Data-Driven Flexibility Capability Modeling of Internet Data Center Considering Task Dependency

Jiahao Ma, *Graduate Student Member, IEEE*, Ruiyang Yao, *Graduate Student Member, IEEE*, Bochao Zhang, Zhaoyang Wang, and Yuejun Yan

***Abstract*—The power consumption flexibility provided by the energy-intensive Internet data centers (IDCs) has been extensively studied as a potential solution for enhancing the flexibility of power systems. In IDCs, computational workloads are further divided into potentially interdependent tasks. To assess the power consumption flexibility of IDCs, it is necessary to consider the interdependency of computational tasks. However, there are no methods for deriving a task dependency-aware IDC load model that is easy to embed in the operation of power systems to fully utilize the power consumption flexibility of IDCs. To this end, this article proposes a framework to derive a compatible task dependency-aware IDC load model. A linear IDC load model is formulated based on typical batch workloads given by a task dependency-aware clustering framework. Afterward, the cost-oriented progressive vertex enumeration (COPVE) algorithm is proposed to derive an easy-to-embed IDC load model from the original linear model. Experiments show that the derived IDC load model accurately reflects the feasible region of the original IDC load model with fewer constraints compared with the model derived by the advanced progressive vertex enumeration (PVE) algorithm.**

***Index Terms*—Aggregation, demand response, Internet data center (IDC), task dependency.**

I. Introduction

INTERNET data centers (IDCs) are the backbones of the modern information industry. With the rapid development of the digital economy, the number and size of IDCs have experienced explosive growth, resulting in an unprecedented surge in electric power consumption. IDCs consume approximately 1.5% of the world's total electricity, and their electric power demand is expected to increase by 15%–20% annually [1]. The power consumption devices in IDCs can be classified into three parts: 1) information technology (IT) equipment; 2) cooling systems; and 3) other auxiliary equipment such as lighting [2]. To satisfy the ever-increasing computation requests from various Internet services, IDCs assign and schedule numerous IT devices to process incoming workloads, and cooling systems are deployed to absorb the heat generated during the operation. The IT equipment and the cooling system consume around 80%–90% of the power consumption in IDCs [3]. Because the power consumption of IT equipment and the cooling demand are closely related to the processing of workloads, the characterization of workloads is the key factor for modeling the power consumption of IDCs [4].

Based on the latency tolerance and computation usage, workloads can be classified into interactive workloads such as online search or payment, and batch workloads, such as data mining and machine learning [5]. Generally, online workloads are random, sensitive to latency, and require fewer computation resources. In contrast, batch workloads are computationally expensive and insensitive to latency [6] and contribute to about 70% of the total power consumption in IDCs [7]. Due to the flexible execution time and high energy demands, batch workloads play a vital role in the flexibility assessment of power consumption in IDCs.

The flexibility modeling of IDC and its application have attracted increasing attention. Researchers investigated different flexibility modeling, including batch workload scheduling [8], the thermal inertia of machine rooms [9], and the self-built storage in the IDC microgrid [10]. For the application, the flexibility of IDC power consumption has been researched to reduce the risk of supply–demand imbalance caused by renewable energy generation [11], achieve better planning of regional integrated electricity–heat systems [12], and improve the operation of power systems [13].

However, most existing studies oversimplified the characterization of workloads. In particular, most works treat the batch workload as the basic scheduling unit. However, in modern IDC scheduling systems, a batch workload is usually further divided into smaller units, including jobs and tasks to improve computing parallelism [14]. While parallel computing enhances the utilization of IT equipment and potentially improves the flexibility of IDC power consumption [15], data dependencies exist between tasks: the subsequent task must wait until the preceding task is finished [16]. Ignoring the dependency relationship in the load modeling will lead to overestimation of the flexibility

Manuscript received 14 March 2024; revised 24 April 2024; accepted 24 April 2024. Date of publication 8 May 2024; date of current version 9 July 2024. This work was supported by the Alibaba Innovative Research Programme. *(Corresponding author: Yuejun Yan.)*

Jiahao Ma and Ruiyang Yao are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, SAR, China (e-mail: jhma@connect.hku.hk; ryyao@connect.hku.hk).

Bochao Zhang is with the Weiyang College, Tsinghua University, Beijing 100084, China (e-mail: zbc21@mails.tsinghua.edu.cn).

Zhaoyang Wang and Yuejun Yan are with Alibaba Group, Hangzhou, China (e-mail: chaoyang.wzy@alibaba-inc.com; yanyuejun.yyj@alibaba-inc.com).

Digital Object Identifier 10.1109/JIOT.2024.3395837

by assuming infeasible execution sequences of tasks, and it is crucial to formulate a task dependency-aware IDC load model.

To deal with the interdependency information, some researchers developed IDC load models that considered the task dependency [17]. However, the proposed load model contained nonlinear terms and numerous binary variables, making it difficult to embed in power system operations. To simplify the scheduling of IDC power consumption for the power system, it is critical to derive a compatible IDC load model. Such a model contains only the temporally coupled constraints of external power consumption variables and is the aggregation of the IT equipment, cooling systems, and auxiliary equipment in the IDC microgrid [18].

The derivation of the compatible IDC load model is mathematically equivalent to the aggregate temporally coupled feasible region of distributed energy resources. Chen et al. [19] pointed out that calculating the aggregate temporally coupled feasible region is mathematically identical to calculating the Minkowski sum of multiple high-dimensional polytopes, for which no efficient exact solution method exists. Therefore, recent research focuses on deriving inner approximation [20] or outer approximation [21] of the feasible region. However, the inner approximation might be too conservative and the outer approximation might include load profiles infeasible to disaggregate [22]. To alleviate the dilemma, the progressive vertex enumeration (PVE) algorithm [23] is proposed to obtain the feasible region by continuously expanding the inner approximation and is proven to get the exact solution after finite iterations [24]. However, the application of the PVE algorithm in deriving temporally coupled feasible regions includes the calculation of a high-dimensional convex hull, which is difficult with a large number of vertices. Therefore, further research on the derivation method of the temporally coupled feasible region is needed.

To the best of the authors' knowledge, there is no published research on deriving a compatible task dependency-aware IDC load model. To address the aforementioned research gap, this article formulates a linear task dependency-aware IDC load model and proposes the cost-oriented PVE (COPVE) algorithm to efficiently derive the compatible IDC load model. The contribution of this article is threefold.

1) We propose a novel framework to derive the compatible task dependency-aware IDC load model. Two major benefits can be achieved: a) the flexibility capability of IDC power consumption is modeled while considering the task dependency relationship and b) the compatible IDC load model is easy to embed in the operation of power systems.
2) We propose the COPVE algorithm that overcomes the application difficulty of the PVE algorithm in deriving the temporally coupled feasible region. Compared with the model derived by the PVE algorithm, the compatible load model derived by the proposed algorithm can describe the original load model more accurately with fewer constraints and is easier to embed.
3) We propose a data-driven method to formulate a linear task dependency-aware IDC load model. The deep graph-temporal clustering framework (DeGTeC) is applied to cluster numerous batch jobs while considering task dependency. The formulated IDC load model has low complexity and can describe the IDC power consumption characteristics while considering the task dependency relationship.

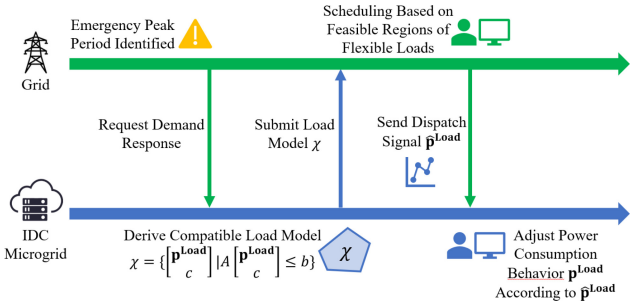

Fig. 1. Process of IDC participating in demand response initiated by grid operators.

The remainder of this article is structured as follows. Section II defines the problem of deriving a compatible IDC load model and the proposed framework. Section III introduces DeGTeC, a task dependency-aware batch job clustering framework. Section IV formulates the linear IDC load model. Section V elaborates on the proposed COPVE algorithm to derive the compatible IDC load model. Section VI reports experimental results and analysis. Section VII concludes this article.

II. Problem Statement and Framework

A. Problem Statement

The process of IDC participating in demand response is shown in Fig. 1. Once an emergency peak period is identified by grid operators, a demand response request will be announced to users including IDC operators. Next, each user will calculate its feasible region during this period and submit it to grid operators. By co-optimizing the operation of the grid and flexible loads, grid operators will then give the reference power consumption signal to each user. After that, users should adjust their power consumption behavior to the signal. If there is too much deviation in the power consumption, the users will be required to pay a penalty. To ensure the compatible IDC load model is easy to embed in power system operations, a linear model is considered in this work and is formulated as follows:

$$\mathcal{X} = \left\{x = \left(\mathbf{p}^{\mathbf{Load}}, c\right) \in \mathbb{R}^{T+1} | Ax \leq b\right\} \tag{1}$$

where $\mathcal{X}$ denotes the feasible region of IDC power consumption and its corresponding scheduling cost; x is the corresponding decision variables, $\mathbf{p}^{\mathbf{Load}}$ is the power consumption vector of the IDC microgrid within T timesteps, and c is the corresponding scheduling cost; A and b, respectively, refer to the coefficient matrix and vector of linear constraints describing the feasible region $\mathcal{X}$.

To model the flexibility of IDC load, batch workloads are important due to their computationally intensive requirements and the flexible processing time. During the processing,

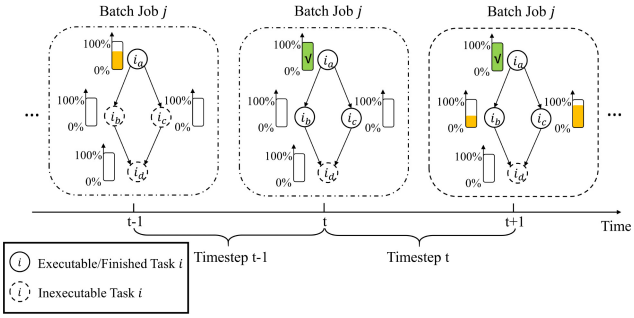

Fig. 2. Illustration of the dependency relationship between tasks in the jth batch job. At time t-1, task i_a is not finished and tasks i_b and i_c are inexecutable. After the timestep t-1, task i_a is finished and tasks i_b and i_c become executable.

batch workloads are divided into several dependent segments with specific meanings. Segments with dependency must be executed in sequence. In task scheduling systems like Google's Borg [25] and Alibaba's Fuxi [26], batch workloads are divided into jobs, and each job is divided into several tasks. In one batch job, a dependency relationship exists between its component tasks. In Alibaba's Fuxi scheduling system, each task is further separated into several independent instances during execution [26].

To consider the heterogeneous computation usage and dependency relationship of component tasks in each batch job, the task dependencies of batch jobs can be modeled as directed acyclic graphs (DAGs) when scheduling. As shown in Fig. 2, the dependency relationship between two components tasks can be formulated as follows:

$$\left\lfloor \sum_{\tau=1}^{t-1} n_{j,i_a,\tau} \right\rfloor \geq \sum_{\tau=1}^{t} n_{j,i_b,\tau}, \text{ if } i_a \rightarrow i_b \quad \forall j,t \tag{2}$$

where $n_{j,i,\tau}$ is the completion percentage of the ith task in the jth job during the τth timestep; $\lfloor \cdot \rfloor$ is the flooring function which rounds down the input; $i_a \rightarrow i_b$ means that the execution of the i_bth task depends on the completion of the i_ath task. When the i_ath task is not finished, the total completion percentage of this task is smaller than 100% and the flooring function gives out 0 to stop the i_bth task from being processed. Once the i_ath task is finished after timestep t-1, the flooring function gives out 1 and the processing of the i_bth task is allowed.

The formulation of (2) presents two challenges when deriving the compatible IDC load model, namely, the high complexity of the original IDC load model and the consideration of numerous heterogeneous DAG topologies in batch jobs. On the one hand, the reformulation of the flooring function brings binary variables to the original load model. To eliminate the nonlinear flooring function, (2) can be reformulated as follows:

$$\sum_{\tau=1}^{t-1} n_{j,i_a,\tau} \geq x_{j,i_a,t} \tag{3a}$$

$$x_{j,i_a,t} \geq \sum_{\tau=1}^{t} n_{j,i_b,\tau}, \text{ if } i_a \rightarrow i_b \quad \forall j,t \tag{3b}$$

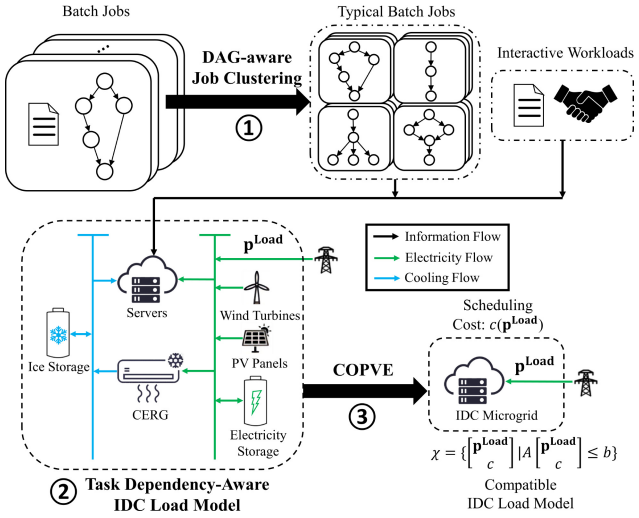

Fig. 3. Structure of the proposed framework. In step 1, DAG-aware job clustering is used to get typical batch jobs. In step 2, the task dependency-aware IDC load model is formulated. In step 3, the compatible IDC load model is derived via the proposed COPVE algorithm.

where $x_{j,i_a,t}$ is the binary variable imported by the reformulation of the flooring function. However, a single IDC receives numerous batch workloads during its operation. For example, the IDC of Alibaba received around 20 000 batch jobs in an hour on average [26], bringing a large number of binary variables and inequalities to the load model.

On the other hand, when considering (2), neither a predefined equivalent energy model (such as a virtual generator or virtual storage) nor a bottom-up analytical approach is suitable for deriving the compatible IDC load model. The performance of a predefined energy model is not guaranteed for all kinds of DAG topologies. Moreover, different forms of inequalities caused by heterogeneous DAG topologies hinder the analytical deduction of the IDC load model.

B. Proposed Framework

To tackle the challenges imported by (2), the proposed framework is shown as Fig. 3. In step 1, DeGTeC [27] is applied to cluster numerous DAG-aware batch jobs. Typical batch jobs are given by DeGTeC based on both the computation usage features and the topology of DAG.

Step 1 tackles the first challenge and brings two advantages. On the one hand, the number of inequalities is reduced, decreasing the complexity of the IDC load model. On the other hand, the relaxation of (2) introduces fewer errors and is acceptable after clustering. The relaxation of (2) is formulated as

$$\sum_{\tau=1}^{t-1} n_{j,i_a,\tau} \geq \sum_{\tau=1}^{t} n_{j,i_b,\tau}, \text{ if } i_a \rightarrow i_b \quad \forall j,t. \tag{4}$$

After clustering, the domain of $n_{j,i,\tau}$ expands from [0, 1] to [0, $\lambda_{j,t}$], where $\lambda_{j,t}$ is the arrival rate of the jth job during the tth timestep. When the number of batch workloads is substantial, the difference between $\sum_{\tau=1}^{t-1} n_{j,i_a,\tau}$ and $\lfloor \sum_{\tau=1}^{t-1} n_{j,i_a,\tau} \rfloor$ is small relative to $\sum_{\tau=1}^{t-1} n_{j,i_a,\tau}$.

The original IDC load model consists of all external and internal decision variables. After the job clustering and relaxation, in step 2, it is modeled as a linear programming (LP) problem and is formulated as

$$\Phi = \left\{(x, y) \in \mathbb{R}^{T+1} \times \mathbb{R}^{n} | Ax + By \leq b\right\} \tag{5}$$

where the feasible region Φ is a polyhedron in space $\mathbb{R}^{T+1} \times \mathbb{R}^{n}$; y denote other internal decision variables in IDC load model. In this case, $\mathcal{X}$ is the projection of polyhedron Φ on the subspace $\mathbb{R}^{T+1}$.

In step 3, the COPVE algorithm is proposed to tackle the second challenge. The algorithm is applied to the original IDC load model to derive the compatible IDC load model $\mathcal{X}$. Instead of predefined energy models, the algorithm does not assume the form of the energy model and is adaptive to various DAG topologies. Besides, the algorithm avoids the analytical deduction by continuously expanding an inner approximation of $\mathcal{X}$.

III. Deep Graph-Temporal Clustering Framework

DAG-aware job clustering categorizes batch jobs into several groups according to the resource usage of each component task, such as the central processing unit (CPU) utilization and the DAG topology. However, DAG-aware job clustering is difficult by adopting typical clustering methods such as K-Means. Based on the Euclidean distance, these methods require the representation of each batch job as vectors with identical sizes, but DAG is a typical non-Euclidean object. On the one hand, the number of tasks is not fixed for all jobs, making it difficult to represent the resource usage vector with identical sizes. On the other hand, the difference between the two graphs is hard to measure by adopting the Euclidean distance.

To overcome the challenges mentioned above, the clustering framework named DeGTeC is proposed [27]. As mentioned in [28], a popular assumption for data analysis in non-Euclidean objects is that although the original objects are high dimensional, they usually reside in low-dimensional subspaces. With the help of two autoencoders, namely, task autoencoder (TaskAE) and job autoencoder (JobAE), the resource usage and dependency relationship features of DAG batch jobs are embedded into a low-dimensional latent space. Using feature vectors in the latent space, all jobs are represented by vectors with a fixed size and can be further clustered by methods based on the Euclidean distance.

A. Overview of DeGTeC

Autoencoder is a common structure to learn the latent representation of input feature vectors. Usually, an autoencoder consists of two deep neural networks with similar structures but opposite forward directions, namely, the encoder and decoder. The encoder tries to represent all information of the input vector in the latent space, while the decoder tries to reconstruct the original input vector from the latent space. By minimizing the reconstruction loss, the weights of the encoder and decoder can be learned.

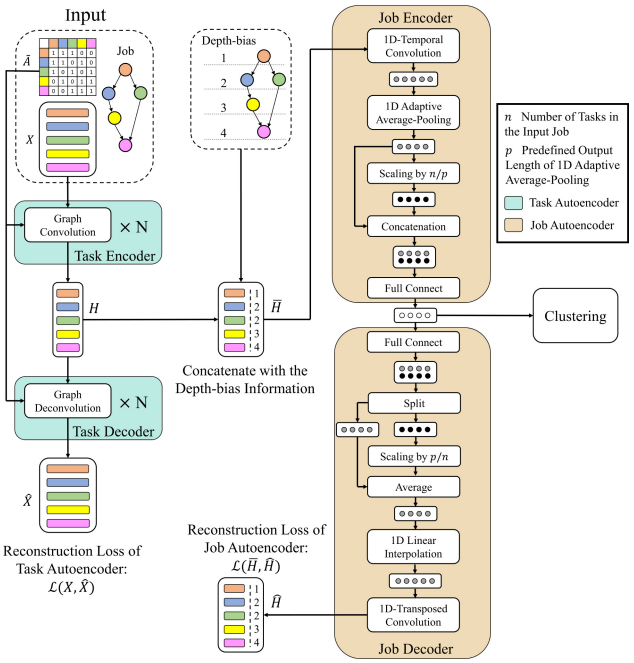

Fig. 4. Overview of the framework of DeGTeC. DeGTeC consists of two autoencoders, namely, TaskAE and JobAE, and can output latent representations with a fixed size.

The overview of the DeGTeC is shown in Fig. 4. DeGTeC has two autoencoders, namely, TaskAE and JobAE. As the first autoencoder, TaskAE focuses on representing the task. The graph convolutional (GC) layers are used in TaskAE to derive the latent embedding of each job by synthesizing the attributes of each component task and the graph context. As the second autoencoder, JobAE includes a 1-D temporal convolutional (1D-TC) layer, an adaptive average pooling (AAP) layer, a scaling layer, and a fully connected (FC) layer to learn the latent representation of the job. With the help of an adaptive pooling layer, JobAE can output latent representations with a fixed size. Finally, all latent representations can be categorized using clustering algorithms based on the Euclidean distance like K-Means.

B. Task Autoencoder

TaskAE adopts the GC autoencoder using Laplacian smoothing and sharpening (GALA) [28] to learn the latent representation of a graph. For a job DAG $G = (V, E)$ whose tasks are nodes denoted as $v_i \in V$ and dependency relationship are edges denoted as $(v_i, v_j) \in E$, note the matrix of task feature vectors as $X \in \mathbb{R}^{n \times d}$ and the binary adjacency matrix $A \in \mathbb{Z}^{n \times n}$, where n is the number of tasks; d is the dimension of task features; $A_{i,j} = 1$ if $\exists (v_i, v_j) \in E$. $D \in \mathbb{Z}^{n \times n}$ is a diagonal degree matrix with $D_{ii} = \sum_{j=1}^{n} A_{ij}$. To apply spectral graph convolution, DAGs are treated as undirected graphs in TaskAE and A is a symmetric adjacency matrix here. The direction information is embedded into the latent space after TaskAE.

1) Graph Convolution Layer and TaskAE Encoder: The spectral convolution on a graph is defined as follows. Note the filter of the convolution is $g_\theta = \mathrm{diag}(\theta)$ parameterized by

$\theta \in \mathbb{R}^n$. The input of the spectral convolution is X and the output is defined as

$$g_\theta * X = U g_\theta U^T X \tag{6}$$

where U is the matrix of eigenvectors of the symmetric normalized Laplacian $L_{\text{sym}} = I_n - D^{-1/2} A D^{-1/2}$.

However, the calculation of eigenvectors is computationally expensive. To reduce complexity, GALA approximated the operation of (6) by truncated Chebyshev polynomials up to first order

$$g_\theta * X \approx \theta\left(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) X. \tag{7}$$

To avoid numerical instabilities, GALA renormalized $I_n + D^{-1/2} A D^{-1/2}$ into $\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$ with $\widetilde{A} = A + I_n$ and $\widetilde{D}_{ii} = \sum_{j=1}^{n} \widetilde{A}_{ij}$. Finally, the operation of a graph convolution layer is simplified as

$$H^{(m+1)} = \xi\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}} H^{(m)} \Theta^{(m)}\right) \tag{8}$$

where $H^{(m)}$ is the $d^{(m)}$-channel input of the mth graph convolution layer and $H^{(0)} = X$; $\Theta^{(m)} \in \mathbb{R}^{d^{(m)} \times d^{(m+1)}}$ is the weight matrix of the mth layer; $\xi(\cdot)$ is the rectified linear unit (ReLU) activation function.

2) TaskAE Decoder: The decoder acts as the counterpart of the encoder and performs graph deconvolutions to reconstruct the original input X. The spectral graph deconvolution layer is defined as

$$H^{(m+1)} = \xi\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}} H^{(m)} \Theta^{(m)}\right) \tag{9}$$

where $H^{(m)}$ is the $d^{(m)}$-channel input of the mth graph convolution layer; $\hat{A}$ and $\hat{D}$ are defined as $\hat{A} = 2I_n - A$ and $\hat{D} = 2I_n + D$; $\Theta^{(m)} \in \mathbb{R}^{d^{(m)} \times d^{(m+1)}}$ is the weight matrix of the mth layer; $\xi(\cdot)$ is the ReLU activation function.

C. Sorting Task Embeddings

TaskAE adopts spectral graph convolution that treats DAG as an undirected graph. However, the depth information of each task is critical when characterizing feasible execution sequences of the job. The depth of the ith task is defined as the length of the longest path from the initial task to the ith task.

In DeGTeC, the depth information is embedded into the latent space directly by sorting task embedding (SortEM). For each job, SortEM concatenates the depth information of each task and the latent representation given by TaskAE. Next, SortEM sorts all task embeddings in ascending order based on the start time of each task. For a matrix $H \in \mathbb{R}^{n \times d}$ given by TaskAE, the output SortEM is

$$\overline{H} = \text{sort}\left(\{\text{concat}(H_i, \text{depth}(i))\}_{i=1}^{n}\right) \tag{10}$$

where depth$(\cdot)$ returns the depth of each task; concat$(\cdot)$ means the operation of concatenation; sort$(\cdot)$ means the operation of ascending-order sorting based on the start time of each task.

D. Job Autoencoder

JobAE is adopted to learn the latent temporal representation of DAGs. The input of JobAE is the sorted embedded sequences of tasks $\overline{H}$.

1) JobAE Encoder: To exploit the temporal characteristic of DAGs, the first layer of JobAE is a 1D-TC layer.

To learn the fixed-size representation of each job, the second layer of JobAE is a 1-D AAP layer. Given a predefined output length p, the layer adjusts the pooling kernel size to handle varying input sizes of jobs n. In the DeGTeC, p is set to be the median number of component tasks.

However, the AAP is too coarse-grained for long or short job sequences. To strengthen the effect of job length, a scaling layer is designed to make a copy of the adaptive pooling result and scale it by multiplying n/p. Suppose the output of adaptive pooling is l-channel and the shape is (p, l), after the scaling, an embedding of shape $(p, 2l)$ is derived.

Finally, an FC layer is adopted to synthesize the embedding and derive a vector with a fixed size in the latent space.

2) JobAE Decoder: The JobAE decoder has a reverse structure compared with the JobAE encoder and operates full connection, descaling (DS), interpolation (IP), and transposed convolution (1D-TPC) [29] to construct the original input given by SortEM. In the DS operation, the input embedding with $(p, 2l)$ shape is divided into two embeddings E, E' with (p, l) shape and an output embedding $\overline{E}$ with (p, l) shape is derived by doing a weight averaging: $\overline{E} = 0.5E + (0.5/n/p)E'$. Then, a linear IP is adopted to recover $\overline{E}$ to its original length and a 1-D transposed convolution is adopted to reconstruct $\overline{H}$.

E. Training and Clustering

In practice, the two autoencoders in DeGTeC are trained level by level. Autoencoder TaskAE is first trained by optimizing the reconstruction loss of the input feature matrix X. After the reconstruction loss converges, the parameters of TaskAE are fixed and JobAE is trained by optimizing the reconstruction loss of the sorted embedded sequences of tasks $\overline{H}$. Finally, the K-means clustering algorithm is applied to the latent vectors given by TaskAE. The job nearest to the centroids given by K-Means is chosen as the typical batch job for this group.

IV. Task Dependency-Aware IDC Load Model

The IDC load model is formulated as a combination of a workload processing model and an energy hub model. In the server model, incoming online and batch workloads are processed by servers while satisfying the constraints of task dependency. In the energy hub model, the operation of the IDC microgrid is considered as the coupling of electricity and cooling system.

A. Workload Processing Model

Typical batch jobs given by DeGTeC are considered to represent the required computation resources and task dependency relationship of all incoming batch jobs.

1) Task Dependency-Aware Batch Workload Model: According to Section II, the task dependency relationship between the i_ath and i_bth task is formulated as (4). However, when the timestep is longer than the duration of the task, it is possible for the execution of the i_ath and i_bth tasks in the

same timestep. Therefore, the task dependency relationship is formulated as follows:

$$\sum_{\tau=1}^{t-1} n_{j,i_a,\tau} + \rho_j n_{j,i_a,t} \geq \sum_{\tau=1}^{t} n_{j,i_b,\tau}, \text{ if } i_a \rightarrow i_b \quad \forall j, t \tag{11}$$

where ρ_j represents the percentage of maximal simultaneous execution number of i_ath and i_bth tasks in one timestep.

The transition equation of the accumulated task is formulated as follows:

$$\hat{n}_{j,i,t+1} = \hat{n}_{j,i,t} - n_{j,i,t} + \lambda_{j,t} \quad \forall j, i \in \Omega_j, t \tag{12}$$

where $\hat{n}_{j,i,t}$ represents the number of the ith accumulated task of the jth job; $\lambda_{j,t}$ denotes the arrival rate of the jth batch job; Ω_j is the set of all component tasks in the jth batch job. When $n_{j,i,t}$ is smaller than $\lambda_{j,t}$, the processing of some batch workloads arriving during the tth timestep is postponed to a later period. When $n_{j,i,t}$ is larger than $\lambda_{j,t}$, servers begin the processing of the batch workloads that arrived before the tth timestep but were interrupted at that time.

To make sure all batch jobs meet their deadlines, servers must not accumulate too many tasks in each timestep. Besides, the maximal number of finished batch jobs is constrained by the forecasted arrival rate

$$0 \leq \hat{n}_{j,i,t} \leq \hat{n}_{j,t}^{\max} \quad \forall j, t \tag{13}$$

where $\hat{n}_{j,t}^{\max}$ denotes the maximal number of accumulated tasks of the jth job.

2) Server Model: The batch task is modeled to be processed in servers of IDC, which is formulated as follows:

$$\sum_{s=1}^{S} \chi_{s,j,i,t} = n_{j,i,t} \quad \forall j, i \in \Omega_j, t \tag{14}$$

where $\chi_{s,j,i,t}$ is the number of the ith tasks in the jth job processed by the sth server during the tth timestep.

In this work, the required resources by tasks include average CPU utilization $U_{j,i}^{\text{task}}$ and the duration of the calculation time $T_{j,i}^{\text{task}}$. Complying with the characterization in IDCs of Google [25], the compute usage (CU) of one task is characterized by the multiple of $U_{j,i}^{\text{task}}$ and $T_{j,i}^{\text{task}}$ whose unit is CPU-hours

$$L_{j,i}^{\text{task}} = U_{j,i}^{\text{task}} T_{j,i}^{\text{task}} \quad \forall j, i \in \Omega_j \tag{15}$$

where $L_{j,i}^{\text{task}}$ is the CU of the ith task in the jth job. For example, a 10 CPU-hour task might have consumed ten machines with 100% CPU utilization for an hour, one machine with 100% CPU utilization for 10 h, or 100 machines with 10% CPU utilization for an hour, etc.

The CPU utilization of a server for processing tasks is formulated as

$$u_{s,j,i,t}^{\text{task}} = \chi_{s,j,i,t} \frac{L_{j,i}^{\text{task}}}{\Delta T} \quad \forall s, j, i \in \Omega_j, t \tag{16}$$

where $u_{s,j,i,t}^{\text{task}}$ is the CPU utilization of the sth server for processing the ith task in the j-job during the tth timestep; ΔT is the length of the timestep.

Considering the CU of the online workload $U_{s,t}^{\text{o}}$, the total CPU utilization of a server is formulated as

$$u_{s,t} = \sum_{j=1}^{J} \sum_{i \in \Omega_j} u_{s,j,i,t}^{\text{task}} + U_{s,t}^{\text{o}} \leq 100\% \quad \forall s, t \tag{17}$$

where $u_{s,t}$ is the CPU utilization of the sth server during the tth timestep.

Although workload resource usage includes CPU, RAM, disk usage, etc., the power consumption of a server can be accurately estimated using only CPU utilization [30]. In this work, the power consumption of a server is assumed to linearly increase with the CPU utilization increasing

$$p_{s,t}^{\text{IT}} = C_s u_{s,t} + p_s^{\text{fix}} \quad \forall s, t \tag{18}$$

where $p_{s,t}^{\text{IT}}$ is the consumption of the sth server during the tth timestep; C_s is the consumption coefficient for the CPU utilization; p_s^{fix} is the static power of the sth server.

3) Comparison With Existing Models in Assumption: To the best of our knowledge, the server model in [17] is the only existing model considering task dependency, whose assumption is different from the proposed model. Specifically, the required completion time and the arriving time of each batch task were assumed to be fixed in [17]. Then, a data-driven way was proposed to calculate each task's slack time, namely, the maximum deferrable time to avoid postponing the process of downstream tasks. The flexibility capability of IDC was evaluated based on the slack time of each task. However, this assumption underestimates the flexibility of IDC power consumption because the time required to complete the task can be modified in the real case [25], bringing more flexibility to the power consumption.

Compared with [17], the model of CU in (15) considers this flexibility and is consistent with [25]. The errors in the proposed framework come from the aggregation of typical jobs. It is assumed that batch jobs can be accurately represented by typical batch jobs, but differences exist in each cluster. Besides, the relaxation proposed in (4) will further bring errors.

B. Energy Hub Model

The IDC microgrid should simultaneously satisfy the electricity demand for servers and the cooling demand for maintaining the indoor temperature of machine rooms. As shown in Fig. 3, the microgrid contains self-built photovoltaic panels, wind turbines, lithium batteries for electricity storage, ice cooling storage, compression electric refrigerator group (CERG), and IDC servers. The formulation is detailed as follows.

1) Storage Devices: The transition of state-of-charges (SoC) variables of storage devices is formulated as

$$\text{SoC}_{g,t+1} = (1 - \gamma_g)\text{SoC}_{g,t} + \Delta T\left(p_{g,t}^{\text{ch}} \eta_g^{\text{ch}} - p_{g,t}^{\text{dch}} / \eta_g^{\text{dch}}\right) \quad \forall g \in \Omega_{\text{ST}}, t \tag{19}$$

where $\text{SoC}_{g,t}$ is the SoC of the gth storage at the tth timestep; γ_g is the self-discharge rate of the gth storage; $p_{g,t}^{\text{ch}}$ and $p_{g,t}^{\text{dch}}$ are the charging and discharging power of the gth storage during

the tth timestep, respectively; η_g^{ch} and η_g^{dch} are the charging and discharging efficiency, respectively; Ω_{ST} is the set of all storage devices.

The upper and lower bounds of the stored energy and charging and discharging power are formulated as

$$\underline{SoC}_{g,t} \le \text{SoC}_{g,t} \le \overline{SoC}_{g,t} \quad \forall t \tag{20a}$$

$$0 \le p_{g,t}^{\text{ch}} \le \overline{p}_{g,t}^{\text{ch}} \quad \forall t \tag{20b}$$

$$0 \le p_{g,t}^{\text{dch}} \le \overline{p}_{g,t}^{\text{dch}} \quad \forall t. \tag{20c}$$

2) CERG: CERG consumes electricity to cool down the machine room. In this work, the output cooling power is assumed to be proportional to the input electricity power

$$p_t^{\text{CERG},C} = \eta^{\text{CERG}} p_t^{\text{CERG},E} \quad \forall t \tag{21a}$$

$$0 \le p_t^{\text{CERG},E} \le \overline{p}_t^{\text{CERG},E} \quad \forall t \tag{21b}$$

where $p_t^{\text{CERG},C}$ and $p_t^{\text{CERG},E}$ are the cooling and electricity power of CERG during the tth timestep; η^{CERG} is the conversion efficiency.

3) Renewable Energy Generation: Considering the curtailed power, the self-built output of renewable energy generation satisfies

$$0 \le p_{g,t}^{\text{REG}} \le p_{g,t}^{\max} \quad \forall g \in \Omega_{\text{REG}}, t \tag{22}$$

where $p_{g,t}^{\text{REG}}$ and $p_{g,t}^{\max}$ are the actual output power and feasible maximal output power of the gth renewable generation during the tth timestep; Ω_{REG} is the set of renewable energy generation devices.

4) Power Balance: The electricity power balance is formulated as

$$p_t^{\text{Load}} = \sum_{s=1}^{S} p_{s,t}^{\text{IT}} + p_t^{\text{CREG},E} + p_{g=E,t}^{\text{ch}} - p_{g=E,t}^{\text{dch}} - \sum_{g \in \Omega_{\text{REG}}} p_{g,t}^{\text{REG}} \quad \forall t \tag{23}$$

where p_t^{Load} is the importing power from the gird; $p_{g=E,t}^{\text{ch}}$ and $p_{g=E,t}^{\text{dch}}$ are the charging and discharging power of the electricity storage.

The cooling power balance is formulated as

$$p_t^{\text{C}} = p_t^{\text{CREG},C} - p_{g=C,t}^{\text{ch}} + P_{g=C,t}^{\text{dch}} - (\text{PUE} - 1) \sum_{s=1}^{S} p_{s,t}^{\text{IT}} \quad \forall t \tag{24}$$

where p_t^{C} is the net absorbing heat from the machine room; $p_{g=C,t}^{\text{ch}}$ and $p_{g=C,t}^{\text{dch}}$ are the charging and discharging power of the cooling storage; PUE refers to power usage effectiveness, a widely used energy efficiency index for IDCs, which is calculated as

$$\text{PUE} = \frac{E_{\text{IDC}}}{E_{\text{IT}}} \tag{25}$$

where E_{IDC} and E_{IT} are the total electricity consumption of IDC and the electricity consumption of servers. In this work, all electricity consumption except for the consumption of servers is assumed to cool down the machine room.

5) Thermal Dynamics of Indoor Temperature: The transition of indoor temperature in the machine room is determined by the outside temperature, the equivalent thermal resistance and capacity of the machine room, and the net absorbing heat from the machine room. In this work, a 1-order resistance-capacity thermal dynamics model is considered [31], which is formulated as

$$r_{t+1}^I = r_t^I e^{-\frac{\Delta T}{\beta}} + r_t^O \left(1 - e^{-\frac{\Delta T}{\beta}}\right) - R^H \left(1 - e^{-\frac{\Delta T}{\beta}}\right) p_t^{\text{C}} \quad \forall t \tag{26}$$

where r_t^I and r_t^O are the indoor and outside temperature at the tth timestep; β and R^H are the equivalent thermal time constant and thermal resistance of the machine room.

The indoor temperature of the server room should be within acceptable ranges depending on operation requirements of servers

$$\underline{r}_t^I \le r_t^I \le \overline{r}_t^I \quad \forall t \tag{27}$$

where $\underline{r}_t^I$ and $\overline{r}_t^I$ are the lower bound and upper bound of the indoor temperature, respectively.

6) Scheduling Cost: The scheduling cost of IDC considers both the payment for importing electricity from the grid and the difference between the initial and final state. The state decision variables of IDC include the SoC of storage, the indoor temperature, and the CU of accumulated batch workloads. Therefore, the scheduling cost is formulated as

$$c = \sum_{t=1}^{T} C_t^e p_t^{\text{Load}} + \sum_{g \in \Omega_{\text{ST}}} C_g^{\text{ST}} \Delta \text{SoC}_g + C^I \Delta r^I \tag{28}$$

where C_t^e is the electricity price during the tth timestep; C_g^{ST} is the equivalent unit cost of recovering stored energy in the gth storage device; C^I is the equivalent unit cost of recovering indoor temperature; ΔSoC_g and Δr^I are the difference of stored energy and indoor temperature between the initial and final time, respectively, which is defined as

$$\Delta \text{SoC}_g = \text{SoC}_{g,t=0} - \text{SoC}_{g,t=T} \tag{29a}$$

$$\Delta r^I = r_{t=0} - r_{t=T}. \tag{29b}$$

To ensure the operation of the IDC microgrid is sustainable for periods after the Tth timestep, the domains of ΔSoC_g and Δr^I are nonnegative reals in the proposed IDC load model.

In summary, the feasible region of the IDC load model is formulated as a polyhedron characterized by linear constraints

$$\Phi = \left\{(x, y) \in \mathbb{R}^{T+1} \times \mathbb{R}^n | Ax + By \le b\right\} \tag{30}$$

where the feasible region Φ is a polyhedron in space $\mathbb{R}^{T+1} \times \mathbb{R}^n$; $x = (\mathbf{p}^{\mathbf{Load}}, c)$ is the decision variables for the IDC electricity load and its scheduling cost; y denote other decision variables.

V. Calculation of the Compatible IDC Load Model

The compatible IDC load model is the projection of polyhedron Φ on the subspace $\mathbb{R}^{T+1}$, which is formulated as

$$\mathcal{X} = \left\{x \in \mathbb{R}^{T+1} | \exists y \in \mathbb{R}^n, \text{ s.t. } (x, y) \in \Phi\right\}. \tag{31}$$

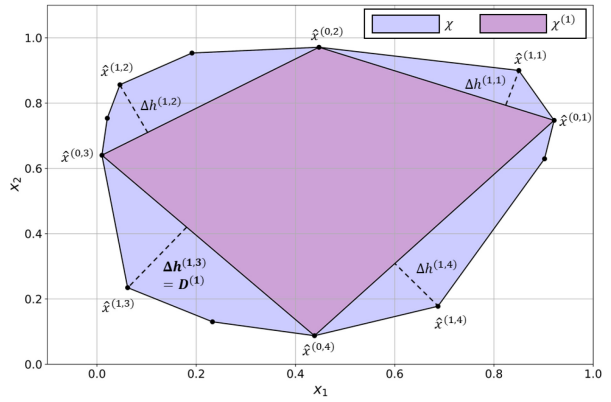

Fig. 5. Illustration of the vertex identification process in the first outer loop of the PVE algorithm.

After projection, other internal scheduling decision variables, including the scheduling of batch tasks or operation of storage, are eliminated. However, the calculation of the exact projection is difficult [19]. To overcome the difficulty, the PVE algorithm [23] and the proposed COPVE algorithm are introduced in this section.

A. Progressive Vertex Enumeration Algorithm

The PVE algorithm is an efficient algorithm for the polyhedron projection problem. The algorithm continuously expands an inner approximation convex hull of the projection polyhedron by efficiently enumerating vertices from the projection polyhedron. In each iteration, the algorithm constructs the convex hull from identified vertices, chooses the outer normal directions of facets, and tries to find vertices that can expand the current inner approximation most.

The procedures of the algorithm are shown in Algorithm 1 and illustrated in Fig. 5. The details of the algorithm are introduced as follows.

1) Vertex Identification: Because the projection region $\mathcal{X}$ is a polyhedron, each vertex of the polyhedron is an extreme point. When the exploration direction α is appointed, the vertex most aligned with the direction can be identified by solving

$$\begin{aligned} &\max_{x,y} h = \alpha^T x \\ &\text{s.t. } Ax + By \le b. \end{aligned} \tag{32}$$

2) Initialization: In the T+1-D space, at least T+2 vertices are required to avoid the degeneration of the initial convex hull $\mathrm{CH}(V^{(0)})$. In the PVE algorithm, initial vertexes are generated by searching along each axis. By solving $\alpha = \pm e_i$, where e_i is the basis vector whose ith element is 1 and is the only nonzero element, $2T+2$ vertexes are generated and formed the set of initial vertices $V^{(0)}$.

3) Inner Loop: Before the inner loop, the algorithm constructs a convex hull $\mathcal{X}^{(k)}$ of identified vertices, which is an inner approximation of the projection polyhedron. Each facet of $\mathcal{X}^{(k)}$ corresponds to a half-space denoted as $\widetilde{A}_j^{(k)} x \le \widetilde{b}_j^{(k)}$. To find vertices that can expand the current inner approximation most, new vertices are searched along the outer normal directions of facets. Because $\widetilde{A}_j^{(k)}$ is the outer normal direction of half-space $\widetilde{A}_j^{(k)} x \le \widetilde{b}_j^{(k)}$, a new vertex is identified by solving (32) with $\alpha^T = \widetilde{A}_j^{(k)}$. Note the new vertex as $x^{(k,j)}$, the contribution of including $x^{(k,j)}$ to expand the inner approximation is calculated as

$$\Delta h^{(k,j)} = \frac{\widetilde{A}_j^{(k)} x^{(k,j)} - \widetilde{b}_j^{(k)}}{\left\| \widetilde{A}_j^{(k)} \right\|} \tag{33}$$

where $\Delta h^{(k,j)}$ is the improvement ratio (IR) and calculates the distance from $x^{(k,j)}$ to the facet. If $\Delta h^{(k,j)} > 0$ and $x^{(k,j)}$ is not in the set of identified vertices, then $x^{(k,j)}$ will be added to the vertex set.

4) Outer Loop: The outer loop calculates the error of the current inner approximation $\mathcal{X}^{(k)}$ and checks if it is acceptable with the given tolerance ϵ. The Hausdorff distance between the projection $\mathcal{X}$ and the inner approximation $\mathcal{X}^{(k)}$ is used as the error metric here [32], which is defined as

$$D^{(k)} = \max_{x_1 \in \mathcal{X}} \min_{x_2 \in \mathcal{X}^{(k)}} ||x_2 - x_1||_2. \tag{34}$$

According to [24], the Hausdorff distance between $\mathcal{X}$ and $\mathcal{X}^{(k)}$ is proved to be the maximum of $H^{(k)}$

$$D^{(k)} = \max_j \Delta h^{(k,j)} = \max H^{(k)}. \tag{35}$$

Algorithm 1: PVE Algorithm

Input: Original Polyhedron Φ to be Projected, Tolerance ϵ
Output: Projection Polyhedron $\mathcal{X}$

Initialization: $k = 0$, $V = V^{(0)}$
repeat
  $k = k + 1$
  Construct a convex hull of identified vertices:
  $\mathcal{X}^{(k)} = \mathrm{CH}(V) = \{x | \widetilde{A}_j^{(k)} x \le \widetilde{b}_j^{(k)}, j \in [J^{(k)}]\}$
  Initialize set of IRs: $H^{(k)} = \emptyset$
  for $j \in [J^{(k)}]$ **do**
    Identify the vertex that expands most:
    Solve (32) with $\alpha^T = \widetilde{A}_j^{(k)}$, obtain vertex $x^{(k,j)}$ and IR of the vertex $\Delta h^{(k,j)}$
    if $\Delta h^{(k,j)} > 0$ *and* $x^{(k,j)} \notin V$ **then**
      Update identified vertices: $V := \{V, x^{(k,j)}\}$
      Save IR: $H^{(k)} := \{H^{(k)}, \Delta h^{(k,j)}\}$
  Calculate error metric $D^{(k)} = \max H^{(k)}$
until $D^{(k)} \le \epsilon$;
Return $\mathcal{X} = \mathrm{CH}(V)$;

B. Cost-Oriented Progressive Vertex Enumeration Algorithm

The PVE algorithm has been proven efficient for the projection problem of low-dimension polyhedron. However, in scenarios with multiple time slots, the convex hull of high-dimension vertices is difficult to calculate. According to [33], the size of the output facets might be exponentially larger than the size of the input vertices. The exponential size of facets

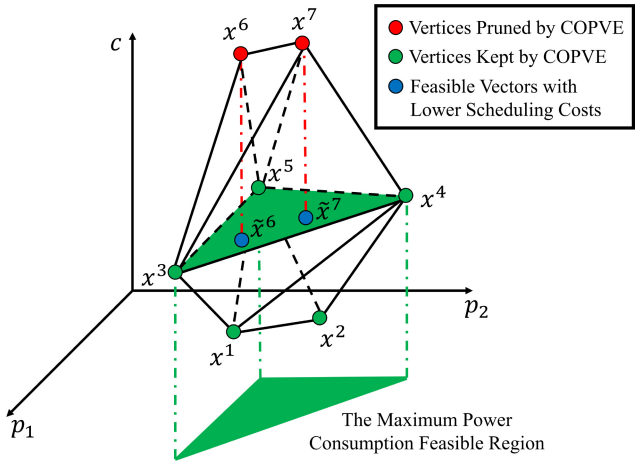

Fig. 6. Illustration of the vertices pruned by COPVE in 3-D space. COPVE prunes the vertices (x^6, x^7) whose scheduling cost is higher than the feasible vectors $(\tilde{x}^6, \tilde{x}^7)$ on the hyperplane corresponding to the maximum power consumption feasible region.

greatly increased the number of inner loops and the times of solving (32). In a complicated load model such as the IDC load model, the exponential times of identifying vertices become the main obstacle to the application of the PVE algorithm.

To overcome the obstacle, an improved PVE algorithm is proposed, called the COPVE algorithm. Compared with PVE, COPVE can realize a more efficient representation of the inner approximation, which means that COPVE achieves higher accuracy of the derived projection with the same number of vertices. Two main improvements made in COPVE are stated as follows.

1) In each outer loop, only the vertex with the highest IR will be added to V for an efficient inner approximation for higher efficiency. Because higher IR means a larger extended area when adding the vertex, COPVE tries to describe the inner approximate projection as largely as possible with limited vertices instead of including all detected vertices as PVE does.
2) In each inner loop, COPVE does not search for a vertex with a positive weight on the scheduling cost. In this way, COPVE prunes the vertices whose scheduling cost is higher than the feasible vectors on the hyperplane corresponding to the maximum power consumption feasible region as shown in Fig. 6. Because we always try to decrease the scheduling cost of IDC given a power consumption working point, these pruned vertices will never be achieved in real power system optimization and are unnecessary.

The procedures of the algorithm are shown in Algorithm 2 and are detailed as follows.

1) Initialization: In the PVE algorithm, initial vertexes are generated by searching along each axis. In the COPVE algorithm, the searching along the e_{T+1} direction is avoided because the scheduling cost is maximized in this scenario. $2T + 1$ vertexes are generated and formed the set of initial vertices $V^{(0)}$.

2) Inner Loop: In each iteration of the inner loop, the COPVE algorithm first changes the positive $T + 1$th element of $\widetilde{A}_j^{(k)}$ to a small negative number $-\upsilon$. The change avoids the search for a possible vertex with an identical power consumption working point but a higher scheduling cost. To avoid degeneration when solving (32), a negative number $-\upsilon$ is appointed here. To avoid resulting in a more conservative approximation of $\mathcal{X}$, υ should be a small number.

Next, the inner loop finds the vertex either by searching the candidate set or by solving (32) to avoid the repeated search for the same direction. Instead of appending the vertex to V, the COPVE algorithm recorded all IRs of identified vertices.

3) Outer Loop: The outer loop finds the vertex $\hat{x}^{(k)}$ with the highest IR $\Delta\hat{h}^{(k)}$ and only append $\hat{x}^{(k)}$ to V. If $\Delta\hat{h}^{(k)}$ is within the tolerance ϵ, the outer loop terminates and returns the convex hull of the identified vertices.

Algorithm 2: COPVE Algorithm

Input: Original Polyhedron Φ to be Projected, Tolerance ϵ, Positive Small Value υ
Output: Projection Polyhedron $\mathcal{X}$

Initialization: $k = 0$, $V = V^{(0)}$
Initialize the candidate set: $\Xi = \emptyset$
repeat
  $k = k + 1$
  Construct a convex hull of identified vertices:
  $\mathcal{X}^{(k)} = \mathrm{CH}(V) = \{x | \widetilde{A}_j^{(k)} x \le \widetilde{b}_j^{(k)}, j \in [J^{(k)}]\}$
  Initialize set of IRs: $H^{(k)} = \emptyset$
  for $j \in [J^{(k)}]$ **do**
    if $\widetilde{A}_{j,T+1}^{(k)} > 0$ **then**
      $\widetilde{A}_{j,T+1}^{(k)} := -\upsilon$
    if $\exists x$ *s.t.* $(\widetilde{A}_j^{(k)}, x) \in \Xi$ **then**
      Record the vertex as $x^{(k,j)}$
    else
      Solve (32) with $\alpha^T = \widetilde{A}_j^{(k)}$ and obtain vertex $x^{(k,j)}$
      Update candidate set: $\Xi := \{\Xi, (\widetilde{A}_j^{(k)}, x^{(k,j)})\}$
    Calculate IR $\Delta h^{(k,j)}$ of the vertex $x^{(k,j)}$
    Save IR: $H^{(k)} := \{H^{(k)}, (\Delta h^{(k,j)}, x^{(k,j)})\}$
  Find the vertex $\hat{x}^{(k)}$ with the highest IR $\Delta\hat{h}^{(k)}$
  Update identified vertices: $V := \{V, \hat{x}^{(k,j)}\}$
until $\Delta\hat{h}^{(k)} \le \epsilon$;
Return $\mathcal{X} = \mathrm{CH}(V)$;

VI. Results and Analysis

A. Experimental Setup

AT2018 [34], the first data center trace containing task dependency information released in 2018 by Alibaba, is chosen for the case study. Four attributes of tasks related to the IDC load model are used in clustering, including task duration time, task instance number, average CPU usage, and maximal CPU usage. If any attribute value of the task is out of the 99th percentile, the total job and all its component tasks are treated as outliers and are removed from the data

set. Several outlier analyses were conducted in [27], showing the robustness of DeGTeC against the outlier training data. According to the analysis of AT2018 in [26], around 20% of tasks are finished within a subsecond, and their computation usages are not considered in this work.

According to AT2018, the IDC microgrid has around 4000 servers. In this work, 200 and 500 W are set as the static and peak power of 2000 servers, respectively, and 250 and 530 W are set for another 2000 servers as in [4]. The PUE indicator is set to 1.2 as in [35]. In this case, the peak power of all servers and the CERG is around 2.5 MW. The penetration is defined as the ratio of the installed renewable generator capacity and the peak power and is set to 50%. A 2-h electricity storage with a capacity of 0.5 MWh and a 5-h ice storage with a capacity of 2.5 MWh are considered. The upper bound and lower bound of indoor temperature are set to 16 °C and 26°C, respectively [18]. For ρ_j, because more than 75% of tasks are finished within 100 s compared with 900 s as the timestep, ρ_j is set to 70%, a less conservative value.

As for the distribution of batch workloads deferrable time in [17] is used here. About 68% of the batch jobs are deferrable within a period shorter than 15 min. The proportion of batch jobs with a deferable time of [15 min, 30 min), [30 min,1 h), [1 h, 2 h), and [2 h, 4 h) are 3.1%, 9%, 11%, and 8.9%, respectively. To avoid the overestimation of the power consumption flexibility, the lower bound of the deferrable time is used, and we ignore the flexibility of jobs with deferrable time shorter than 15 min.

The IDC load model is modeled by Pyomo [36] and solved by Gurobi 10.0 [37]. All experiments were conducted on Intel Xeon W-3335 CPU @ 340 GHz and NVIDIA GeForce RTX3080T.

B. Batch Job Clustering Results

The benchmarks for batch job clustering are existing methods in job clustering.

1) *Statistics Followed by K-Means (statKM) [38]:* A job is represented only by its resource usage feature including the total CU $\sum_{i\in\Omega_j} L_{j,i}^{\text{task}}$ and the number of component tasks.
2) *Padding Followed by K-Means (padKM) [39]:* padKM characterizes the CU $L_{j,i}^{\text{task}}$ of each task. By zero padding missing values with zero, padKM represents jobs as m-sized vectors, where m is the maximal number of component tasks in the data set.

Both the resource and topology metrics are considered to evaluate the clustering results' performance. The total CU of each job is chosen as the resource metric. The number of tasks (NT), the critical path length (CL), and the number of edges (NE) in DAG are chosen as the topology metrics, where the critical path is defined as the longest sequence of dependent tasks in a job. The performance of each clustering method is measured by the normalized mean absolute deviation (NMAD) of each clustering result, which is defined as

$$\text{NMAD} = \frac{1}{N}\sum_{k=1}^{K}\sum_{j:C(j)=k}\frac{||m_j - m_k||}{\overline{m}} \tag{36}$$

TABLE I
PERFORMANCE OF THE DEGTEC AND BENCHMARKS

Methods	Resource Metric	Graph Metric		
	CU	NT	CL	NE
DeGTeC	1.001	**0.276**	**0.677**	**0.487**
statKM	**0.355**	0.483	0.953	0.860
padKM	0.883	1.087	1.339	1.709

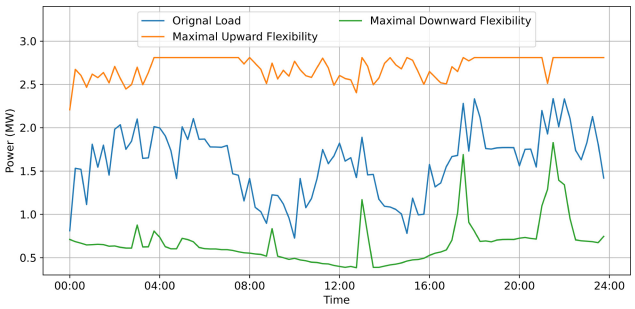

Fig. 7. Maximal upward and downward power consumption flexibility of IDC microgrid throughout the day.

where N is the total number of batch jobs; K is the number of typical batch jobs; $C(\cdot)$ denotes the mapping function to the typical job; m_k is the metric of the kth typical job; $\overline{m}$ is the mean value of the metric in the data set. For all clustering methods, the job nearest to the centroids given by K-Means is chosen as the typical batch job for this group.

Table I shows the performance of the DeGTeC and the benchmarks with 8 as the number of typical batch jobs. For the resource metric that measures the CU of batch jobs, statKM performs best among all methods. For the graph metric that measures the DAG information of batch jobs, DeGTeC outperforms all benchmarks. The result shows that the graph features are embedded into the latent vectors and make the clustering result more task dependency-aware.

C. Flexibility–Cost Assessment Results

The flexibility of IDC power consumption and its scheduling cost is analyzed using the eight typical batch jobs given by DeGTeC.

1) Power Consumption Flexibility of the IDC Microgrid: The maximal upward and downward power consumption flexibility of the IDC microgrid is shown in Fig. 7. The original load profile in the figure is derived from optimizing the power consumption without any demand response. For each timestep, the consumption flexibility intervals of the IDC microgrid are shown to be around 1.5 MW in the test case.

2) Impact of Task Dependency Relationship: To show the impact of the task dependency relationship intuitively when assessing the flexibility of IDC power consumption, the maximal upward and downward flexibility in each timestep are derived from the IDC load model. Because diurnal patterns exist in the input profiles, including the number of arriving batch workloads, output of solar and wind power, and outdoor temperature, the horizon is chosen as 24 h to show the time-varying flexibility capability. The temporal resolution is set

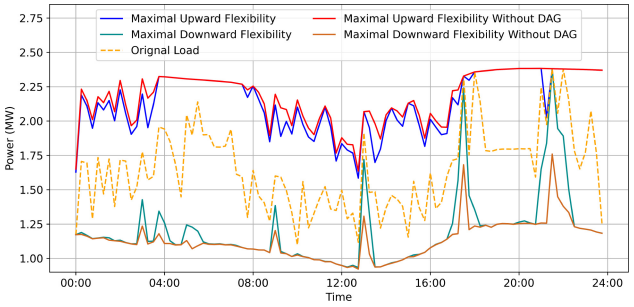

Fig. 8. Comparison of maximal upward and downward flexibility of IDC power consumption with and without task dependency. To clarify the comparison, only the flexibility caused by deferring batch workloads is considered.

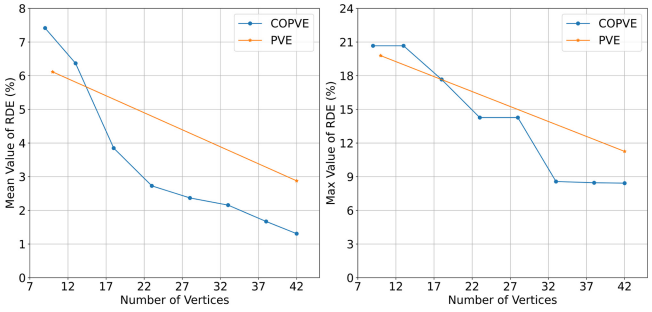

Fig. 9. Maximum and mean RDE of the compatible IDC load model with different numbers of vertices derived by the PVE and COPVE algorithms.

to 15 min consistent with other existing research works for IDC [4], [17].

Fig. 8 shows the maximal upward and downward flexibility of the IDC power consumption with or without (11), the constraints of the task dependency relationship. To clarify the difference, only the flexibility caused by deferring batch workloads is considered. It is shown that the ignorance of the task dependency overestimates the capacity of the IDC power consumption in both upward and downward flexibility. Because dependency relationships limit the number of tasks processed in parallel, more tasks need to be processed in advance to meet the deadlines, and the overestimation of downward flexibility is more remarkable than upward flexibility.

3) Performance of the COPVE Algorithm: In this section, the performance of the PVE algorithm and the proposed COPVE algorithm is compared based on the derived compatible IDC load model $\mathcal{X}$. A precise compatible load model should honestly reflect the scope of the multiperiod load profile and its corresponding minimal scheduling cost with as few as possible constraints.

To quantify the accuracy of the derived compatible load model, the power system is assumed to require an incentive demand response lasting for 1 h during the peak hour. With 15 min as the timestep, the dimension of the decision variables in the compatible load model is 5. One thousand random load profiles are generated by Latin Hypercube Sampling. Ideally, the compatible load model and the original load model should give the same expectation of the minimal scheduling cost and the feasibility of disaggregating the load profile. The expectation can be calculated by solving the following optimization problem:

$$E^C = \min_{\mathbf{p}^{\mathbf{Load}},c} \; c + M\sum_{t=1}^{T}\left|p_t^{\mathrm{Load}} - \hat{p}_t^{\mathrm{Load}}\right| \quad \text{s.t.} \; \left(\mathbf{p}^{\mathbf{Load}}, c\right) \in \mathcal{X} \tag{37}$$

$$E^O = \min_{\mathbf{p}^{\mathbf{Load}},c,y} \; c + M\sum_{t=1}^{T}\left|p_t^{\mathrm{Load}} - \hat{p}_t^{\mathrm{Load}}\right| \quad \text{s.t.} \; \left(\mathbf{p}^{\mathbf{Load}}, c, y\right) \in \Phi \tag{38}$$

TABLE II
COMPARISON OF THE ORIGINAL IDC LOAD MODEL (OM) AND THE COMPATIBLE IDC LOAD MODEL DERIVED BY THE PVE AND COPVE ALGORITHMS

	OM	42 Vertices		Same Mean RDE	
		PVE	COPVE	PVE	COPVE
Variables	824	5	5	5	5
Vertices	/	42	42	517	169
Constraints	766	551	548	10952	**3285**
Max RDE (%)	/	11.25	**8.422**	4.221	**1.601**
Mean RDE (%)	/	2.877	**1.308**	0.442	0.437
Run Time (s)	/	2.089	75.71	27.98	805.7

where E^C and E^O are the expectation for the sum of the scheduling cost and the penalty for the load profile bias given by the compatible model and the original model, respectively; M is a large positive number for the penalty of power consumption bias and is set to 10 000 here. The relative disaggregation error (RDE) of the compatible load model $\mathcal{X}$ when given by the load profile $\hat{\mathbf{p}}^{\mathbf{Load}}$ is defined as

$$\mathrm{RDE} = \frac{\left|E^C - E^O\right|}{E^O} \times 100\%. \tag{39}$$

Fig. 9 shows the maximum and mean RDE of the compatible IDC load model with different numbers of vertices derived by the PVE and COPVE algorithms. After one inner loop, the number of vertices derived by the PVE algorithm achieves 42. However, with the same number of derived vertices, the compatible IDC load model given by the proposed COPVE algorithm achieves higher accuracy.

Table II compares the original IDC load model (OM) and the compatible IDC load model derived by the PVE and COPVE algorithms in two scenarios. In the first scenario, the number of vertices in the compatible load model is fixed at 42. With the same number of constraints, COPVE outperforms PVE in both the maximal and mean values of RDE. In the second scenario, the mean value of RDE is fixed at around 0.44%. In this case, COPVE outperforms PVE in both the number of constraints in the compatible load model and the maximal value of RDE. As for the run time, it takes a longer time for COPVE to derive the compatible model because only one vertex is added to the vertices set in each outer loop of COPVE. However, even for the second scenario with more vertices, it takes less than 15 min to derive the compatible

load model with high accuracy, which is acceptable for the 1-h incentive demand response.

VII. Conclusion

This article proposes a novel framework to derive the compatible task dependency-aware IDC load model. DAG-aware job clustering is first applied to numerous typical batch jobs to get typical batch jobs. Next, the linear task dependency-aware IDC load model is formulated. Finally, the compatible IDC load model is derived via the proposed COPVE algorithm. Experiments show that the DAG information is reserved after clustering via the DAG-aware clustering framework DeGTeC. Besides, ignorance of task dependency causes the overestimation of the capacity of the IDC power consumption in both upward and downward flexibility. The comparison of the COPVE algorithm and the PVE algorithm shows that the compatible load model derived by the proposed algorithm can describe the original load model more accurately with fewer constraints and is easier to embed in the operation of power systems.

Future works on deriving the compatible task dependency-aware IDC load model include two aspects. First, the uncertainty of the incoming batch and interactive workloads should be considered. Because the scheduling of batch workloads is vital for the power consumption flexibility of IDCs, how to decrease the impact of the uncertainty on the compatible load model needs further research. Second, more operation techniques of IDCs, including the dynamic voltage and frequency scaling (DVFS), should be considered. While DVFS further improves the power consumption flexibility of servers, it imports integer variables to the original load model, and it is necessary to design the appropriate formulation of the compatible IDC load model in this case.

References

[1] M. Koot and F. Wijnhoven, "Usage impact on data center electricity needs: A system dynamic forecasting model," *Appl. Energy*, vol. 291, Jun. 2021, Art. no. 116798.

[2] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu, "A review of power consumption models of servers in data centers," *Appl. Energy*, vol. 265, May 2020, Art. no. 114806.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[4] W. Liu et al., "Online job scheduling scheme for low-carbon data center operation: An information and energy nexus perspective," *Appl. Energy*, vol. 338, May 2023, Art. no. 120918.

[5] Z. Liu et al., "Renewable and cooling aware workload management for sustainable data centers," in *Proc. 12th ACM SIGMETRICS/Perform. Joint Int. Conf. Meas. Model. Comput. Syst.*, 2012, pp. 175–186.

[6] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, and M. J. Neely, "Power cost reduction in distributed data centers: A two-time-scale approach for delay tolerant workloads," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 200–211, Jan. 2014.

[7] C. Lu, K. Ye, G. Xu, C.-Z. Xu, and T. Bai, "Imbalance in the cloud: An analysis on Alibaba cluster trace," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2017, pp. 2884–2892.

[8] J. Li and W. Qi, "Toward optimal operation of Internet data center microgrid," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 971–979, Mar. 2018.

[9] Y. Wang, X. Wang, and Y. Zhang, "Leveraging thermal storage to cut the electricity bill for datacenter cooling," in *Proc. 4th Workshop Power-Aware Comput. Syst.*, 2011, pp. 1–5.

[10] Y. Guo, Y. Gong, Y. Fang, P. P. Khargonekar, and X. Geng, "Energy and network aware workload management for sustainable data centers with thermal storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2030–2042, Aug. 2014.

[11] R. Basmadjian, J. F. Botero, G. Giuliani, X. Hesselbach, S. Klingert, and H. De Meer, "Making data centers fit for demand response: Introducing GreenSDA and GreenSLA contracts," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3453–3464, Jul. 2018.

[12] W. Li, T. Qian, Y. Zhang, Y. Shen, C. Wu, and W. Tang, "Distributionally robust chance-constrained planning for regional integrated electricity–heat systems with data centers considering wind power uncertainty," *Appl. Energy*, vol. 336, Apr. 2023, Art. no. 120787.

[13] M. Chen, C. Gao, M. Shahidehpour, and Z. Li, "Incentive-compatible demand response for spatially coupled Internet data centers in electricity markets," *IEEE Trans. Smart Grid*, vol. 12, no. 4, pp. 3056–3069, Jul. 2021.

[14] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput.*, 2012, pp. 1–13.

[15] K. Li, "Power and performance management for parallel computations in clouds and data centers," *J. Comput. Syst. Sci.*, vol. 82, no. 2, pp. 174–190, 2016.

[16] J. Lee, H. Ko, J. Kim, and S. Pack, "Data: Dependency-aware task allocation scheme in distributed edge clouds," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7782–7790, Dec. 2020.

[17] Y. Cao et al., "Data-driven flexibility assessment for Internet data center towards periodic batch workloads," *Appl. Energy*, vol. 324, Oct. 2022, Art. no. 119665.

[18] M. Chen, C. Gao, M. Shahidehpour, Z. Li, S. Chen, and D. Li, "Internet data center load modeling for demand response considering the coupling of multiple regulation methods," *IEEE Trans. Smart Grid*, vol. 12, no. 3, pp. 2060–2076, May 2021.

[19] X. Chen, E. Dall'Anese, C. Zhao, and N. Li, "Aggregate power flexibility in unbalanced distribution systems," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 258–269, Jan. 2020.

[20] W. Lin, Y. Chen, Q. Li, and C. Zhao, "An AC-feasible linear model in distribution networks with energy storage," *IEEE Trans. Power Syst.*, vol. 39, no. 1, pp. 1224–1239, Jan. 2024.

[21] H. Hao, B. M. Sanandaji, K. Poolla, and T. L. Vincent, "Aggregate flexibility of thermostatically controlled loads," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 189–198, Jan. 2015.

[22] Y. Wen, Z. Hu, and L. Liu, "Aggregate temporally coupled power flexibility of DERs considering distribution system security constraints," *IEEE Trans. Power Syst.*, vol. 38, no. 4, pp. 3884–3896, Jul. 2023.

[23] Z. Tan, H. Zhong, X. Wang, and H. Tang, "An efficient method for estimating capability curve of virtual power plant," *CSEE J. Power Energy Syst.*, vol. 8, no. 3, pp. 780–788, May 2022.

[24] Z. Tan, Z. Yan, H. Zhong, and Q. Xia, "Non-iterative solution for coordinated optimal dispatch via equivalent projection—Part II: Method and applications," *IEEE Trans. Power Syst.*, vol. 39, no. 1, pp. 899–908, Jan. 2024.

[25] M. Tirmazi et al., "Borg: The next generation," in *Proc. 15th Eur. Conf. Comput. Syst.*, 2020, pp. 1–14.

[26] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Service*, 2019, pp. 1–10.

[27] Y. Liang, K. Chen, L. Yi, X. Su, and X. Jin, "DeGTeC: A deep graph-temporal clustering framework for data-parallel job characterization in data centers," *Future Gener. Comput. Syst.*, vol. 141, pp. 81–95, Apr. 2023.

[28] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6519–6528.

[29] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2018, *arXiv:1603.07285*.

[30] A. Radovanovic, B. Chen, S. Talukdar, B. Roy, A. Duarte, and M. Shahbazi, "Power modeling for effective datacenter planning and compute management," *IEEE Trans. Smart Grid*, vol. 13, no. 2, pp. 1611–1621, Mar. 2022.

[31] M. Song, C. Gao, H. Yan, and J. Yang, "Thermal battery modeling of inverter air conditioning for demand response," *IEEE Trans. Smart Grid*, vol. 9, no. 6, pp. 5522–5534, Nov. 2018.

[32] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, Sep. 1993.

[33] D. Avis and D. Bremner, "How good are convex hull algorithms?" in *Proc. 11th Annu. Symp. Comput. Geom.*, 1995, pp. 20–28.
[34] (Alibaba Group E-Commer. Co., Hangzhou, China). *Alibaba Cluster Trace Program (Cluster-Trace-v2018)*. (2018). [Online]. Available: https://github.com/alibaba/clusterdata/blob/master/cluster-trace-v2018
[35] H. Ji et al., "Robust operation for minimizing power consumption of data centers with flexible substation integration," *Energy*, vol. 248, Jun. 2022, Art. no. 123599.
[36] M. L. Bynum et al., *Pyomo–Optimization Modeling in Python*, vol. 67, 3rd ed., Berlin, Germany: Springer, 2021.
[37] "Gurobi optimizer reference manual." 2023. [Online]. Available: https://www.gurobi.com
[38] W. Chen, K. Ye, Y. Wang, G. Xu, and C.-Z. Xu, "How does the workload look like in production cloud? Analysis and clustering of workloads on Alibaba cluster trace," in *Proc. IEEE 24th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2018, pp. 102–109.
[39] D. Gu, J. Chen, X. Shi, L. Ran, Y. Zhang, and M. Shang, "Heterogeneous-aware online cloud task scheduler based on clustering and deep reinforcement learning ensemble," in *Proc. Int. Conf. Nat. Comput., Fuzzy Syst. Knowl. Discov.*, 2020, pp. 152–159.

Jiahao Ma (Graduate Student Member, IEEE) received the B.S. degree in electrical engineering and its automation from Tsinghua University, Beijing, China, in 2023. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with The University of Hong Kong, Hong Kong.

His current research interests include long-term storage and aggregation methods of distributed energy resources.

Ruiyang Yao (Graduate Student Member, IEEE) received the M.Math. degree in mathematics and statistics from the University of Oxford, Oxford, U.K., and the M.S. degree in computing from Imperial College London, London, U.K. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with The University of Hong Kong, Hong Kong.

His current research interests include data analytics and data security in smart grids.

Bochao Zhang is currently pursuing the B.S. degree in electrical engineering and its automation with Tsinghua University, Beijing, China.

His current interests include energy forecasting and Internet data center demand response.

Zhaoyang Wang is currently the General Manager of Global Data Center, Alibaba Cloud, Hangzhou, China. He leads global data center planning, delivery, research and development, and operations.

Yuejun Yan received the Ph.D. degree in mechanical engineering from the University of Pennsylvania, Philadelphia, PA, USA.

She is currently the Technical Lead of the Global Data Center Energy and Carbon Innovation, Alibaba Cloud, Hangzhou, China. Her research interests include green AI and green cloud computing.