$ all in the 10^{-6} level, showing that the result provides the social optimal solution within the whole VPP. In summary, we can conclude the obtained results are close enough to the optimal solution.

C. Convergence Trend

The variables' residues and the consensus errors during iterations are defined as follows:

$$r_{i(\text{p})} = \|\Delta \boldsymbol{x}_{i(k,k-1)}\|, r_{i(\text{d})} = \|\Delta \boldsymbol{\alpha}_{i(k,k-1)}\|,$$
$$r_{i(\text{s})} = \|\boldsymbol{E}_{i(k)} - \boldsymbol{P}_{i(k)}\|$$

$r_{i(\text{p})}$, $r_{i(\text{d})}$, and $r_{i(\text{s})}$ correspond to the three quantities in the convergence stopping criterion in the partial-update ADMM algorithm. To ensure convergence, it is required that the individual $r_{i(\text{p})}$, $r_{i(\text{d})}$, and $r_{i(\text{s})}$ are less than ε_1, ε_2, and ε_3, respectively. In other words, when the maximum residue $\max_i r_{i(\text{p})} \leq \varepsilon_1$, $\max_i r_{i(\text{d})} \leq \varepsilon_2$, and $\max_i r_{i(\text{s})} \leq \varepsilon_3$, the algorithm terminates. The average residues $\frac{1}{I}\sum_i r_{i(\text{p})}$, $\frac{1}{I}\sum_i r_{i(\text{d})}$ and $\frac{1}{I}\sum_i r_{i(\text{s})}$ for the two cases are exhibited in Fig.8 and Fig.9, respectively.

In Fig.8 with Fig.9, the standard ADMM achieves the fastest convergence rate. Partial-update ADMM costs more iterations to reach convergence since it can only involve a part of the prosumers into the update. Because the stopping condition is the same as $\max_i r_{i(\text{d})} \leqslant \varepsilon_2$, $\max_i r_{i(\text{d})}$ for different cases are nearly the same. However, $\frac{1}{I}\sum_i r_{i(\text{d})}$ differ, and the standard ADMM achieves the largest $\frac{1}{I}\sum_i r_{i(\text{d})}$ among all the cases. A smaller difference between the maximum residue $\max_i r_{i(\text{d})}$ and the average residue $\frac{1}{I}\sum_i r_{i(\text{d})}$ indicates that the residue values are more even among prosumers. This is also true when comparing the scheduling case with the round-robin case. The pure round-robin policy is fair but also wastes its time decreasing the residues that are already small enough,

which will not contribute much to the convergence. On the contrary, the scheduling case identifies those convergence-critical prosumers and makes correct efforts to bring them down.

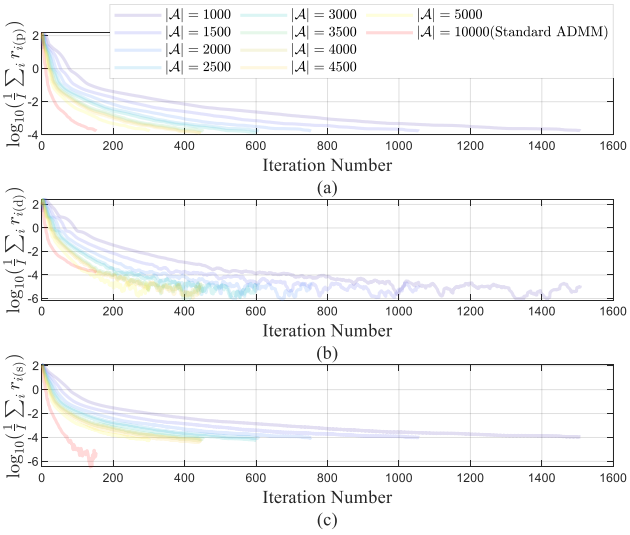


Fig. 8. Evolution of $\frac{1}{I}\sum_i r_{i(\mathrm{p})}$, $\frac{1}{I}\sum_i r_{i(\mathrm{d})}$ and $\frac{1}{I}\sum_i r_{i(\mathrm{s})}$ (Round-robin case).

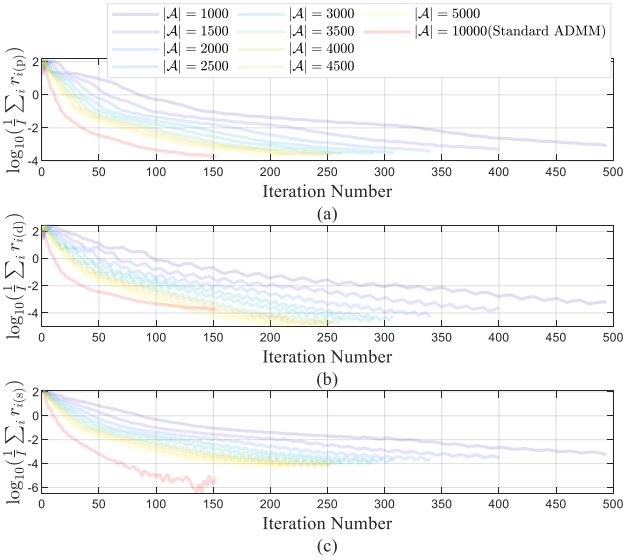


Fig. 9. Evolution of $\frac{1}{I}\sum_i r_{i(\mathrm{p})}$, $\frac{1}{I}\sum_i r_{i(\mathrm{d})}$ and $\frac{1}{I}\sum_i r_{i(\mathrm{s})}$ (Scheduling case).

To see how the scheduling case converges faster, comparisons among the evolution of the optimality gaps during iterations under different $|\mathcal{A}|$ are shown in Fig.10. It can be seen that the scheduling case has an 'acceleration effect': it can reach closer to the optimal point faster. The less the update set size $|\mathcal{A}|$ is, the more obvious the 'acceleration effect' is. Fig.10 proves that the efficient policy helps the partial update ADMM to update with the steepest direction as expected.

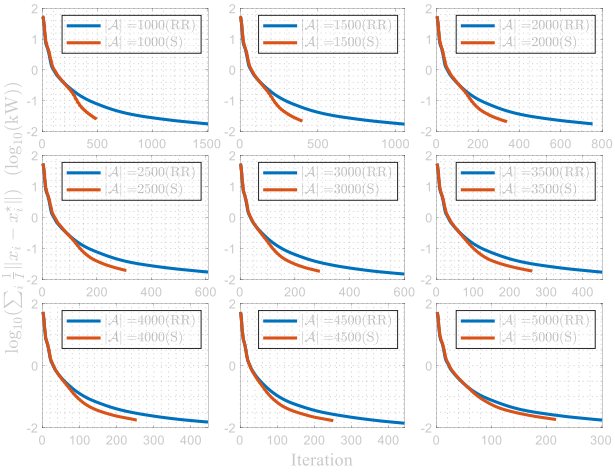


Fig. 10. The evolution of optimality gaps under various $|\mathcal{A}_k|$ (RR stands for the round-robin case and S stands for the scheduling case).

D. Optimal Update Set Size and Convergence Time

Having proved that the scheduling policy in Algorithm 3 is effective, the effect of the communication congestion is further elaborated. The additional parameters are as follows: $\tau_1 = 10$ ms. $\tau_2 = 1.48$ s. $\tau_4 = 10$ ms $\tau_0 = 1.5$ s. The number of preambles delicately reserved for the VPP is $M = 120$. The length of a time slot is $\tau_{i,\mathrm{RA}} = 20$ ms. The waiting deadline τ_{tol} is set as 11.5 s such that the prosumer are allowed to repeat its access establishment request with the base station at most 500 times (τ_3 is less than 10 s).

No Waiting Deadline: First, the single-round waiting deadline is removed. The VPP keeps waiting until it receives all the updates that it wants to collect in each round. The access delay τ_3 under different $|\mathcal{A}|$ is simulated and the simulation result is shown in Fig.11(a). As the update set size grows, there is a great chance that the communication access network gets congested. Prosumers contest access opportunities with each other and have to retry establishing connections with the base stations for more times. Thereby, the access delay increases.

In Fig.11(a), the access delay increases exponentially. The simulation results are fitted against the exponential function and the fitted result is as follows:

$$\mathcal{F}_{\mathrm{Access}}\left(|\mathcal{A}|\right) = 1.412 \times 10^{-2} \exp\left(6.74 \times 10^{-4}\,|\mathcal{A}|\right) \quad \mathrm{s}$$

Combined the fitting result of the convergence iterations $\mathcal{F}_{\mathrm{Iteration}}\left(|\mathcal{A}|\right)$ and the access delay $\mathcal{F}_{\mathrm{Access}}\left(|\mathcal{A}|\right)$, the total iteration time can be approximately expressed as:

$$\tau K = \left(3.0 + \mathcal{F}_{\mathrm{Access}}\left(|\mathcal{A}|\right)\right) \mathcal{F}_{\mathrm{Iteration}}\left(|\mathcal{A}|\right) \quad \mathrm{s} \qquad (24)$$

Fig.11(b) exhibits the total iteration time obtained from the fitting result obtained by Eq.(24) and the simulation result. For the full-update ADMM algorithm ($|\mathcal{A}| = 10^4$), the VPP involves all the prosumers to update their variables at the cost of an exponentially longer access delay. Thereby, the total convergence time of the full-update ADMM is not the smallest. The minimum convergence time can be obtained by listing the value of Eq.(24) against different $|\mathcal{A}|$. It proves that the minimum convergence time is achieved when only the update

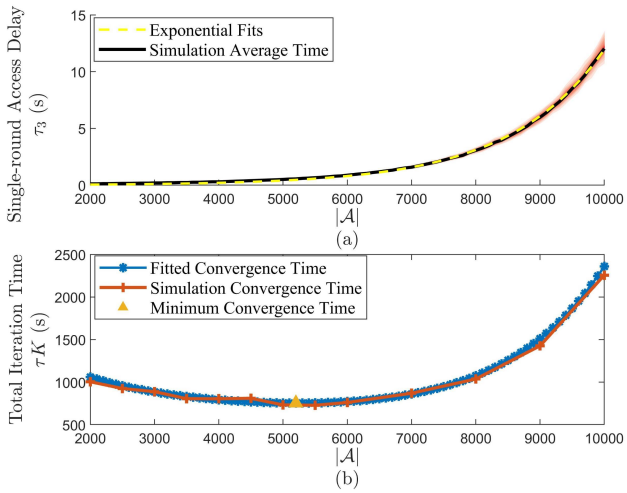


Fig. 11. (a) Single-round access delay τ_3 under different $|\mathcal{A}|$ (no waiting deadline); (b) Total convergence time τK under different $|\mathcal{A}|$ (no waiting deadline).

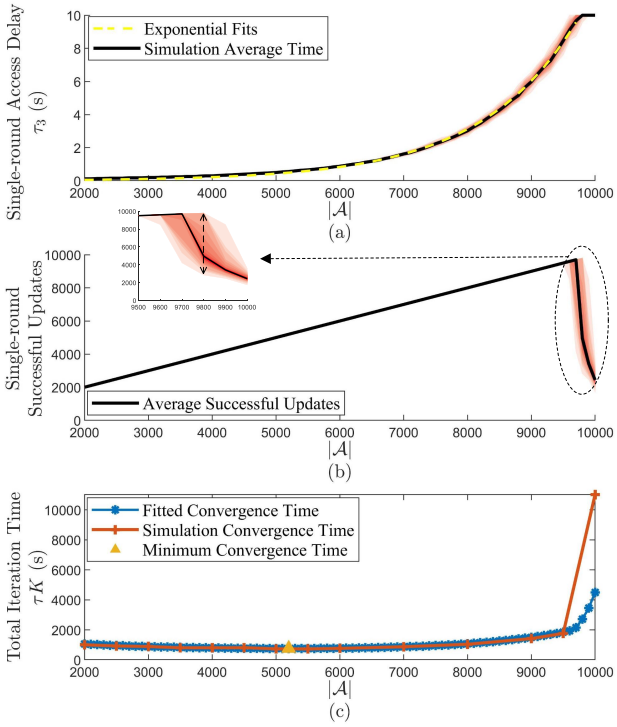


Fig. 12. (a) Single-round access delay τ_3 under different $|\mathcal{A}|$ (with waiting deadline); (b) Single-round successful updates before the deadline under different $|\mathcal{A}|$ (with waiting deadline);(c) Total convergence time τK under different $|\mathcal{A}|$ (with waiting deadline).

of partial prosumers is involved (when $|\mathcal{A}| = 5100$). With the update size $|\mathcal{A}| = 5100$ selected, the trade-off between the waiting time and the number of convergence iterations is balanced. It avoids a high iteration number when the partial-update set is too small and a long acces delay when the full-update ADMM is adopted.

With Waiting Deadline: When there is a single-round waiting deadline $\tau_{\text{tol}} = 11.5$ s, the maximum allowed access delay is 10 s. The VPP will directly move to the next iteration after the deadline is reached. The access delay τ_3 under different $|\mathcal{A}|$ with the waiting deadline is simulated and the simulation result is shown in Fig.12(a). The access delay τ_3 is always less than 10 s.

The number of single-round successful updates is shown in Fig.12(b). The simulation result shows that when $|\mathcal{A}|$ is larger than 9700, there is a great chance that the prosumers that participate in updates fail to upload their data successfully before the deadline. The actual successful updates deviate from what the VPP expect and are less than $|\mathcal{A}|$. When $|\mathcal{A}|$ is less than 9700, the invited prosumers all succeed in uploading their updates before the deadline. Fig.12(b) also shows that when $|\mathcal{A}|$ is larger than 9700, the actual successful updates fluctuate around the average value (exhibited as the red area), which may make the performance of the ADMM algorithm uncertain.

Fig.12(c) exhibits the total convergence time obtained from the fitting result and the simulation results. The fitting iteration number when $|\mathcal{A}|$ is larger than 9700 is approximated by setting $|\mathcal{A}|$ to the average update set size. For example, when $|\mathcal{A}| = 10^4$, the average successful update is 2130, so its iteration number is approximately obtained by $\mathcal{F}_{\text{Iteration}}(|2130|)$. The result shows that the waiting deadline makes the convergence time become longer when $|\mathcal{A}|$ is larger than 9700. Especially when the full-update ADMM is adopted ($|\mathcal{A}| = 10^4$), the convergence time almost doubles due to the failed updates before the deadline.

In addition, it is noticed that the random time-varying successful updates are harmful. The average successful updates when $|\mathcal{A}| = 10^4$ is similar to $|\mathcal{A}| = 2000$. However, the simulation convergence time is much larger than the fitting convergence time. This is because random successful updates can not ensure convergence-critical prosumers to get updated first like that when $|\mathcal{A}| = 2000$. Both the waiting time and the iteration number rise, together contributing to an extremely long iteration time. It highlights that waiting deadline should be carefully designated to guarantee that most of the updates can successfully arrive before the deadline as the VPP wants.

VI. Conclusions and Future Works

The paper proposes to use the online partial-update ADMM algorithm to perform energy-sharing negotiations. The update set size is selected to minimize the overall convergence time. Prosumer scheduling strategy is investigated carefully, including the efficient robin policy and the round-robin policy, to improve the efficiency and fairness of the algorithm. The whole algorithm is changed to an online version via approximation and sparsification. Numerical studies highlight that the proposed scheduling policy can reduce $30\% - 50\%$ iterations while achieving a small optimality gap. It also exhibits that careful selection of the update set size can drastically reduce the overall convergence time.

Despite these results, questions remain. Some other parameters, like the switching round number, also affect the convergence of the whole algorithm. It deserves a further investigation into how to systematically determine these parameters. The detailed effects of the distribution network

(including the power loss and the nodal voltage) are not considered in this paper. Incorporating convexified distribution power flow equations into constraints and loss related terms into the objective function may help, which deserves future investigations. A general convergence proof of the asynchronous ADMM algorithms for the constrained optimization problems is still needed. Further studies, which investigate the convergence condition of the partial-update ADMM algorithm or a more reliable partial-update ADMM variant with theoretical convergence guarantees, will need to be undertaken. Besides, this paper only discusses the communication-related issues in pool-based energy sharing. Fully decentralized energy sharing is also important, and optimizing its communication topology is a promising research direction.

APPENDIX

A. The Expressions for Ω_i

1. Power balance equation for the prosumer i ($Q_{i,t}^{\mathrm{PV}}$ is the generation power of PV panels):

$$P_{i,t}^{\mathrm{EX}} = P_{i,t}^{\mathrm{L}} + P_{i,t}^{\mathrm{CH}} - P_{i,t}^{\mathrm{DIS}} - P_{i,t}^{\mathrm{SH}} - Q_{i,t}^{\mathrm{PV}}$$

2. Storage's SOC changes, continuity, and limits (η_i^{CH}, η_i^{DIS} are the charging and discharging efficiency; $\underline{S}_i$,$\overline{S}_i$ are the SOC's lower and upper limits):

$$\begin{aligned} &S_{i,t} = S_{i,t-1} + \eta_i^{\mathrm{CH}} P_{i,t}^{\mathrm{CH}} - \eta_i^{\mathrm{DIS}} P_{i,t}^{\mathrm{DIS}} \\ &S_{i,0} = S_{i,T}, S_{i,t} \in \left[\underline{S}_i, \overline{S}_i\right] \end{aligned}$$

3. Load shift constraints ($\underline{P_i^{\mathrm{L}}}, \overline{P_i^{\mathrm{L}}}$ are the load power's lower and upper limits; $P_{i\Sigma}^{\mathrm{L}}$ is the minimum total daily loads):

$$P_{i,t}^{\mathrm{L}} \in [\underline{P_i^{\mathrm{L}}}, \overline{P_i^{\mathrm{L}}}], \sum\nolimits_t P_{i,t}^{\mathrm{L}} \geqslant P_{i\Sigma}^{\mathrm{L}}$$

4. Other limits ($\underline{P_{i,t}^{\mathrm{EX}}}, \overline{P_{i,t}^{\mathrm{EX}}}$ are the lower and upper limits for exchanged power; $\overline{P_i^{\mathrm{CH}}}, \overline{P_i^{\mathrm{DIS}}}$ are the upper limits for the storage's charging and discharging power):

$$P_{i,t}^{\mathrm{EX}} \in [\underline{P_{i,t}^{\mathrm{EX}}}, \overline{P_{i,t}^{\mathrm{EX}}}], P_{i,t}^{\mathrm{CH}} \in [0, \overline{P_i^{\mathrm{CH}}}], P_{i,t}^{\mathrm{DIS}} \in [0, \overline{P_i^{\mathrm{DIS}}}]$$

B. Proof for Lemma 1

The trivial steady state-point where no prosumers get updated is ruled out since the update set cannot be empty. Besides, since every prosumer has infinite chances to get updated, we can conclude:

$$\hat{\boldsymbol{\alpha}}_i = \hat{\boldsymbol{\alpha}}_i + \rho(\hat{\boldsymbol{E}}_i - \hat{\boldsymbol{P}}_i) \Longrightarrow \hat{\boldsymbol{E}}_i = \hat{\boldsymbol{P}}_i$$

which shows the fixed point satisfy the consensus constraint.

The optimality of the VPP's sub-problem indicates:

$$\begin{aligned} \hat{\boldsymbol{E}}_i &\overset{(a)}{\in} \arg\min_{\sum \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0}} \mathcal{L}(\hat{\mathbf{x}}_i, \mathbf{E}, \hat{\boldsymbol{\alpha}}_i) \\ &\overset{(b)}{\in} \arg\min_{\sum \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0}} -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum_i \left(\hat{\boldsymbol{\alpha}}_i + \rho(\hat{\boldsymbol{E}}_i - \hat{\boldsymbol{P}}_i)\right)^\top \boldsymbol{E}_i \\ &\overset{(c)}{\in} \arg\min_{\sum \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0}} -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum_i \hat{\boldsymbol{\alpha}}_i^\top \boldsymbol{E}_i \end{aligned}$$

where the reasoning is as follows: (a): the definition of the augmented Lagrangian function. (b): use Theorem 1 for the optimization problem. (c): $\hat{\boldsymbol{E}}_i = \hat{\boldsymbol{P}}_i$.

The minimum further indicates:

$$-\tilde{U}_{\mathrm{VPP}}(\hat{\mathbf{E}}^{\mathrm{EX}}) + \textstyle\sum_i \hat{\boldsymbol{\alpha}}_i^\top \hat{\boldsymbol{E}}_i \leqslant -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum_i \hat{\boldsymbol{\alpha}}_i^\top \boldsymbol{E}_i \tag{25}$$

Similarly, the optimality of the prosumer i's sub-problem indicates that:

$$\begin{aligned} \hat{\boldsymbol{x}}_i &\in \arg\min\nolimits_{\boldsymbol{x}_i \in \Omega_i} -\tilde{U}_i(\boldsymbol{x}_i) - \left(\hat{\boldsymbol{\alpha}}_i + \rho(\hat{\boldsymbol{E}}_i - \hat{\boldsymbol{P}}_i)\right)^\top \boldsymbol{P}_i \\ &\in \arg\min\nolimits_{\boldsymbol{x}_i \in \Omega_i} -\tilde{U}_i(\boldsymbol{x}_i) - \hat{\boldsymbol{\alpha}}_i^\top \boldsymbol{P}_i \end{aligned}$$

The minimum further indicates:

$$-\tilde{U}_i(\hat{\boldsymbol{x}}_i) - \hat{\boldsymbol{\alpha}}_i^\top \hat{\boldsymbol{P}}_i \leqslant -\tilde{U}_i(\boldsymbol{x}_i) - \hat{\boldsymbol{\alpha}}_i^\top \boldsymbol{P}_i, \forall i \tag{26}$$

Sum Eq.(25) and Eq.(26) for every i:

$$\begin{aligned} &-\tilde{U}_{\mathrm{VPP}}(\hat{\mathbf{E}}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\hat{\boldsymbol{x}}_i) + \hat{\boldsymbol{\alpha}}_i^\top(\hat{\boldsymbol{P}}_i - \hat{\boldsymbol{E}}_i) \leqslant \\ &-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\boldsymbol{x}_i) + \hat{\boldsymbol{\alpha}}_i^\top(\boldsymbol{P}_i - \boldsymbol{E}_i) \end{aligned} \tag{27}$$

Set the right-hand side as the optimal point $(\boldsymbol{E}_i^\star, \boldsymbol{x}_i^\star)$:

$$\begin{aligned} &-\tilde{U}_{\mathrm{VPP}}(\hat{\mathbf{E}}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\hat{\boldsymbol{x}}_i) + \hat{\boldsymbol{\alpha}}_i^\top(\hat{\boldsymbol{P}}_i - \hat{\boldsymbol{E}}_i) \leqslant \\ &-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}\star}) + \textstyle\sum_i -\tilde{U}_i(\boldsymbol{x}_i^\star) + \hat{\boldsymbol{\alpha}}_i^\top(\boldsymbol{P}_i^\star - \boldsymbol{E}_i^\star) \Rightarrow \\ &-\tilde{U}_{\mathrm{VPP}}(\hat{\mathbf{E}}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\hat{\boldsymbol{x}}_i) \leqslant \\ &-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}\star}) + \textstyle\sum_i -\tilde{U}_i(\boldsymbol{x}_i^\star) \end{aligned} \tag{28}$$

where the transformation is due to $\hat{\boldsymbol{P}}_i = \hat{\boldsymbol{E}}_i$ at the fixed point and $\boldsymbol{P}_i^\star = \boldsymbol{E}_i^\star$ at the optimal point. Thereby, the point $(\boldsymbol{E}_i^\star, \boldsymbol{x}_i^\star)$ must be primal optimal and the $\leqslant$ in Eq.(27) must be $=$. From Eq.(27), it can also be inferred that:

$$\begin{aligned} &-\tilde{U}_{\mathrm{VPP}}(\hat{\mathbf{E}}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\hat{\boldsymbol{x}}_i) \leqslant \\ &\min\nolimits_\Omega -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \textstyle\sum_i -\tilde{U}_i(\boldsymbol{x}_i) + \hat{\boldsymbol{\alpha}}_i^\top(\boldsymbol{P}_i - \boldsymbol{E}_i) \end{aligned} \tag{29}$$

where $\Omega = \{\boldsymbol{x}_i \in \Omega_i, \sum \boldsymbol{E}_i^{\mathrm{SH}} = \mathbf{0}\}$. Let the dual function be:

$$d(\boldsymbol{\alpha}) = \min_{\Omega} -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \sum\nolimits_i -\tilde{U}_i(\boldsymbol{x}_i) + \boldsymbol{\alpha}_i^\top(\boldsymbol{P}_i - \boldsymbol{E}_i)$$

Since $\boldsymbol{\alpha}_i^\star$ is the optimal solution of the dual function, thereby:

$$\begin{aligned} d(\boldsymbol{\alpha}) &\leqslant \max\nolimits_{\boldsymbol{\alpha}} d(\boldsymbol{\alpha}) = d(\boldsymbol{\alpha}^\star) \\ &= -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}\star}) + \textstyle\sum_i -\tilde{U}_i(\boldsymbol{x}_i^\star) \end{aligned} \tag{30}$$

where the last equality is due to the strong duality. If we replace $\boldsymbol{\alpha}$ by $\hat{\boldsymbol{\alpha}}_i$ in Eq.(30) and combine Eq.(29), it is proved that $d(\hat{\boldsymbol{\alpha}}_i) = d(\boldsymbol{\alpha}^\star)$. So $\hat{\boldsymbol{\alpha}}_i$ must be dual optimal. In summary, $(\hat{\boldsymbol{E}}_i, \hat{\boldsymbol{x}}_i, \hat{\boldsymbol{\alpha}}_i)$ is the KKT point.

C. Proof for Lemma 2

The optimality of $\boldsymbol{x}_{i(k+1)}$ indicates that:

$$\begin{aligned}\boldsymbol{x}_{i(k+1)} &\overset{(a)}{\in} \arg\min_{\boldsymbol{x}_i\in\Omega_i} -\tilde{U}_i\left(\boldsymbol{x}_i\right) - \\ &\quad \left(\boldsymbol{\alpha}_{i(\bar{k}_i)} + \rho(\boldsymbol{E}_{i(\bar{k}_i+1)} - \boldsymbol{P}_{i(\bar{k}_i+1)})\right)^\top \boldsymbol{P}_i, \forall i \\ &\overset{(b)}{\in} \arg\min_{\boldsymbol{x}_i\in\Omega_i} -\tilde{U}_i\left(\boldsymbol{x}_i\right) - \left(\boldsymbol{\alpha}_{i(k+1)}\right)^\top \boldsymbol{P}_i, \forall i\end{aligned}$$

where the reasoning is as follows: (a): the definition of the augmented Lagrangian function. (b): use $\bar{k}_i = k$ for $i \in \mathcal{A}_k$; use $\boldsymbol{\alpha}_{i(\bar{k}_i+1)} = \boldsymbol{\alpha}_{i(k+1)}$ for $i \notin \mathcal{A}_k$.
The minimum further indicates:

$$\begin{aligned}&-\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right) - \left(\boldsymbol{\alpha}_{i(k+1)}\right)^\top \boldsymbol{P}_{i(k+1)} \leqslant \\ &-\tilde{U}_i\left(\boldsymbol{x}_i^\star\right) - \left(\boldsymbol{\alpha}_{i(k+1)}\right)^\top \boldsymbol{P}_i^\star, \forall i\end{aligned} \tag{31}$$

Similarly, the optimality of $\boldsymbol{E}_{i(k+2)}$ is equivalent to:

$$\begin{aligned}\boldsymbol{E}_{i(k+2)} \in\ &\arg\min_{\sum \boldsymbol{E}_i^{\mathrm{SH}}=\boldsymbol{0}} -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}) + \\ &\sum\nolimits_i \left(\boldsymbol{\alpha}_{i(k+1)} + \rho\left(\boldsymbol{E}_{i(k+2)} - \boldsymbol{P}_{i(k+1)}\right)\right)^\top \boldsymbol{E}_i\end{aligned}$$

The minimum further indicates:

$$\begin{aligned}&-\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}}_{(k+2)}) + \\ &\sum\nolimits_i \left(\boldsymbol{\alpha}_{i(k+1)} + \rho(\boldsymbol{E}_{i(k+2)} - \boldsymbol{P}_{i(k+1)})\right)^\top \boldsymbol{E}_{i(k+2)} \\ &\leqslant -\tilde{U}_{\mathrm{VPP}}(\mathbf{E}^{\mathrm{EX}\star}) + \\ &\sum\nolimits_i \left(\boldsymbol{\alpha}_{i(k+1)} + \rho(\boldsymbol{E}_{i(k+2)} - \boldsymbol{P}_{i(k+1)})\right)^\top \boldsymbol{E}_i^\star\end{aligned} \tag{32}$$

Combing Eq.(31) and Eq.(32), the bound can be obtained.

D. Proof for Inequalities (14)-(16):

Theorem 2. (*Minimum Principle*) Let a generic convex optimization problem be $\min_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x})$, where $f(\boldsymbol{x})$ is a convex and differentiable function from $\boldsymbol{x}$ to $\mathbb{R}$ and Ω is a closed convex set. A feasible point $\boldsymbol{x}^\star \in \Omega$ is an optimal solution if and only if

$$(\boldsymbol{x} - \boldsymbol{x}^\star)^\top \nabla f(\boldsymbol{x}^\star) \geqslant 0,\ \forall \boldsymbol{x} \in \Omega \tag{33}$$

The proof is based on the theorem above. For $\Delta\mathcal{L}_{\mathrm{PRO}}$:

$$\begin{aligned}&\Delta\mathcal{L}_{\mathrm{PRO}} = \mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)}) - \mathcal{L}(\mathbf{x}_{(k)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)}) \\ &\overset{(a)}{=} \sum_i -\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right) - \boldsymbol{\alpha}_{i(k)}^\top \boldsymbol{P}_{i(k+1)} + \frac{\rho}{2}\|\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k+1)}\|^2 \\ &\quad - \left(-\tilde{U}_i\left(\boldsymbol{x}_{i(k)}\right) - \boldsymbol{\alpha}_{i(k)}^\top \boldsymbol{P}_{i(k)} + \frac{\rho}{2}\|\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k)}\|^2\right) \\ &\overset{(b)}{=} \sum_{i\in\mathcal{A}_k} -\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right) - \boldsymbol{\alpha}_{i(k)}^\top \boldsymbol{P}_{i(k+1)} + \frac{\rho}{2}\|\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k+1)}\|^2 \\ &\quad - \left(-\tilde{U}_i\left(\boldsymbol{x}_{i(k)}\right) - \boldsymbol{\alpha}_{i(k)}^\top \boldsymbol{P}_{i(k)} + \frac{\rho}{2}\|\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k)}\|^2\right) \\ &\overset{(c)}{\leqslant} \sum_{i\in\mathcal{A}_k} \partial\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right)^\top \left(\boldsymbol{x}_{i(k)} - \boldsymbol{x}_{i(k+1)}\right) - \boldsymbol{\alpha}_{i(k)}^\top \Delta\boldsymbol{P}_{i(k+1,k)} \\ &\quad + \frac{\rho}{2}(\boldsymbol{P}_{i(k+1)} - \boldsymbol{P}_{i(k)})^\top(\boldsymbol{P}_{i(k+1)} + \boldsymbol{P}_{i(k)} - 2\boldsymbol{E}_{i(k+1)}) \\ &\overset{(d)}{\leqslant} -\sum_{i\in\mathcal{A}_k} \frac{\rho}{2}\left\|\boldsymbol{P}_{i(k+1)} - \boldsymbol{P}_{i(k)}\right\|^2\end{aligned}$$

where the reasoning is as follows: (a): the definition of the augmented Lagrangian function. (b): for $i \notin \mathcal{A}_k$, $\boldsymbol{x}_{i(k+1)} = \boldsymbol{x}_{i(k)}, \boldsymbol{P}_{i(k+1)} = \boldsymbol{P}_{i(k)}$. (c): use the convexity of $-\tilde{U}_i\left(\boldsymbol{x}_i\right)$ shown as follows:

$$\tilde{U}_i\left(\boldsymbol{x}_{i(k)}\right) - \tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right) \leqslant \partial\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right)^\top \left(\boldsymbol{x}_{i(k)} - \boldsymbol{x}_{i(k+1)}\right)$$

(d): since $\boldsymbol{x}_{i(k+1)}$ is the optimal solution, use Theorem 2 for $\forall \boldsymbol{x}_i, \boldsymbol{P}_i \in \Omega_i$:

$$\begin{aligned}&\left(-\partial\tilde{U}_i\left(\boldsymbol{x}_{i(k+1)}\right)\right)^\top \left(\boldsymbol{x}_i - \boldsymbol{x}_{i(k+1)}\right) + \\ &\left(-\boldsymbol{\alpha}_{i(k)}^\top + \rho(\boldsymbol{P}_{i(k+1)} - \boldsymbol{E}_{i(k+1)})\right)^\top \left(\boldsymbol{P}_i - \boldsymbol{P}_{i(k+1)}\right) \geqslant 0\end{aligned}$$

and replacing $\boldsymbol{x}_i$, $\boldsymbol{P}_i$ with $\boldsymbol{x}_{i(k)}$,$\boldsymbol{P}_{i(k)}$ prove Eq.(14).

For the second term $\Delta\mathcal{L}_{\mathrm{MUL}}$:

$$\begin{aligned}\Delta\mathcal{L}_{\mathrm{MUL}} =&\mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k+1)}) - \mathcal{L}(\mathbf{x}_{(k+1)}, \mathbf{E}_{(k+1)}, \boldsymbol{\alpha}_{(k)}) \\ &\overset{(a)}{=} \sum_i \left(\boldsymbol{\alpha}_{i(k+1)} - \boldsymbol{\alpha}_{i(k)}\right)^\top (\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k+1)}) \\ &\overset{(b)}{=} \sum_{i\in\mathcal{A}_k} \frac{1}{\rho}\left\|\boldsymbol{\alpha}_{i(k+1)} - \boldsymbol{\alpha}_{i(k)}\right\|^2\end{aligned}$$

where the reasoning is as follows:
(a): the definition of the augmented Lagrangian function. (b): use the updating rule for multipliers. For $i \in \mathcal{A}_k$, $\boldsymbol{\alpha}_{i(k+1)} - \boldsymbol{\alpha}_{i(k)} = \rho(\boldsymbol{E}_{i(k+1)} - \boldsymbol{P}_{i(k+1)})$; for $i \notin \mathcal{A}_k$, $\boldsymbol{\alpha}_{i(k+1)} = \boldsymbol{\alpha}_{i(k)}$.

For the third term $\Delta\mathcal{L}_{\mathrm{VPP}}$, similar results can be obtained just like $\Delta\mathcal{L}_{\mathrm{PRO}}$ and the proofs are omitted for simplicity.

E. Proof for Inequality (17)

The solution of the VPP's sub-problem is:

$$E_{i,t(k+1)}^{\mathrm{SH}} = P_{i,t(k)}^{\mathrm{SH}} - \frac{v_{i,t(k)}}{\rho} \underbrace{- \frac{\sum_i P_{i,t(k)}^{\mathrm{SH}}}{I} + \frac{\sum_i v_{i,t(k)}}{I\rho}}_{=\mu_{t(k)}/\rho}$$

$$E_{i,t(k+1)}^{\mathrm{EX}} = P_{i,t(k)}^{\mathrm{EX}} - \frac{w_{i,t(k)}}{\rho} \underbrace{- \left[\frac{\sum_i P_{i,t(k)}^{\mathrm{EX}}}{I} - \frac{\sum_i w_{i,t(k)}}{I\rho}\right]_{\frac{\lambda_t^{\mathrm{s}}}{\rho}}^{\frac{\lambda_t^{\mathrm{b}}}{\rho}}}_{=\nu_{t(k)}/\rho}$$

where the part in the brackets are the same for every prosumer i. For brevity, they are expressed as $\mu_{t(k)}/\rho$ and $\nu_{t(k)}/\rho$, respectively. For convenience, they are expressed in the vector form as $\boldsymbol{\beta}_{(k)} = [\boldsymbol{\mu}_{t(k)}^\top, \boldsymbol{\nu}_{t(k)}^\top]^\top$. Thereby:

$$\begin{aligned}&\sum\nolimits_i \|\Delta\boldsymbol{E}_{i(k+1,k)}\|^2 = \\ &\sum\nolimits_i \|\Delta\boldsymbol{P}_{i(k,k-1)} - \frac{\Delta\boldsymbol{\alpha}_{i(k,k-1)}}{\rho} + \frac{\Delta\boldsymbol{\beta}_{(k,k-1)}}{\rho}\|^2\end{aligned}$$

It can be proved that

$$\sum\nolimits_i \|\Delta\boldsymbol{P}_i - \frac{\Delta\boldsymbol{\alpha}_i}{\rho} - \boldsymbol{\zeta}\|^2 \geqslant \tag{34}$$

$$\sum\nolimits_i \|\Delta\boldsymbol{P}_i - \frac{\Delta\boldsymbol{\alpha}_i}{\rho}\|^2 - \frac{1}{I}\|\sum\nolimits_i \Delta\boldsymbol{P}_i - \frac{\Delta\boldsymbol{\alpha}_i}{\rho}\|^2 \tag{35}$$

where $\boldsymbol{\zeta}$ is any real vector in same the dimension with $\boldsymbol{P}_i$. Replace $\boldsymbol{\zeta}$ by $\Delta\boldsymbol{\beta}_i/\rho$ and the bound can be obtained.

REFERENCES

[1] Y. Parag and B. K. Sovacool, "Electricity market design for the prosumer era," *Nature Energy*, vol. 1, no. 4, 2016.

[2] S. M. Nosratabadi, R.-A. Hooshmand, and E. Gholipour, "A comprehensive review on microgrid and virtual power plant concepts employed for distributed energy resources scheduling in power systems," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 341–363, 2017.

[3] J. Wang, H. Zhong, J. Qin, W. Tang, R. Rajagopal, Q. Xia, and C. Kang, "Incentive mechanism for sharing distributed energy resources," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 4, pp. 837–850, 2019.

[4] A. Pena-Bello, D. Parra, M. Herberz, V. Tiefenbeck, M. K. Patel, and U. J. J. Hahnel, "Integration of prosumer peer-to-peer trading decisions into energy community modelling," *Nature Energy*, 2021.

[5] Y. Chen, C. Zhao, S. H. Low, and S. Mei, "Approaching prosumer social optimum via energy sharing with proof of convergence," *IEEE Transactions on Smart Grid*, vol. 12, no. 3, pp. 2484–2495, 2021.

[6] T. Sousa, T. Soares, P. Pinson, F. Moret, T. Baroche, and E. Sorin, "Peer-to-peer and community-based markets: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 104, pp. 367–378, 2019.

[7] Z. Guo, P. Pinson, Q. Wu, S. Chen, Q. Yang, and Z. Yang, "An asynchronous online negotiation mechanism for real-time peer-to-peer electricity markets," *IEEE Transactions on Power Systems*, 2022, in Press.

[8] E. Mashhour and S. M. Moghaddas-Tafreshi, "Bidding strategy of virtual power plant for participating in energy and spinning reserve markets—part i: Problem formulation," *IEEE Transactions on Power Systems*, vol. 26, no. 2, pp. 949–956, 2011.

[9] M. Vahedipour-Dahraie, H. Rashidizadeh-Kermani, M. Shafie-Khah, and J. P. S. Catalão, "Risk-averse optimal energy and reserve scheduling for virtual power plants incorporating demand response programs," *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1405–1415, 2021.

[10] R. Zhang and B. Hredzak, "Distributed dynamic clustering algorithm for formation of heterogeneous virtual power plants based on power requirements," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 192–204, 2021.

[11] Z. Yi, Y. Xu, X. Wang, W. Gu, H. Sun, Q. Wu, and C. Wu, "An improved two-stage deep reinforcement learning approach for regulation service disaggregation in a virtual power plant," *IEEE Transactions on Smart Grid*, vol. 13, no. 4, pp. 2844–2858, 2022.

[12] H. Yang, D. Yi, J. Zhao, and Z. Dong, "Distributed optimal dispatch of virtual power plant via limited communication," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 3511–3512, 2013.

[13] P. Li, Y. Liu, H. Xin, and X. Jiang, "A robust distributed economic dispatch strategy of virtual power plant under cyber-attacks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4343–4352, 2018.

[14] Q. Wang, W. Wu, B. Wang, G. Wang, Y. Xi, H. Liu, S. Wang, and J. Zhang, "Asynchronous decomposition method for the coordinated operation of virtual power plants," *IEEE Transactions on Power Systems*, pp. 1–1, 2022.

[15] N. Liu, X. Yu, C. Wang, and J. Wang, "Energy sharing management for microgrids with pv prosumers: A stackelberg game approach," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1088–1098, 2017.

[16] J. Li, C. Zhang, Z. Xu, J. Wang, J. Zhao, and Y.-J. A. Zhang, "Distributed transactive energy trading framework in distribution networks," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 7215–7227, 2018.

[17] S. Cui, Y.-W. Wang, Y. Shi, and J.-W. Xiao, "An efficient peer-to-peer energy-sharing framework for numerous community prosumers," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7402–7412, 2020.

[18] J. Li, Y. Ye, D. Papadaskalopoulos, and G. Strbac, "Computationally efficient pricing and benefit distribution mechanisms for incentivizing stable peer-to-peer energy trading," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 734–749, 2021.

[19] H. Sheng, C. Wang, X. Dong, K. Meng, and Z. Dong, "Incorporating P2P trading into DSO's decision-making: A DSO-prosumers cooperated scheduling framework for transactive distribution system," *IEEE Transactions on Power Systems*, 2022, in Press.

[20] Y. Chen, Y. Yang, and X. Xu, "Towards transactive energy: An analysis of information-related practical issues," *Energy Conversion and Economics*, vol. 3, no. 3, pp. 112–121, 2022.

[21] K. Umer, Q. Huang, M. Khorasany, M. Afzal, and W. Amin, "A novel communication efficient peer-to-peer energy trading scheme for enhanced privacy in microgrids," *Applied Energy*, vol. 296, p. 117075, 2021.

[22] J. Wang, H. Zhong, Q. Xia, G. Li, and M. Zhou, *Information and Communication Technology for Sharing Economy*. Singapore: Springer Singapore, 2022, pp. 271–318.

[23] F. Moret, T. Baroche, E. Sorin, and P. Pinson, "Negotiation algorithms for peer-to-peer electricity markets: Computational properties," in *Power Systems Computation Conference (PSCC)*, 2018, pp. 1–7.

[24] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh, "Machine-type communications: current status and future perspectives toward 5G systems," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 10–17, 2015.

[25] S. K. Sharma and X. Wang, "Toward massive machine type communications in ultra-dense cellular IoT networks: Current issues and machine learning-assisted solutions," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 426–471, 2020.

[26] R. Sun, Z.-Q. Luo, and Y. Ye, "On the efficiency of random permutation for admm and coordinate descent," *Mathematics of Operations Research*, vol. 45, no. 1, pp. 233–271, 2020.

[27] C. Chen, B. He, Y. Ye, and X. Yuan, "The direct extension of admm for multi-block convex minimization problems is not necessarily convergent," *Mathematical Programming*, vol. 155, no. 1, pp. 57–79, 2016.

[28] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2233–2246, 2012.

[29] A. Goncalves, X. Liu, and A. Banerjee, "Two-block vs. multi-block admm: An empirical evaluation of convergence," *arXiv preprint arXiv:1907.04524*, 2019.

[30] D. Gebbran, S. Mhanna, A. C. Chapman, and G. Verbič, "Multiperiod der coordination using admm-based three-block distributed ac optimal power flow considering inverter volt-var control," *IEEE Transactions on Smart Grid*, 2022.

[31] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[32] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed admm for large-scale optimization—part i: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[33] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[34] M. S. Ali, E. Hossain, and D. I. Kim, "LTE/LTE-A random access for massive machine-type communications in smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 76–83, 2017.

[35] W. Zhan and L. Dai, "Access delay optimization of M2M communications in LTE networks," *IEEE wireless communications letters*, vol. 8, no. 6, pp. 1675–1678, 2019.

[36] A. V. Fiacco, "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical programming*, vol. 10, no. 1, pp. 287–311, 1976.

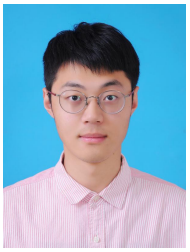
[37] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[38] Irish Social Science Data Archive, "Commission for energy regulation (cer) smart metering project," https://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

[39] UK Data Service, "Low carbon london project: Data from the dynamic time-of-use electricity pricing trial." https://beta.ukdataservice.ac.uk/datacatalogue/studies/study?id=7857.

Cheng Feng received the B.S. degree from the Department of Electrical Engineering in Huazhong University of Science and Technology in 2019. He is currently pursuing Ph.D. degree in Tsinghua University. From February 2023, he is a visiting scholar in Automatic Control Lab (ifA), ETH Zurich, Switzerland. His research interests include virtual power plants, cyber-physical systems data analytics in smart grids.

Kedi Zheng received the B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2017 and 2022, respectively. He is currently a Postdoctoral Researcher with Tsinghua University. His research interests include data analytics in power systems and electricity markets.



Yangze Zhou is expected to receive B.Eng. Degree in the College of Energy Engineering, at Zhejiang University in 2023. His research includes data analytics and data valuation in energy systems, energy load forecasting, and fault detection and diagnosis.

Peter Palensky received the M.Sc. degree in electrical engineering and the Ph.D. and Habilitation degrees from the Vienna University of Technology, Austria, in 1997, 2001, and 2015, respectively. He co-founded a Envidatec, a German startup on energy management and analytics, and joined the Lawrence Berkeley National Laboratory, Berkeley, CA, USA, as a Researcher and the University of Pretoria, South Africa, in 2008. In 2009 he became an Appointed Head of Business Unit on Sustainable Building Technologies with the Austrian Institute of Technology (AIT) and later the First Principle Scientist for Complex Energy Systems with AIT. In 2014, he was appointed as a Full Professor of Intelligent Electric Power Grids, TU Delft. His main research fields are energy automation networks, smart grids, and modeling intelligent energy systems. He is active in international committees like ISO or CEN and serves as IEEE IES AdCom member-at-large in various functions for the IEEE. He is the Editor in Chief of the IEEE Industrial Electronics Magazine and an Associate Editor for several other IEEE publications, and regularly organizes IEEE conferences.

Qixin Chen received the Ph.D. degree from the Department of Electrical Engineering, Tsinghua University, Beijing, China, in 2010. He is currently a tenured professor with Tsinghua University. His research interests include electricity markets, power system economics and optimization, low-carbon electricity, and data analytics in power systems.