Contents list available at CBIORE journal website

International Journal of Renewable Energy Development

Journal homepage: https://ijred.cbiore.id

Research Article

Flexible scheduling strategy and optimization algorithm for container resources in power environment

Jiao Zhu, Shixu He*, Jie Dou, You Chen

Hainan Power Grid Co., Ltd., Haikou, 570100, China

Abstract. The power business load has significant periodicity and suddenness characteristics, and has high requirements for operational reliability. Traditional container resource static allocation methods and general elastic scheduling strategies are difficult to achieve efficient and energy-saving resource utilization while ensuring service quality. To address the above issues, research and analyze the load patterns and various constraints of typical power business scenarios, construct a mixed integer programming model with the goal of minimizing service response time, resource fragmentation rate, and system energy consumption, and propose a hierarchical elastic scheduling framework. This framework integrates a threshold based passive scaling mechanism and a pre scheduling mechanism based on bidirectional long short-term memory network load prediction. For high proportion new energy power scenarios, the wind and photovoltaic output cycle encoding is embedded into the model, and grid load constraints are introduced to achieve deep coupling between power system energy flow and container computing power flow. In a typical mixed load scenario, comparative experiments were conducted with Kubernetes' default horizontal container auto scaling and classic best fit algorithms. The results showed that the optimized scheduling scheme proposed in this paper reduced the average response time of applications by 31.2%, increased the average resource utilization rate of the cluster from 58.72% to 86.63%, and controlled the service violation rate from 8.55% to below 1.00%; Through the integration of intelligent nodes, the overall energy consumption of the system has decreased by about 22.3%. This article provides effective theoretical methods and engineering practice references for the dynamic management and optimization of cloud native infrastructure resources in the power industry.

Keywords: Container scheduling; Elastic scaling; Resource optimization; Multi-objective particle swarm optimization; Kubernetes

Received: 17th April 2026; Revised:25th July 2026; Accepted: 15th August 2026; Available online: 9st Sept 2026

1. Introduction

Driven by the "dual carbon" goal, renewable energy (wind power and photovoltaic) is being connected to the grid on a large scale, and the new power system presents a high proportion of renewable energy and high power electronicization characteristics (Duan *et al.* 2025; Deng *et al.* 2025; Rezaei *et al.* 2025). The operation and regulation of the power grid (new energy consumption, source grid load storage coordination, wind and solar power prediction) have surged, and the load fluctuation is strong, with strict requirements for real-time and reliability. This type of business is mostly deployed on cloud native container platforms, and traditional scheduling strategies are difficult to adapt to the strong random fluctuations and high reliability demands brought by renewable energy grid connection, which can easily lead to resource waste, response delay, and high energy consumption (Cao, *et al.* 2025; Roi, *et al.* 2025; Vinayak, *et al.* 2025). Therefore, for power environments with a high proportion of renewable energy, exploring flexible scheduling and optimization algorithms for container resources is of great significance in supporting efficient consumption of renewable energy and improving the operational energy efficiency of new power systems (Liu *et al.* 2024; Kumar *et al.* 2023). Currently, traditional container resource scheduling strategies mostly use static allocation or general elastic scaling mechanisms, which are difficult to adapt to the strong periodicity and sudden fluctuation characteristics of power business loads (Kafhali, 2026; Peng, *et al.* 2025; Baldouski *et al.* 2025). Moreover, the existing Kubernetes elastic scaling mechanism has scaling lag and lack of coordination in horizontal and vertical scheduling, which can easily lead to delayed service response, excessive resource fragmentation rate, system energy consumption redundancy, etc. Meanwhile, it also cannot balance the high reliability requirements of power services and the energy-saving resource utilization (Liu, Wang. 2025; Shan *et al.* 2025). In response to the above challenges, the research focuses on the intelligent elastic scheduling of container resources in the power environment. By analyzing the power business load mode and scheduling constraints, a multi-objective optimization model is constructed. The elastic scheduling strategy is designed by integrating load prediction and hybrid intelligent optimization algorithms to collaboratively optimize service response time, resource utilization, and system energy consumption, and provide theoretical and practical support for dynamic resource management of cloud native infrastructure in the power industry. The innovations of the research are as follows: For the first time, bidirectional long short-term memory networks (Bi-LSTM), discrete particle swarm optimization (DPSO), and simulated annealing (SA) are

* Corresponding author
Email: hsxexplore@163.com (S. He)

https://doi.org/10.61435/ijred.2026.62635
ISSN: 2252-4940/© 2026.The Author(s). Published by CBIORE

integrated for scheduling, tailored for high proportion renewable energy power scenarios: Periodic coding of wind and solar output is embedded. Then, grid load constraints are introduced. Energy consumption-consumption-delay collaborative goals are constructed. A scheduling framework is formed to deeply couple the energy flow in the power system and the container computing power flow, rather than simply reusing it in a general cloud environment. The core contributions of this paper focus on sustainable energy systems: a container dispatching framework adapted to high-proportion renewable energy power generation scenarios is constructed to coordinately optimize energy consumption, delay, and resource utilization. Carbon emission reduction potential is quantified to provide feasible solutions for low-carbon operation of smart grids. Model definition, algorithm comparison, statistical verification and scalability testing are improved. Method transparency and scientific rigor are significantly improved, and the relevance and practical application value in the fields of renewable energy and sustainable infrastructure are enhanced.

2. Literature review

In recent years, green computing, low-carbon data centers, and renewable energy-driven cloud systems have become research hotspots. It focuses on the decarbonization and energy efficiency optimization of computing power resources, exploring solutions for renewable energy sources such as photovoltaics and wind energy to power data centers, and reducing carbon emissions through energy-saving scheduling and load migration. Energy saving edge computing has also developed rapidly, aiming to deal with renewable energy related businesses nearby and reduce transmission energy consumption. However, existing research lacks in-depth integration of low-carbon scheduling with renewable energy consumption and smart grid energy efficiency management in power systems, and lacks customized solutions for power scenarios. Li Z *et al*. (2025) proposed a layer-aware online joint request scheduling, container placement and resource provision strategy to address the problem that container orchestration in edge computing only focuses on a single link and ignores joint optimization. This strategy used a regularization-based method to separate time-dependent placement and wake-up costs, and combined stepwise rounding to generate feasible solutions to reduce costs and adapt to dynamic needs. Compared with the baseline algorithm, the performance was improved by about 20%. Sarma (2023) proposed the Dragon Levy updating squirrel algorithm for container-aware application scheduling problems in cloud environments. This algorithm optimized resource allocation and reduced scheduling costs by defining objective functions such as total network distance, system failure, balanced cluster usage, and threshold distance. The results showed that the average cost was lower than that of the existing algorithm, and the optimal cost value reached 761.95 (Sarma 2023). Wu H *et al*. proposed a container scheduling strategy based on image layer reuse and sequence arrangement to solve the low image reuse efficiency caused by the limited storage capacity of edge servers in mobile edge computing scenarios. This strategy modeled the execution sequence as an optimal Hamiltonian path problem by greedily deploying containers and designed a decomposition algorithm to solve it. An efficient image layer update strategy was used to achieve layer reuse. The results showed that this strategy shortened the computing task completion time by 91.3% (Wu *et al*. 2025).

Container load prediction is a key technology in cloud native environments. It aims to predict future loads through historical resource usage data to achieve elastic scaling, resource optimization and cost reduction. Ye X *et al*. (2024) used empirical mode decomposition to process non-stationary load data. The long short-term memory network was used for prediction. The computing resources were automatically allocated according to the load fluctuation. The prediction accuracy was better than that of existing models, and it effectively improved the efficiency of computing resource utilization in real-time preprocessing of astronomical data (Ye *et al*. 2024). Park S and Bahn H proposed a cloud resource prediction engine that combined genetic algorithms and Bayesian neural networks to solve the inefficient resource allocation or resource bottlenecks caused by workload variability in container microservice architecture. This method learned interval distribution based on historical data and comprehensively considered the resource usage characteristics of the central processor and memory for prediction. Compared with the traditional auto-regressive integral moving average model and exponential smoothing method, the prediction accuracy of resource utilization was significantly improved and the risk of insufficient resources was effectively reduced (Park & Bahn 2024). Mo R *et al*. proposed a cross-workload power prediction method based on multi-source migration Gaussian process regression to solve the model failure caused by domain transfer between different workloads in cloud data center server power prediction. This method handled non-Gaussian characteristics by migrating rich power data knowledge from multiple source workloads to the target workload and adapting a continuous normalization flow to adjust the Gaussian process posterior distribution. The results showed that this method outperformed four traditional machine learning methods and three transfer learning methods in cross-workload power consumption prediction, effectively improving prediction accuracy and reducing data collection costs (Mo *et al*. 2025).

To sum up, existing methods are mostly designed for specific scenarios and lack adaptation to the periodicity and suddenness of power business loads. Some of them only focus on a single dispatching link or single-dimensional prediction, and do not collaboratively optimize dispatching and load prediction. There are few comprehensive solutions that take into account multi-objective dispatching and engineering practicality. Therefore, focusing on the intelligent elastic scheduling of container resources in the power environment, a multi-objective optimization model is constructed by analyzing the power business load mode and scheduling constraints. A flexible scheduling strategy is designed by integrating load prediction and hybrid intelligent optimization algorithms to collaboratively optimize service response time, resource utilization, and system energy consumption.

3. Elastic scheduling design of power containers based on improved hybrid algorithm

3.1 Analysis and modeling of container resource scheduling characteristics in power environment

Container technology has become a mainstream choice for power business deployment due to its lightweight, fast startup and resource isolation characteristics. Power business systems are accelerating their transformation to cloud-native architecture, using microservices and containerized deployment methods to improve the flexibility and maintainability. Kubernetes is Google's open source container orchestration platform. Its core function is to automatically manage the entire life cycle of large-scale containerized applications. Its elastic scaling mechanism is divided into Horizontal Pod Autoscaler (HPA) and Vertical Pod Autoscaler (VPA) (Subramanyam, Padhi 2023; Hyder *et al*. 2024). The architecture of HPA is shown in Figure 1.

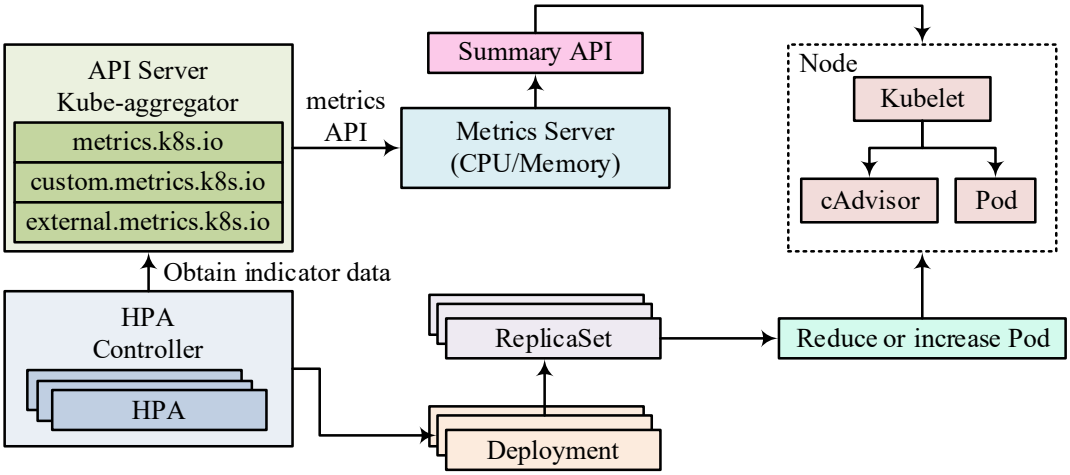


Fig. 1 Architecture of HPA

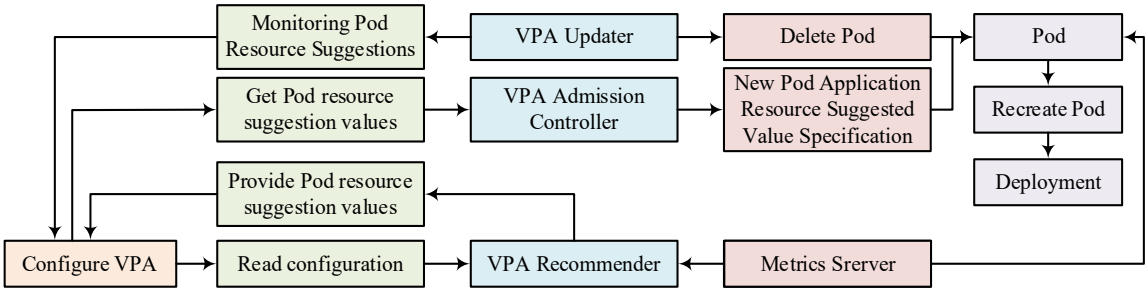


Fig. 2 Architecture of VPA

From Figure 1, the HPA controller regularly calls the Application Programming Interface (API) service through polling to obtain business load indicator data. HPA supports different metrics: basic metrics, custom metrics, and external metrics. Custom indicators and external indicators are mainly collected through Prometheus. The structure of VPA is shown in Figure 2. From Figure 2, the Recommender in VPA continuously monitors and analyzes the historical and real-time resource usage of Pods, and generates optimal resource allocation recommendations accordingly. The Updater is responsible for executing the adjustment strategy. When the difference between the recommended value and the current configuration exceeds the set threshold, the target Pod will be deleted. Then, the Deployment or ReplicaSet controller will automatically create a new Pod with new resource specifications to maintain the desired number of copies and complete vertical expansion and contraction (Thallapally 2024). However, both HPA and VPA have problems with lag and lack of coordination mechanisms, and cannot meet the requirements of power business container scheduling. In the power business container cloud cluster, the demand for business container resources changes dynamically, and elastic scaling is required to meet the demand and improve resource utilization (Li, *et al.* 2024; Shakya, *et al.* 2024; Sharma *et al.* 2025). Therefore, a mathematical model based on Kubernetes is built. From the perspective of energy systems, this model is essentially an energy efficiency optimization tool for high proportion renewable energy grids: by minimizing response delay to ensure real-time control of wind and solar power output, reducing resource fragmentation and redundant energy consumption, optimizing cluster energy consumption, and directly reducing carbon emissions in the power system. The model deeply couples container scheduling with energy flow and load fluctuations in the power grid, taking into account the efficiency of renewable energy consumption and the efficiency of computing resources, achieving synergy between smart grid energy management and cloud resource optimization. This model takes service response time, resource fragmentation rate, and system energy consumption minimization as its core objectives. It uses a weighted summation method to convert multi-objectives into a single-objective mixed integer programming problem to adapt to the multi-constraint scheduling needs of the power business. The objective function is shown in equation (1).

$$min\, F = \omega_1 \cdot F_1 + \omega_2 \cdot F_2 + \omega_3 \cdot F_3 \quad (1)$$

In equation (1), F represents the objective function. ω_1, ω_2, and ω_3 are all weights. F_1 represents the average service response time. F_2 represents the cluster's comprehensive resource fragmentation rate. F_3 represents the total energy consumption of the cluster per unit time. The weights ω_1-ω_3 are set based on the priority of renewable energy power systems: with low-carbon and energy-saving as the core. ω_3 is 0.5. Considering service reliability, ω_1 is 0.3. Considering balance resource utilization rate, ω_2 is 0.2. The weight sensitivity analysis shows that under this ratio, the balance of the three objectives is optimal. When the weight fluctuates by ±15%, the overall performance of the system decreases by less than 4%, which is suitable for stable operation requirements in power scenarios. The average service response time is shown in equation (2).

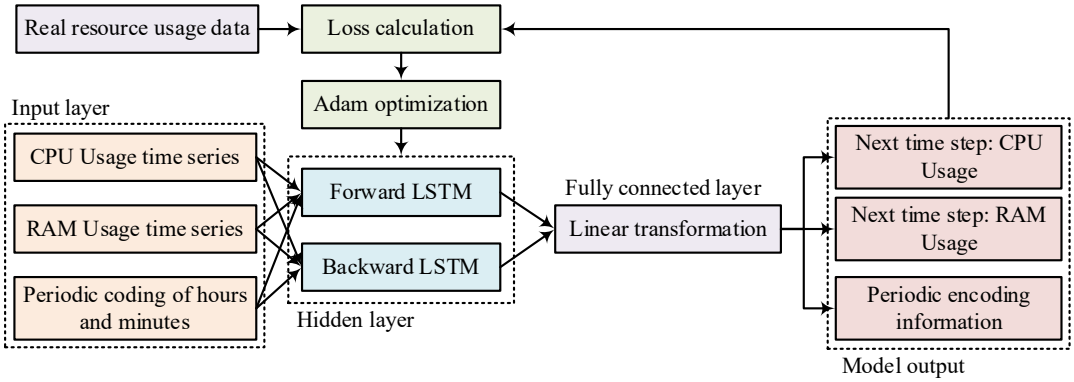


Fig 3 Load prediction model based on Bi-LSTM

$$F_1 = \frac{\sum_{i=1}^{M}\sum_{z_{ir}}^{j=1}\sum_{N}^{k=1} T_{ijk}\cdot x_{ijk}}{\sum_{i=1}^{M} z_{ir}} \quad (2)$$

In equation (2), M represents the total number of power business types. z_{ir} represents the number of Pod auto-scaling copies of resource type r for power business i. N represents the total number of container cloud cluster nodes. T_{ijk} represents the service response time from the Pod j of power service i scheduled to node k. x_{ijk} is a variable in 0-1, indicating the scheduling status of the power service Pod instance. The cluster comprehensive resource fragmentation rate is shown in equation (3).

$$F_2 = \frac{\sum_{2}^{r=1}\alpha_r \cdot \sum_{N}^{k=1} \frac{max\left(C_{kr} - U_{kr} - \sum_{M}^{i=1}\sum_{z_{ir}}^{j=1} P_j(r)\cdot x_{ijk},0\right)}{C_{kr}}\cdot y_k}{\sum_{2}^{r=1}\alpha_r} \quad (3)$$

In equation (3), α_r represents the weight coefficient of the resource type. C_{kr} represents the node's total capacity for resource types. U_{kr} represents the node's allocated resource amount for the resource type. $P_j(r)$ represents the demand value of the resource type by the Pod. y_k is a variable in 0-1, used to represent the node operating status. The total energy consumption of the cluster per unit time is shown in equation (4).

$$F_3 = \sum_{N}^{k=1} E_k \cdot y_k + \sum_{M}^{i=1}\sum_{z_{ir}}^{j=1}\sum_{N}^{k=1} e_{ijk}\cdot x_{ijk} \quad (4)$$

In equation (4), E_k represents the basic energy consumption per unit time when the node is running. e_{ijk} represents the incremental energy consumption per unit time of the power business Pod running on the node. Through the above method, a mixed integer programming model for container resource scheduling in a power environment is constructed. Constraints include: the upper limit of node CPU/memory capacity, the lower limit of Pod resource requirements, the uniqueness constraints of scheduling and the boundary constraints of renewable energy load fluctuations, ensuring the reproducibility and adaptability of the model to power scenarios.

3.2 Design of flexible scheduling strategy for power business containers

To solve the hysteresis and lack of coordination mechanism in both HPA and VPA, and meet the optimization goal, a Hybrid Pod Autoscaler for Load Forecasting (HPALF) is proposed. The HPALF framework is divided into three core parts: first, the vertical scaling resource recommendation algorithm, which dynamically calculates the optimal CPU and memory configuration of the Pod based on historical resource usage data. The second is an active horizontal scaling strategy based on the Bi-LSTM load prediction model, which uses a Bi-LSTM to predict traffic trends in advance and achieve pre-expansion to avoid response delays. The third is a horizontal and vertical hybrid elastic scaling strategy that intelligently coordinates the triggering timing and priority of horizontal and vertical scaling to ensure service stability while maximizing resource utilization efficiency. The vertical scaling resource recommendation algorithm is shown in equation (5).

$$A_{d_i}^{varr}(t) = \begin{cases} A_{pod_j}^{R}, pod_j \in P_{d_i}, ift = 0 \\ \beta \times A_{max\,CPU\left(P_{d_i}\right)}^{Used} + (1-\beta)\times A_{d_i}^{varr}(t-1), ift > 1 \end{cases} \quad (5)$$

In equation (5), $A_{d_i}^{varr}(t)$ represents the recommended value of Pod vertical scaling resource specifications at the current moment. $A_{pod_j}^{R}$ represents pod_j's request for resource R. P_{d_i} represents the Pod set that manages d_i. β represents the smoothing coefficient. $A_{max\,CPU\left(P_{d_i}\right)}^{Used}$ represents the actual resource usage of the Pod with the largest CPU resource request at the current moment. The load prediction model based on Bi-LSTM is shown in Figure 3.

From Figure 3, the load prediction model based on Bi-LSTM is divided into input, hidden, and fully connected layers. The input layer receives a 36-length time series of CPU and memory usage and sine-cosine encoded hour and minute periodicity encoding. The hidden layer consists of forward and backward LSTM layers with 128 units, learning bidirectional timing dependencies. The fully connected layer converts the 256-dimensional bidirectional output into 6-dimension to predict the CPU, memory usage, and cycle coding at the next moment (Akkepalli 2025; Mandal *et al.* 2023). The Bi-LSTM model uses 29 days of real load data from the power system, divided into training/validation/testing sets in a ratio of 7:2:1. Hyperparameters: 128 units in the hidden layer, 64 batches, and 50 training cycles. The Adam optimizer with an initial learning rate of 0.001 is used. Dropout (0.2) and an early stopping mechanism are introduced to prevent overfitting, and the time characteristics of renewable energy are combined to enhance

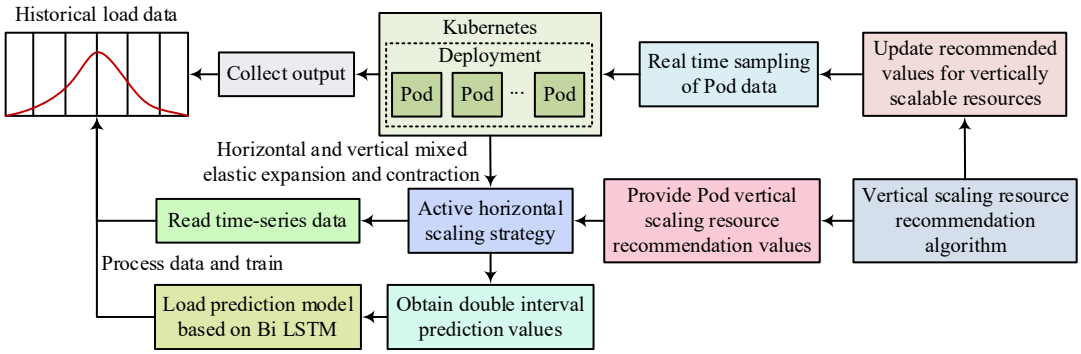


Fig 4 Structure of HPALF

generalization capabilities and ensure a balance between prediction accuracy and real-time performance. The periodic encoding of hours is shown in equation (6).

$$\begin{cases} c_h\, sin = sin\left(\frac{2\pi h}{24}\right) \\ c_h\, cos = cos\left(\frac{2\pi h}{24}\right) \end{cases} \quad (6)$$

In equation (6), $c_h sin$ and $c_h cos$ represent the sine and cosine codes of hours, respectively. h represents hour. The periodic encoding of minutes is shown in equation (7).

$$\begin{cases} c_m\, sin = sin\left(\frac{2\pi m}{60}\right) \\ c_m\, cos = cos\left(\frac{2\pi m}{60}\right) \end{cases} \quad (7)$$

In equation (7), $c_m sin$ and $c_m cos$ represent the sine and cosine codes of minutes, respectively. m represents minutes. The active horizontal scaling strategy based on Bi-LSTM uses the bidirectional structure of Bi-LSTM to capture the forward and backward dependencies of time series to achieve multi-step recursive prediction of future dual intervals (short-term and medium-term). The interval prediction is shown in equation (8).

$$\begin{cases} T_{pre} = W_{p,d} + W_{p,in} \\ s_{pre} = \left\lceil \frac{T_{pre}}{G_T} \right\rceil + 1 \end{cases} \quad (8)$$

In equation (8), T_{pre} represents the total forward prediction time. $W_{p,d}$ represents the prediction delay time window. $W_{p,in}$ is the reserved time window. s_{pre} is the number of prediction interval steps. G_T is the time granularity of the data set. However, since VPA needs to be evicted and then rebuilt when updating Pod resource specifications, it can easily lead to a breach of the service level agreement (Sanaa, *et al.* 2024; Nguyen, *et al.* 2024). VPA and HPA cannot be opened together due to differences in design mechanisms. Therefore, the above problems are solved through a horizontal and vertical hybrid elastic scaling strategy. Combining the horizontal and vertical hybrid elastic scaling strategies, the structure of HPALF is shown in Figure 4.

From Figure 4, the Deployment in Kubernetes feeds back historical load data to the load prediction model based on Bi-LSTM through the collection output. The model processes the data and reads the time series data after training, and inputs the active horizontal scaling strategy to obtain the dual-interval prediction value. Kubernetes samples Pod data in real time and updates it to the vertical scaling resource recommendation algorithm. The active horizontal scaling strategy and the vertical scaling resource recommendation algorithm together form a horizontal and vertical hybrid elastic scaling strategy. Finally, the number of Pods and resource specifications of the Deployment in Kubernetes are adjusted through horizontal and vertical hybrid elastic scaling to achieve closed-loop control.

3.3 Design of container scheduling algorithm integrating multi-objective optimization

After constructing a flexible scheduling strategy for power business containers, a container scheduling algorithm can be constructed accordingly. For the multi-objective optimization problem of container scheduling, the SA and DPSO algorithms are selected to solve it. SA is a general optimization algorithm based on the physical annealing process. It is used to find the global optimal solution in a huge search space. It avoids falling into local optima by allowing bad moves, but it has slow

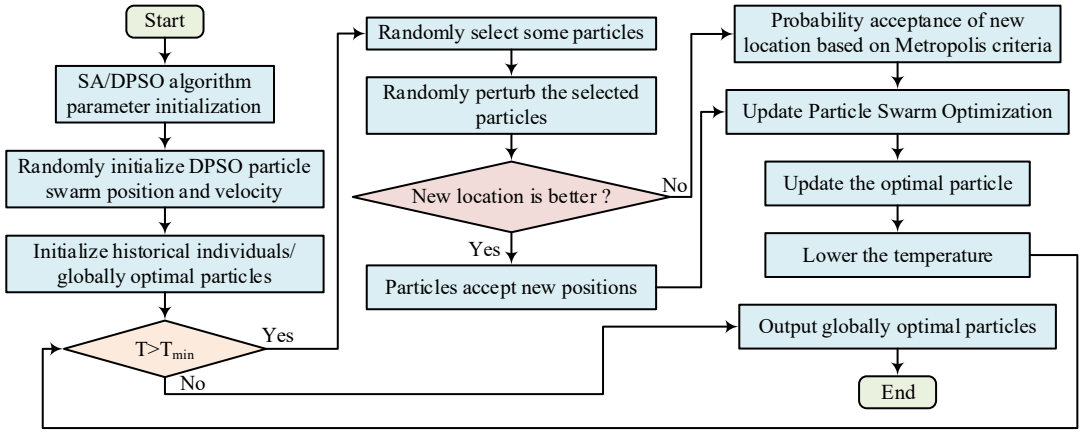
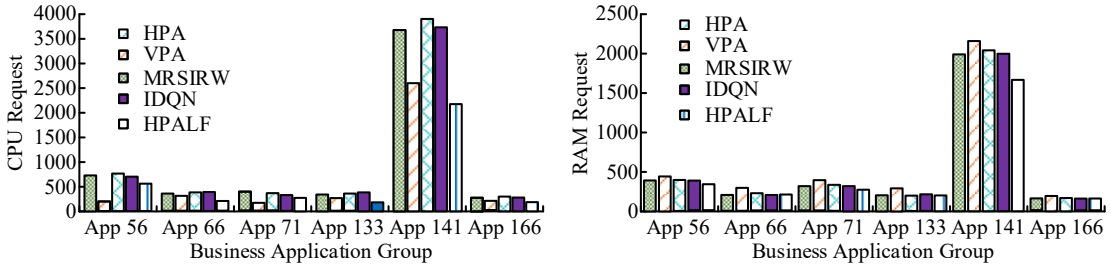


Fig 5 Process of SA-DPSO

Table 1
SLAV for different methods (%)

Business Application Group	HPA	VPA	MRSIRW	IDQN	HPALF
App 56	0.00	10.15	0.00	0.00	0.00
App 66	1.36	0.18	3.72	0.00	0.00
App 71	0.52	15.94	0.00	0.52	0.00
App 133	17.15	11.33	0.65	0.13	0.15
App 141	0.16	1.22	0.00	0.08	0.00
App 166	0.08	12.45	0.13	0.04	0.00
Average	3.21	8.55	0.75	0.13	0.03



(a) CPU Request (b) RAM Request

Note: RAM stands for Random Access Memory.

Fig 6 Mean CPU and memory requests for different methods

convergence and difficult parameter tuning (Momposhi *et al.* 2025; Wellalage, Fackrell & Zhang 2024). The core idea of DPSO is to convert velocity vectors into probabilities or operators so that particles can "fly" in discrete space. However, DPSO has parameter sensitivity and high computational cost. Therefore, the study constructs a hybrid swarm intelligence algorithm by combining SA and DPSO. Compared to classic multi-objective algorithms such as NSGA-II and MOEA/D, SA-DPSO combines the global escape capability of SA with the fast convergence advantage of DPSO, making it suitable for strong discrete constraints in power container scheduling and large fluctuations in renewable energy loads. Compared with whale optimization algorithm and reinforcement learning, it has fewer parameters, higher robustness, lower engineering deployment cost, and can balance optimization accuracy and real-time scheduling overhead, which is more in line with the comprehensive needs of low latency, high reliability, low-carbon, and energy-saving in power systems. The process of SA-DPSO is shown in Figure 5.

From Figure 5, the SA and DPSO parameters are first initialized in parallel, and the position, velocity and individual/global optimal solutions of the particle swarm are randomly generated. When the temperature is higher than the termination temperature, the loop is executed. Some particles are randomly selected for position perturbation. If the new position is better, it is accepted directly. Otherwise, the original position is accepted based on the Metropolis criterion. Next, the particle speed, position, and individual/global optimal records are updated, and the cooling operation is performed. It is looped until the temperature drops to the threshold and the global optimal solution is output (Sun *et al.* 2024; Song, Jiang & Yan 2025; Cen 2023). The fitness function of SA-DPSO is divided into four parts: energy efficiency, node resource balancing, cluster load balancing, and penalty. The energy efficiency part is shown in equation (9).

$$f_E = \frac{\Delta P_c}{\Delta P_t} \quad (9)$$

In equation (9), f_E represents the energy efficiency part of the fitness function. ΔP_c represents the power consumption increase of the cluster after deploying a set of Pods. ΔP_t represents the threshold value of the energy efficiency part. The node resource balance part is shown in equation (10).

$$f_N = \left| \frac{C_{node_i}^{Alloc}}{C_{node_i}^{Cap}} - \frac{M_{node_i}^{Alloc}}{M_{node_i}^{Cap}} \right| \quad (10)$$

In equation (10), f_N represents the node resource balance part of the fitness function. $C_{node_i}^{Alloc}$ and $M_{node_i}^{Alloc}$ represent the allocated resources of CPU and memory, respectively. $C_{node_i}^{Cap}$ and $M_{node_i}^{Cap}$ represent the total capacity of CPU and memory, respectively. The cluster load balancing part is shown in equation (11).

$$f_L = \sigma_{CPU} + \sigma_{RAM} \quad (11)$$

In equation (11), f_L represents the cluster load balancing part. σ_{CPU} and σ_{RAM} represent the sample standard deviation of CPU and memory utilization, respectively. At this time, the fitness function is shown in equation (12).

$$f(X) = \omega_E \cdot f_E(X) + \omega_N \cdot f_N(X) + \omega_L \cdot f_L(X) + \lambda \cdot PE(X) \quad (12)$$

In equation (12), $f(X)$ is the fitness function. ω_E, ω_N, ω_L, and λ are the weights of each part, respectively. $PE(X)$ is the punishment part. By combining SA-DPAO with the Kubernetes scheduling process, batch decisions can be made for all Pods. The details are as follows: In the filtering stage, the pre-selection strategy that relies on Pod indicators is abandoned, and unreasonable decisions are automatically avoided through the fitness function penalty. In the scoring stage, SA-DPSO is used to replace the default optimization strategy and allocate nodes

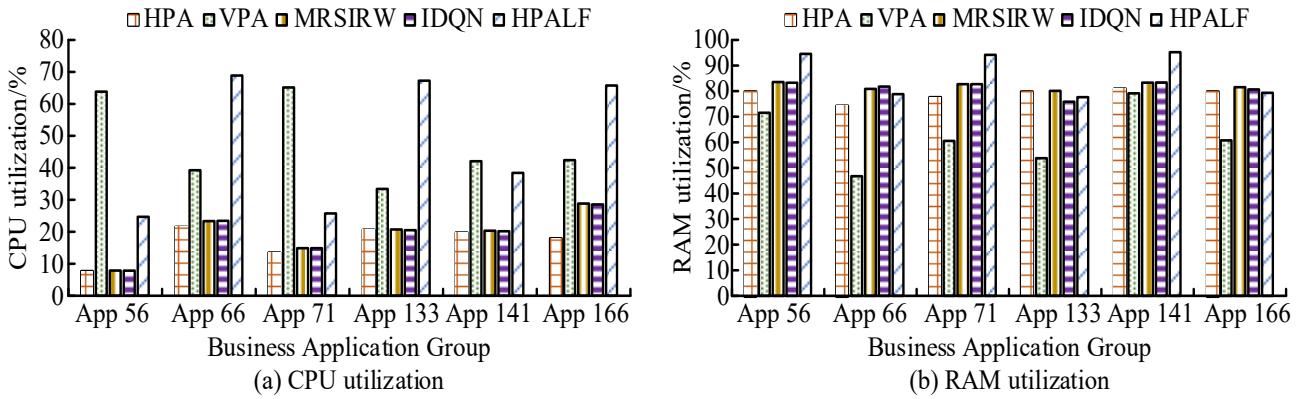


Fig 7 CPU and memory utilization of different methods

to multiple Pods at one time. Next, the above method needs to be combined with HPALF. HPALF is responsible for elastic scaling to create Pods and add them to the queue to be scheduled to optimize.

4. Simulation experiments and result analysis

4.1 Flexibility effect verification

To verify the performance of the container resource elastic scheduling method, it was compared with HPA, VPA, Multidimensional resource scheduling method based on idle rate weight (MRSIRW), and Improved Deep Q Network (IDQN). Experimental environment: 8 servers (2.5GHz 16-core CPU, 64GB memory, 512GB SSD), Kubernetes 1.24 cluster. Unified node specifications and 10 G network interconnection are used. The workload is based on Google cluster data and includes 6 types of power services related to renewable energy grid connection. The load cycles match the fluctuations in wind and solar output and cover peak/off-peak/emergency situations to ensure that the experiments approximate the actual operating conditions of the power system. The Service Level Agreement Violation (SLAV) rates of different methods are shown in Table 1. From Table 1, HPALF had a lower SLAV. The average SLAV of HPA, VPA, MRSIRW and IDQN was 3.21%, 8.55%, 0.75% and 0.13%, respectively, while the average SLAV of MRSIRW was only 0.03%. The above results show that HPALF has better service quality. The average CPU and memory requests of different methods are shown in Figure 6. From Figure 6(a), HPALF had fewer CPU requests. Taking App 141 as an example, the number of CPU requests for HPA, VPA, MRSIRW and IDQN was 3,784, 2,482, 3,558 and 3,615, respectively, while the number of CPU requests for HPALF was only 2,056. From Figure 6(b), the number of RAM requests for HPALF was also less than that of other methods. Taking App 141 as an example, the number of CPU requests for HPA, VPA, MRSIRW and IDQN was 1,967, 2,085, 1,916 and 1,924, respectively, and the number of RAM requests for HPALF was 1,593. The above results showed that the number of CPU and memory requests for HPALF was much less than that of other methods. The CPU and memory utilization of different methods are shown in Figure 7.

From Figure 7(a), compared to other methods, HPALF had higher CPU utilization. The average CPU utilization of HPA, VPA, MRSIRW and IDQN was 16.64%, 47.68%, 19.33% and 19.23%, respectively, while the average CPU utilization of HPALF was 48.46%. From Figure 7(b), the RAM utilization of HPALF was significantly higher than that of other methods. The average RAM utilization of HPA, VPA, MRSIRW and IDQN was 78.54%, 62.09%, 82.01% and 81.28%, respectively, and the average RAM utilization of HPALF was 86.63%. The above results show that HPALF has higher resource utilization.

4.2. Multi-objective optimization performance comparison

To verify the performance of the optimization algorithm, it was compared with the Shuffle Dynamics Multi-objective Grey Wolf Optimization Algorithm (SD-MOGWO) and the Multi-objective Seagull Optimization Algorithm (MOSOA). In the experiment, the inertia weight is 0.5, the individual and social learning factors are 1.0 and 1.5, respectively, the initial temperature is 1,000, and the cooling rate is 0.98. The experiment uses a dataset of renewable energy grid connected measurement in the power system, collecting containerized business data such as grid scheduling, new energy consumption, and load regulation under high proportion wind and photovoltaic grid connection for 29 days. The data covers day and night cycles, random fluctuations in wind and solar output, load peaks and valleys, and sudden scenarios, which are in line with the real load characteristics of the smart grid.

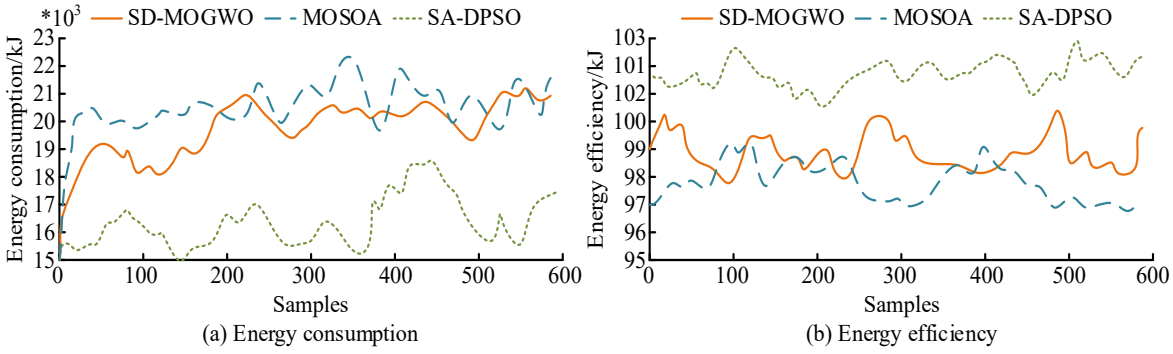


Fig 8 Energy consumption and efficiency optimization results of different methods

Table 2
Algorithm overhead and scalability

Index	Pod scale	SA-DPSO	SD-MOGWO	MOSOA
Time cost/ms	100	46.2	68.3	75.9
	500	89.4	156.7	178.2
	1,000	165.3	325.4	369.8
	2,000	312.7	689.2	786.5
	10,000	1526.8	3587.9	4128.7
Space overhead/MB	100	18.5	25.7	29.3
	500	36.8	58.9	65.4
	1,000	69.2	115.6	132.8
	2,000	132.5	248.9	286.3
	10,000	638.2	1245.8	1432.5
CPU utilization performance degradation rate/%	100	0	0	0
	500	-2.1	3.5	4.2
	1,000	1.8	8.9	10.5
	2,000	4.5	16.8	19.2
	10,000	12.3	41.2	46.8
SLAV performance degradation rate/%	100	0	0	0
	500	0.8	3.2	3.8
	1,000	1.5	7.9	9.4
	2,000	3.2	15.6	18.2
	10,000	10.8	39.7	45.2

Compared with general cluster data, it can better reflect the workload characteristics of renewable energy power infrastructure. The energy consumption and energy efficiency optimization results of different methods are shown in Figure 8.

From Figure 8(a), compared with other methods, SA-DPSO had lower energy consumption. The highest energy consumption of SD-MOGWO, MOSOA and SA-DPSO was 21,189kJ, 22,484kJ and 18,672kJ, respectively, and the average energy consumption was 19,725kJ, 20,653kJ and 16,047kJ, respectively. The unified energy consumption measurement adopts the standard model of power infrastructure, covering the power consumption of server CPU, memory, hard disk, and network equipment. Modeling is based on the basic idle energy consumption of nodes and the incremental energy consumption of Pod loads, and is normalized to the total energy consumption of the cluster per unit time. The 22.3% reduction is calculated based on this model, with the benchmark being an unoptimized scheduling plan. The results are reproducible and in line with the energy consumption evaluation standards for renewable energy power data centers. From Figure 8(b), the highest energy efficiency of SD-MOGWO, MOSOA and SA-DPSO was 100.4kJ, 99.0kJ and 102.7kJ, respectively, and the average energy efficiency was 98.9kJ, 97.8kJ and 101.8kJ, respectively. SA-DPSO has higher energy efficiency. The study used t-test to verify the significance of performance differences. The results showed that SA-DPSO had a statistically significant difference in response time and energy consumption compared to the comparative algorithm, with $p<0.05$. The 95% confidence interval analysis showed that the response time confidence interval was [0.42,0.48] s, and the energy consumption was [1578216312] kJ. The narrow and stable interval proves the prediction and dispatch performance robustness, and the results are repeatable and suitable for high reliability requirements in renewable energy generation scenarios. The above results show that the energy consumption and energy efficiency of SA-DPSO are better than those of other methods. The application response time and cluster balance of different methods are shown in Figure 9.

From Figure 9(a), compared with other methods, the application response time of SA-DPSO was shorter. The average application response time of SD-MOGWO, MOSOA and SA-DPSO was 0.60s, 0.65s and 0.45s, respectively. From Figure 9(b), the average cluster balance degrees of SD-MOGWO, MOSOA and SA-DPSO were 0.25, 0.27 and 0.14, respectively, among which the cluster balance degree of SA-DPSO was significantly lower than that of other methods. The above results show that SA-DPSO can effectively reduce application response time and cluster balance.

4.3 Algorithm overhead and scalability analysis

The algorithm overhead and scalability are shown in Table 2.

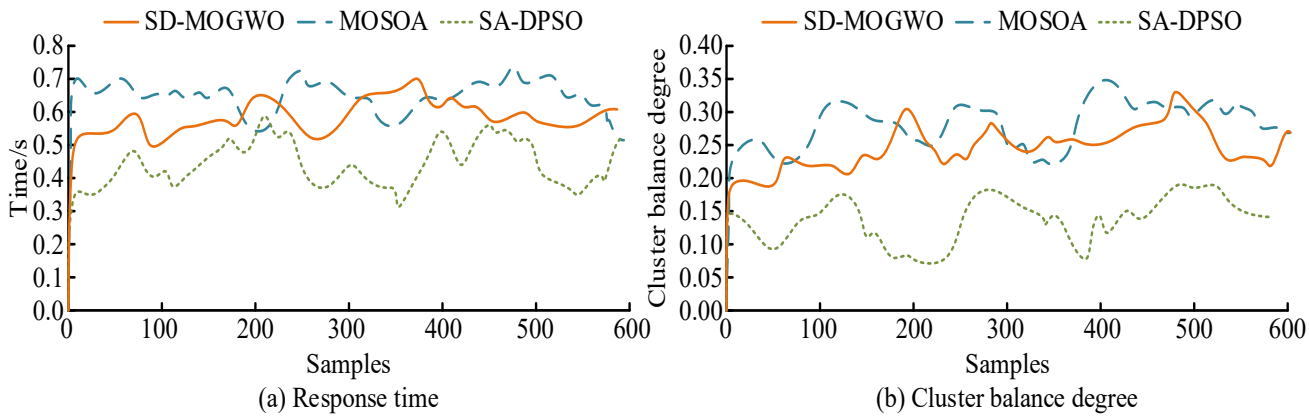


Fig 9 Application response time and cluster balance of different methods

From Table 2, the time and space overhead of SA-DPSO was much lower than that of SD-MOGWO and MOSOA. The reason is that it simplifies the optimization process through hybrid optimization of SA and discrete particle swarm, and the pre-scheduling mechanism of HPALF reduces the calculation amount of scheduling decisions. As the Pod scale increases, SA-DPSO has the lowest performance attenuation rate. In a large-scale scenario of 2,000 Pods, the CPU utilization and SLAV performance attenuation rate were only 4.5% and 3.2%, respectively. This is due to its parallel initialization design and HPALF's hierarchical elastic scheduling framework, which can effectively adapt to the needs of large-scale container scheduling in the power business.

4.4 Parameter sensitivity analysis

The parameter sensitivity analysis is shown in Table 3. From Table 3, the sensitivity of the proposed algorithm to core parameters was generally low. Only when the SA cooling rate was reduced by 20%, the performance fluctuated moderately. The remaining parameters were within the ±20% adjustment range, and the fluctuations in the application average response time and service violation rate were both controlled within 5%. This shows that the SA-DPSO algorithm has robustness, can adapt to dynamic changes in power business loads without fine parameter adjustment, and has strong engineering practicability. The cooling rate determines the speed of SA cooling: a too low rate leads to prolonged high-temperature search and slow convergence, which can amplify scheduling delays caused by fluctuations in renewable energy loads. If the speed is too high, it is prone to premature convergence and falling into local optima, which limits energy consumption optimization. The inertial weight controls DPSO exploration/development balance: If the weight is too large, the global search is strong but prone to oscillation. If it is too small, it is prone to local convergence and affects load balancing. Wind and solar output periods match prediction time steps: If the step is too long, there will be lag. If it is too short, noise interference will directly affect the pre-scheduling accuracy. The initial temperature control SA initially accepts the probability of poor solutions: searching for inefficient solutions when the temperature is too high, and premature solutions when the temperature is too low. The physical and computational effects of the above parameters jointly determine the scheduling stability and performance upper limit of the algorithm in renewable energy power scenarios.

5. Discussion

In the process of digital transformation of the new power system, services such as intelligent inspection and online monitoring carried by the power Internet of Things are characterized by strong load periodicity, significant burst characteristics, and strict service reliability requirements. Containerization and cloud native have become the mainstream trends in power business deployment (Tang, *et al*. 2024; Poojitha & Ravindranath. 2025; Neugebauer, *et al*. 2026). However, Kubernetes' native HPA and VPA scheduling strategies have problems with elastic scaling lag and lack of coordination in horizontal and vertical scheduling. Traditional static resource allocation and general elastic scheduling solutions are difficult to efficiently utilize cluster resources and optimize system energy consumption while ensuring the quality of power business services (Aruna, Pradeep. 2024; Petrosyan, 2025). This has become a key bottleneck in the dynamic management of cloud-native infrastructure resources in the power industry. Based on this research background, the study constructs a mixed integer programming model with the goal of minimizing service response time, resource fragmentation rate and system energy consumption based on the multi-constraint requirements of power business container scheduling. A HPALF hybrid elastic scaling strategy integrating Bi-LSTM load prediction is proposed. A SA-DPSO hybrid optimization algorithm is designed, aiming to achieve intelligent elastic scheduling and multi-objective optimization of container resources in a power environment.

The experimental results verify the effectiveness and superiority of the proposed method. In terms of elastic scheduling effect, the HPALF strategy controlled the average service violation rate to 0.03%, which was significantly lower than the 8.55% of VPA. The cluster memory utilization increased to 86.63%. The CPU and memory resource requests were significantly lower than those of HPA and IDQN, effectively improving service quality and resource utilization. In terms of multi-objective optimization performance, the average energy consumption of the SA-DPSO algorithm was reduced to 16,047kJ, which was 18.6% and 22.3% lower than that of SD-MOGWO and MOSOA, respectively. The average application response time was only 0.45s, and the cluster balance was 0.14, collaboratively optimizing energy consumption, response efficiency and load balancing. This scheduling framework provides technical support for operating smart grids under the high proportion of renewable energy grid integration. Accurately predicting the output fluctuations of new energy sources such as wind and solar energy, and dynamically and flexibly dispatching container resources can ensure low-latency and high-reliability operation of new energy consumption, source grid load storage collaborative control and other services. Simultaneously, reducing data center energy consumption and carbon emissions helps achieve the carbon emission reduction goals of the power system, provides efficient resource scheduling solutions for sustainable energy management of smart grids, and supports the green and low-carbon transformation of the new power system. The algorithm overhead and scalability showed that in a large-scale scenario of 2,000 Pods, SA-DPSO had far lower time and space overhead than those of the comparison algorithm, and the CPU utilization and service violation rate attenuation rate were only 4.5% and 3.2%. Parameter sensitivity analysis showed that the algorithm was overall robust to core parameters. Moderate performance fluctuations only occurred when the SA cooling rate was reduced by 20%. It can adapt to dynamic changes in power business loads without fine-tuning parameters, and has engineering practicality. From the perspective of carbon reduction potential, based on the data center energy consumption and grid carbon emission factor (0.58 tCO_2/MWh), this plan reduced energy consumption by 22.3%, corresponding to an annual emission reduction of approximately 128.6 tCO_2/10,000 cores, directly contributing to the "dual carbon" goal of the power industry. This quantitative evaluation clarifies the contribution of scheduling optimization to sustainable infrastructure and enhances its relevance to renewable energy journals. At the engineering level, this framework can be directly deployed on the power cloud platform to support renewable energy grid integration and smart grid scheduling. However, there is a trade-off between a slight increase in scheduling latency and an increase in computing power overhead under large-scale nodes, which needs to be alleviated through edge collaborative deployment. From the perspective of energy sustainability, its low-energy scheduling directly reduces data center carbon emissions, helps the low-carbon transformation of the power system, and

Table 3
Results of parameter sensitivity analysis

Parameter	Base value	Parameter adjustment	Application response time fluctuation	SLAV fluctuation	Sensitivity level
Initial temperature	1000	+20%	<2%	<3%	Low
		-20%	<4%	<5%	Low
Cooling rate	0.98	+20%	<1%	<2%	Low
		-20%	<6%	<8%	Moderate
Inertial weight	0.5	+20%	<3%	<4%	Low
		-20%	<3%	<4%	Low
Time step	36	+20%	<2%	<3%	Low
		-20%	<3%	<4%	Low

provides feasible resource optimization solutions for high proportion renewable energy power infrastructure.

In summary, the research provides an effective method for flexible scheduling of container resources in the power industry, but there are still certain limitations. The load prediction is only for the CPU and memory dimensions, and does not consider key indicators such as network bandwidth and storage IO. In the future, the multi-dimensional load sensing model can be expanded and combined with the differentiated service level requirements of the power business to further improve the accuracy and adaptability of the scheduling strategy.

6. Conclusion

In view of the periodic and sudden load characteristics and high reliability requirements of the power business, to solve traditional container scheduling strategy lag, inefficient resource utilization, etc., a mixed integer programming model was constructed with the goal of minimizing service response time, resource fragmentation rate, and system energy consumption. An HPALF hybrid elastic scaling strategy incorporating Bi-LSTM load prediction was designed. A SA-DPSO hybrid optimization algorithm was proposed to intelligently schedule power container resources. Experimental results showed that the HPALF strategy reduced the average service violation rate to 0.03%, increased the cluster memory utilization to 86.63%, and the CPU and memory resource requests were significantly lower than those of the comparison methods. The average energy consumption of the SA-DPSO algorithm reached 16047kJ, and the average application response time was only 0.45s. Compared with similar algorithms, it achieved energy consumption and response efficiency. This algorithm has low overhead, strong scalability, and strong robustness to core parameters. Only moderate performance fluctuations occur when the SA cooling rate is slightly adjusted. The above results show that the scheduling strategy and optimization algorithm effectively realize the elastic scheduling and multi-objective optimization of power container resources, providing theoretical methods and practical reference for dynamic resource management of cloud native infrastructure in the power industry.

Fundings

The research is supported by: Key Science and Technology Project, Research on Key Technologies of Power Monitoring System Situational Awareness 2.0 and Unified Prevention and Control System - Subject 6: Research on Key Technologies of Edge-End Collaborative Prevention and Control for Network Security Center Based on Unified Application Virtualization, No.: 070000KC24010008.

Reference

Aruna, K., & Pradeep, G. (2024). Container-to-fog service integration using the dis-lc algorithm. *International Journal of Computer Network and Information Security,* 16(1), 85-96. https://doi.org/10.5815/ijcnis.2024.01.07.

Akkepalli, S. (2025). A Novel Framework of Anomaly-Based Network Intrusion Detection using Hybrid CNN, Bi-LSTM Deep Learning Techniques. *Journal of Information Systems Engineering & Management*, 10(19), 247-254. https://doi.org/10.52783/jisem.v10i19s.3015

Baldouski, D., Krész, M., & Dávid, B. (2025). Scheduling truck arrivals for efficient container flow management in port logistics. *Central European journal of operations research*, 33(3), 791-818. https://doi.org/10.1007/s10100-025-00976-x.

Cen, H. (2023). Assembly Sequence and Assembly Path Planning of Robot Automation Products Based on Discrete Particle Swarm Optimization. *International Journal of Vehicle Structures and Systems*, 15(3), 400-408. https://doi.org/10.4273/ijvss.15.3.20

Cao, R., Zhang, P., Wu, Y., Liu, J., & Su, H. (2025). Adaptive container scheduling based on reinforcement learning in kubernetes. *CCF Transactions on High Performance Computing*, 7(3), 291-304. https://doi.org/10.1007/s42514-025-00223-4.

Damayanti, A., Sarto, S., & Sediawan, W. (2020). Biohydrogen Production by Reusing Immobilized Mixed Culture in Batch System. *International Journal of Renewable Energy Development*, 9(1), 37-42. https://doi.org/10.14710/ijred.9.1.37-42

Du, B., Guo, S., Guo, J., Sun, X., Wang, K., & Yang, Z. (2024). A Pareto-based hybrid genetic simulated annealing algorithm for multi-objective hybrid production line balancing problem considering disassembly and assembly. *International Journal of Production Research*, 62(13), 4809-4830. https://doi.org/10.1080/00207543.2023.2280696

Duan, J., Xiong, Z., & Li, L. (2025). Integrated sustainable scheduling in u-shaped automated container terminal considering collection–distribution operations and adaptive partitioning strategies. *Transportation Research Record*, 2679(7), 353-384. https://doi.org/10.1177/03611981251327218.

Deng, W., Zhu, L., Shen, Y., Zhou, C., Guo, J. , & Cheng, Y. (2025). A novel multi-objective optimized dag task scheduling strategy for fog computing based on container migration mechanism. *Wireless Networks*, 31(2), 1005-1019. https://doi.org/10.1007/s11276-024-03811-4.

Gill, S. S., Kumar, M., Samriya, J. K., & Dubey, K. (2023). QoS-aware resource scheduling using whale optimization algorithm for microservice applications. *Software: Practice and Experience*, 54(4), 546-565. https://doi.org/10.1002/spe.3211

Guo, J., Li, Z., Lou, J., Tang, Z., Wang, T., & Jia, W. (2025). Online Layer-Aware Joint Request Scheduling, Container Placement, and Resource Provision in Edge Computing. *IEEE Transactions on Services Computing*, 18(1), 328-341. https://doi.org/10.1109/TSC.2024.3504237

Han, Q., Shan, C., Gao, R., Liu, T., Yang, Z., & Zhang, J. (2025). KCES: A Workflow Containerization Scheduling Scheme Under Cloud-Edge Collaboration Framework. *IEEE Internet of Things Journal*, 12(2), 2026-2042. https://doi.org/10.1109/JIOT.2024.3466231.

Hansen, F., Bayo, A., Ye, X., Zhang, Y., Du, X., & Li, J. (2024). Molecular spectral line data preprocessing container load grouping prediction algorithm based on EMD-LSTM. *Journal of Applied*

Artificial Intelligence, 1(1), 68-84. https://doi.org/10.59782/aai.v1i1.279

Hyder, M. F., Ahmed, S. H., Latif, M., Aslam, K., Rab, A. U., & Siddiqui, M. T. (2024). Towards Digital Forensics Investigation of WordPress Applications Running Over Kubernetes. *IETE Journal of Research*, 70(4), 3856-3871. https://doi.org/10.1080/03772063.2023.2195837

Jiang, X., Song, B., Yan, H., & Yan, H. (2025). Optimization of Active Response Efficiency of Power Demand Side Based on Discrete Particle Swarm Optimization. *Journal of Intelligent & Fuzzy Systems*, 49(3), 822–835. https://doi.org/10.1177/18758967251353378

Khan, S., Calheiros, R. N., Maurushat, A., & Momposhi, L. (2025). Next-Gen Threat Hunting: A Comparative Study of ML Models in Android Ransomware Detection. *FinTech and Sustainable Innovation*, 1, A3-A3. https://doi.org/10.47852/bonviewfsi52025685

Li, K., Lin, W., Mo, R., Xu, M., & Zhong, H. (2025). A Cross-Workload Power Prediction Method Based on Transfer Gaussian Process Regression in Cloud Data Centers. *IEEE Transactions on Cloud Computing*, 13(3), 910-921. https://doi.org/10.1109/TCC.2025.3575790

Li, K., Lin, W., Wu, H., Zhang, H., Shi, F., & Shen, W. (2025). Container Scheduling Strategy Based on Image Layer Reuse and Sequential Arrangement in Mobile Edge Computing. *IEEE Transactions on Mobile Computing*, 24(9), 8700-8713. https://doi.org/10.1109/TMC.2025.3557160

Li, R., Liu, S., Wang, W., Zhong, S., Peng, Y., & Tian, Q. (2024). A graph-based approach for integrating massive data in container terminals with application to scheduling problem. *International Journal of Production Research*, 62(16), 5945-5965. https://doi.org/10.1080/00207543.2024.2304021

Liu, Y., Wang, K. (2025). Cloud computing based green and low-carbon building energy consumption monitoring and forecasting. *International Journal of Global Warming*, 36(4):350-364. https://doi.org/10.1504/IJGW.2025.147643.

Li, Z., Lou, J., Wu, J., Guo, J., Tang, Z., & Shen, P., *et al.* (2024). Online container scheduling with fast function startup and low memory cost in edge computing. *IEEE Transactions on Computers*, 73(12), 2747-2760. https://doi.org/10.1109/TC.2024.3441836.

Kafhali, S. E. . (2026). A survey of adaptive scheduling techniques, goals, and challenges in kubernetes. Archives of Computational Methods in Engineering, 33(4), 6047-6070. https://doi.org/10.1007/s11831-026-10497-8.

Mandal, R., Mondal, M. K., Banerjee, S., Srivastava, G., Alnumay, W., & Ghosh, U., *et al.* (2023). MECpVmS: an SLA aware energy-efficient virtual machine selection policy for green cloud computing. *Cluster computing,* 2023. https://doi.org/10.1007/s10586-022-03684-2.

Neugebauer, J., Ahmet Cürebal, Heilig, L., & Vo, S. (2026). Straddle carrier routing optimization at container terminals utilizing telemetry-based prediction. *International Transactions in Operational Research,* 34(1), 429-459. https://doi.org/10.1111/itor.70186.

Nguyen, V. S., Pham, Q. D., & Huynh, T. T. (2024). Modelling and solving a real-world truck-trailer scheduling problem in container transportation with separate moving objects. *Opsearch,* 61(2), 628-661.

Petrosyan, D. A. (2025). Scheduling ml and hpc jobs with shoc platform over kubernetes. *Physics of Particles and Nuclei,* 56(6), 1581-1585. https://doi.org/10.1134/S106377962570090X.

Peng, W., Tang, Z., Guo, J., Lou, J., Wang, T., & Jia, W. (2025). LR2Scheduler: layer-aware, resource-balanced, and request-adaptive container scheduling for edge computing. *CCF Transactions on Pervasive Computing and Interaction,* 7(4), 377-391. https://doi.org/10.1007/s42486-025-00186-z.

Park, S., & Bahn, H. (2024). Prediction Techniques of Container Workload Considering the Resource Usage Characteristics of CPU and Memory. *The Journal of the Institute of Internet, Broadcasting and Communication*, 24(6), 115-120. https://doi.org/10.7236/JIIBC.2024.24.6.115

Poojitha, S. A., & Ravindranath, K. (2025). Quality aware batch scheduling of containers in cloud computing environment. *International Journal of Information Technology*, 17(2), 1155-1163. https://doi.org/10.1007/s41870-024-02331-w.

Roi, H. V., You, S. S., Kim, H. S., Long, L. N. B., Cuong, T. N., & Nguyen, D. A., et al. (2025). Heuristic-assisted deep learning for integrated scheduling of multiple equipment in automated container terminals. *Maritime economics & logistics*, 27(4), 786-818. https://doi.org/10.1057/s41278-025-00327-2.

Rezaei, P., Hadid, M., Elomri, A., & Aydin, N. (2025). Enhancing maritime container logistics through optimized repair scheduling. *Procedia Computer Science*, 274(3), 248-257. https://doi.org/10.1016/j.procs.2025.12.025.

Sanaa, A., Imane, T., & Mohamed, M. (2025). Quay crane scheduling in container terminals using a hybrid genetic algorithm. *Journal of Computer Science,* 21(1), 197-202. https://doi.org/10.3844/jcssp.2025.197.202.

Sarma, S. K. (2023). Metaheuristic based auto-scaling for microservices in cloud environment: a new container-aware application scheduling. *International Journal of Pervasive Computing and Communications*, 19(1), 74-96. https://doi.org/10.1108/IJPCC-12-2020-0213

Shakya, S., Tripathi, P. (2024). Using light weight container a mesh based dynamic allocation task scheduling algorithm for cloud with IoT network. *International Journal of Information Technology*, 16(5), 2847-2861. https://doi.org/10.1007/s41870-024-01772-7.

Sharma, S., & Vishwakarma, R. (2025). An efficient container scheduling framework for resource allocation in a cloud computing environments. *Communications on Applied Electronics*, 7(40), 39-48. https://doi.org/10.5120/cae2025652909.

Subramanyam, R. V. B., Padhi, S. (2023). Efficient Renewable Energy-Based Geographical Load Balancing Algorithms for Green Cloud Computing. *International Journal of Web and Grid Services*, 19(4), 401-426. https://doi.org/10.1504/ijwgs.2023.10058557.

Tang, Z., Mou, F., Lou, J., Jia, W., Wu, Y., & Zhao, W. (2024). Joint resource overbooking and container scheduling in edge computing. *IEEE Transactions on Mobile Computing*, 23(12), 10903-10917. https://doi.org/10.1109/TMC.2024.3386936

Thallapally, N. (2024). Scalable Application Deployment with Docker and Kubernetes. *Journal of Computer Science and Technology Studies*, 6(1), 249-257. https://doi.org/10.32996/jcsts.2024.6.1.29

Vinayak, P. S., Mallick, S. S., Jagarlamudi, L., Chakraborty, A., & Simmhan, Y. (2025). Carl: cost-optimized online container placement on vms using adversarial reinforcement learning. *IEEE Transactions on Cloud Computing,* 13(1), 321-335. https://doi.org/10.1109/TCC.2025.3528446.

Wellalage, A., Fackrell, M., & Zhang, L. (2024). A Monte Carlo simulation-based simulated annealing algorithm for predicting the minimum staffing requirement at a blood donor centre. *Annals of Operations Research*, 342(3), 1945-1990. https://doi.org/10.1007/s10479-023-05297-3.

Wu, J., Guo, J. Tang, Z., Luo, C., Wang, T., & Jia, W. (2025). Sequence-aware online container scheduling with reinforcement learning in parked vehicle edge computing. Vehicular Technology, *IEEE Transactions*, 74(8), 12921-12934. https://doi.org/10.1109/TVT.2025.3554595.