TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 13/22 pp158–173
DOI: 10.26599/TST.2023.9010009
Volume 29, Number 1, February 2024

A Server Placement Algorithm for Reducing Risk and Improving Power Utilization in Data Centers

Rui Chen, Huikang Huang, Xiaoxuan Luo, and Weiwei Lin*

Abstract: As the power demand in data centers is increasing, the power capacity of the power supply system has become an essential resource to be optimized. Although many data centers use power oversubscription to make full use of the power capacity, there are unavoidable power supply risks associated with it. Therefore, how to improve the data center power capacity utilization while ensuring power supply security has become an important issue. To solve this problem, we first define it and propose a risk evaluation metric called Weighted Power Supply Risk (WPSRisk). Then, a method, named Hybrid Genetic Algorithm with Ant Colony System (HGAACS) , is proposed to improve power capacity utilization and reduce power supply risks by optimizing the server placement in the power supply system. HGAACS uses historical power data of each server to find a better placement solution by population iteration. HGAACS possesses not only the remarkable local search ability of Ant Colony System (ACS), but also enhances the global search capability by incorporating genetic operators from Genetic Algorithm (GA). To verify the performance of HGAACS, we experimentally compare it with five other placement algorithms. The experimental results show that HGAACS can perform better than other algorithms in both improving power utilization and reducing the risk of power supply system.

Key words: server placement; power utilization; power supply risk; swarm intelligence algorithm

1 Introduction

In recent years, the rapidly growing demand of computing capacity in various enterprises has led to a surge in data center energy consumption, and also need to deploy more servers to satisfy the computing capacity, which makes it necessary to scale up data centers. However, the supporting infrastructures are very expensive, resulting in high capital expense for data centers, almost proportional to the size of the power capacity that can be provided, ranging from \$ 10 to \$ 20 per watt[1]. Global expense spending on data center construction is predicted to reach \$ 207.44 billion in 2022[2], and this high expenditure makes it practical to improve the power capacity utilization of infrastructures of existing data centers.

Power overprovisioning[3] is a major reason for underutilization of power resources (power infrastructure and power budget). Therefore, power oversubscription is commonly used in modern data centers[3–5]. Power oversubscription means that the theoretical total maximum power of the computing devices deployed at a power supply node is higher than the power threshold that depends on the power supply infrastructure, thus mitigating power capacity waste. However, while power

• Rui Chen, Huikang Huang, and Xiaoxuan Luo are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. E-mail: cssion@mail.scut.edu.cn; huikanghuang0321@gmail.com; 714197010@qq.com.
• Weiwei Lin is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: linww@scut.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2022-12-18; revised: 2023-01-15; accepted: 2023-02-16

oversubscription increases power utilization, it poses power supply risks as well. As the level of power oversubscription increases, the power supply risk also rises and will become more vulnerable to power attacks[6], resulting in circuit breaker tripping and damage to IT devices. According to estimates in Ref. [7], the losses of data center downtime are high and average cost has increased from $ 505 502 in 2010 to $ 740 357 in 2016. Therefore, how to cope with power supply risk is essential for data centers with power oversubscription.

In addition, the problem of power budget fragmentation cannot be ignored. In data centers, servers with similar workloads are often grouped together in the same power supply node, such as a rack or a Power Distribution Unit (PDU). This means that many servers with the same power consumption pattern experience a notable surge in power demand simultaneously. To avoid the risk of overload, it is crucial to equip power supply nodes with sufficient power resources. However, during periods of low workload levels, power resources at the power supply nodes are wasted. Evidently, to improve the power utilization, besides the adoption of power oversubscription, the power budget fragmentation should be actively addressed.

A large number of studies have been proposed to cope with the power supply risks associated with power oversubscription, including power capping, Energy Storage Devices (ESDs) supplement, and workload scheduling. Power capping is a popular way to control the power requests of servers through power control techniques (e.g., DVFS, RAPL[8], and Thunderbolt[9]). However, power capping inevitably leads to a Quality of Service (QoS) degradation, especially in multi-tenant data centers, which can violate the Service Level Agreement (SLA) and lead the enterprise to pay out money to customers[10]. Using ESDs supplement is able to meet the demand of peak shaving for a short time. However, the capacity and lifetime of the battery are issues to be considered[11–15]. Workload scheduling can effectively avoid the possibility of peak overlap through an optimal scheduling strategy, as well as reduce power budget fragmentation, but it is necessary to consider whether QoS meets user requirements[16, 17].

In term of scheduling with greater granularity, it is crucial to consider that servers executing specific services often demonstrate periodic power consumption patterns. By placing servers strategically, we can effectively minimize the overlap of peak power usage and alleviate the fragmentation of the power budget. To resolve the conflict between improving power utilization and reducing power supply risk in data centers, we propose a server placement algorithm called Hybrid Genetic Algorithm with Ant Colony System (HGAACS). Based on the historical power data of each server, the swarm intelligence algorithm HGAACS, which is a mixture of Genetic Algorithm (GA) and Ant Colony System (ACS), is used to solve for the better server placement solution, which enables increased power utilization and more secure power supply without changing the power infrastructure of data center.

Specifically, the main contributions of this article are as follows:

(1) We propose a new power supply risk evaluation metric, namely Weighted Power Supply Risk (WPSRisk), which takes into account not only the probability of overload, but also the impact of overload magnitude on power supply risk.

(2) We propose HGAACS, a hybrid swarm intelligence algorithm that combines ACS and GA to optimize server placement considering risk and power utilization. HGAACS enhances the global search ability of the algorithm and generates high-quality solutions by performing selection, mutation, and crossover operations on the solutions at each iteration.

(3) The HGAACS algorithm is experimentally validated on different data center entities and compared with other placement algorithms. The results show that HGAACS is able to improve power utilization and reduce power supply risks more effectively than other algorithms.

The rest of the article is organized as follows. Section 2 presents the related work. Section 3 defines the server placement problem. Section 4 details the overall flow of HGAACS and the specific design of the algorithm. Section 5 shows the performance of HGAACS in simulation experiments and compares it with several other algorithms. We finally conclude the research in Section 6. Table 1 gives the variables and the corresponding explanations commonly used in this paper.

2 Related Work

Power oversubscription can significantly improve power utilization, but the risks associated with power overloads must be considered. Since power overloads can cause downtime depending on whether the circuit breaker trips or not. Therefore, the tripping characteristics of the circuit breaker need to be considered when

Table 1 List of variables.

Variable	Description
$server_j$	Server with index j in the server set
$rack_i$	Rack with index i in the rack set
sP_j	Power of $server_j$
rP_i	Power of $rack_i$
A_i	Asynchronous score of $rack_i$
M_i	Set of the indexes of servers that have been assigned to $rack_i$
I_r	Rated current of the circuit breaker
B	Upper limit of oversubscription level
η	Ratio of overload current to rated current of the circuit breaker
$C(\eta)$	Function of the lower limit of the tripping time for the ratio of overload current to rated current
$risk_i$	Risk metric of $rack_i$
$P_i(z)$	Probability distribution function of the power of $rack_i$
$limit_i$	Rated power of the circuit breaker at $rack_i$
R	Set of all racks
S	Set of all servers
m	Number of servers
n	Number of racks
x_{ij}	Record whether $server_i$ is assigned to $rack_j$
X	Server placement solution
r_i	Peak power ratio of $rack_i$
$serverCount_i$	Number of servers in $rack_i$
$NumLimit_i$	Maximum number of servers that can be placed in $rack_i$
$Solution$	Server placement solution obtained by HGAACS
$T_{k,i}$	Pheromone between $server_k$ and $rack_i$
$\tau_{k,j}$	Pheromone between $server_k$ and $server_j$
$p_{k,i}$	Probability that $server_k$ is assigned to $rack_i$
$\eta_{k,i}$	Heuristic value between $server_k$ and $rack_i$
$initDistribution$	Initial server placement in each rack determined during the initial phase of HGAACS
$AntSwarm$	Population obtained after constructing ant
$\Delta\tau_i$	Increment of pheromone among the servers in $rack_i$
K	Maximum number of iterations
L	Population size
T	Length of the server history power consumption data sequence
P_c	Crossover probability parameter
P_m	Mutation probability parameter in GA
Pm_1	Probability parameter of performing mutation mode 1 in HGAACS
Pm_2	Probability parameter of performing mutation mode 2 in HGAACS
ρ	Local pheromone evaporation coefficient
μ	Gobal pheromone evaporation coefficient
q_0	Parameter for exploitation
α	Parameter to determine the importance of pheromone information
β	Parameter to determine the importance of heuristic information
τ_0	Initial value of pheromone
c	Power supply risk penalty factor

assessing the power supply risk. There are few works to quantitatively assess the power supply risk due to power oversubscription. In Ref. [18], the authors gave the power supply risk evaluation metric of overload probability for data centers with power oversubscription, but the overload probability only considers the frequency of power overloads, which cannot reflect the magnitude of the overloads and the impact caused by the characteristics of circuit breaker. Many researchers noticed the effect of the tripping characteristics of circuit breakers on power oversubscription. For example, Fu et al.[19] first proposed that the utilization of the delayed tripping characteristics of circuit breakers can further improve the power utilization. Some researchers designed power capping strategies based on the tripping characteristics of circuit breakers in power management systems[20, 21].

Apart from power oversubscription, many researchers noted that reducing power budget fragmentation by optimizing workload placement can improve power utilization. Kumbhare et al.[17] proposed a prediction-based VM placement technique to protect critical workloads while increasing power oversubscription levels. Jiang et al.[22] proposed a peak-aware job scheduling method that can significantly reduce the peak power of a cluster, smooth the power curve, and also reduce energy expenses, while meeting task completion deadlines. Zhang et al.[23] proposed an offline workload placement strategy, namely Flex-Offline, which can reduce the stranded power while ensuring the security of power supply in any maintenance scenario.

For more granular workload placement, i.e., server deployment, some researchers proposed hot-aware server placement approaches oriented to reduce the outlet temperature of servers to save cooling energy[24, 25]. As for the research on power resource optimization using temporal differences in workload power consumption patterns, Hsu et al.[26] proposed SmoothOperator, which uses the k-means to cluster service instances or servers with periodic synchronous power consumption patterns , and uses heuristics for optimal placement to reduce the sum of peaks in power supply nodes. However, SmoothOperator only considers the degree of asynchrony in the power consumption patterns and does not take into account the power magnitude generated by the service instance or server placement, which may result in excessive peak power generated by individual power supply nodes, especially in data centers with heterogeneous servers. In addition, Yan et al.[27] proposed a GA-based algorithm for server placement. The optimization objective combines the reduction of local hot spots in server rooms and the reduction of power budget fragmentation, which can save the energy consumption of cooling as well as improve the power utilization.

3 Problem Definition and Modeling

Server placement in data center, which is the placement of servers into racks to create a “server-rack” mapping relationship, is a bin packing problem. In data centers, due to the presence of various types of workloads, servers running different types of workloads tend to display distinct power consumption patterns. If server placement is performed regardless of these differences, it may cause serious power budget fragmentation and may even lead to power supply risks. Therefore, optimizing server placement is of great importance for data centers with power oversubscription. In this section, the power budget fragmentation problem and the power supply risk problem are described in detail, followed by a description of the corresponding optimization objectives.

3.1 Power budget fragmentation

There are differences in power consumption patterns among servers running different types of workloads. In Ref. [26], the authors pointed out that the data center at Facebook contains a Web cluster, a database cluster, and a Hadoop cluster. There are differences in power consumption patterns among the servers of these three types of clusters. The authors also indicated that even in the same service cluster, there are some variations in the power consumption patterns of the servers due to the imbalance in access to the servers. When a large number of servers with synchronized power patterns are placed in the same power supply node, the power budget is rapidly consumed at that power supply node because the power demands of these servers reaches high levels at the same time. Even though the parent node has more power budget remaining, the child node cannot join any more workloads, which results in the power budget of the parent node being locked, this wasted power margin is called power budget fragments.

Taking into account the differences in power consumption patterns among different types of workloads, the placement of servers can be properly planned to alleviate power budget fragments and thus improve power utilization. Figure 1 illustrates the influence of different server placements on power utilization. As shown in the left part of Fig. 1, when servers with synchronous power pattern are placed into the same rack, the power curve of the rack fluctuates with higher amplitude, making the available power budget of the rack relatively small. In contrast, when servers with asynchronous power patterns are placed into the same rack as shown in the right part, the power curve of the rack becomes smoother, freeing up more power budget and thus allowing for better power utilization.

To mitigate power budget fragmentation, the degree of asynchrony in power patterns among servers needs to be quantified, so that the suitability of a server combination

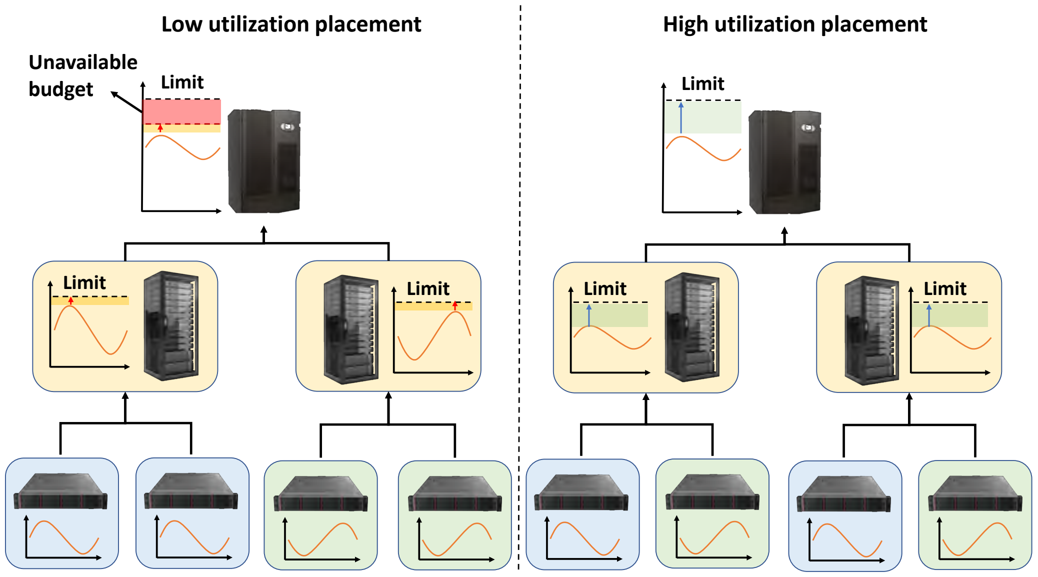


Fig. 1 Low power utilization compared with high power utilization server placement.

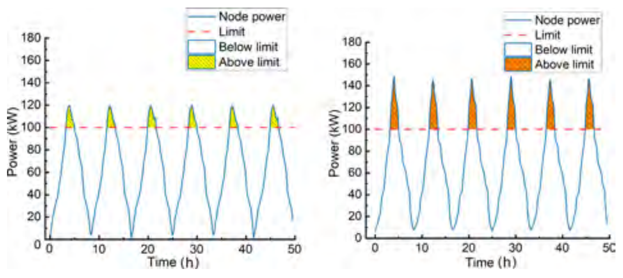
for placement in the same power node can be evaluated. In this article, an asynchronous score[26] is used to assess,

$$A_i = \Gamma(M_i) = \frac{\sum_{j \in M_i} peak(sP_j)}{peak(\sum_{j \in M_i} sP_j)} \tag{1}$$

where A_i is the asynchronous score of $rack_i$, M_i is the set of the indexes of servers that have been assigned to $rack_i$, $peak(sP_j)$ is the peak power of $server_j$, and $peak(\sum_{j \in M_i} sP_j)$ is the peak power sum of all servers in M_i. A higher asynchronous score indicates less power overlap among servers and more suitable to be placed together. Conversely, it means that the more power overlap, the less suitable to be assigned together.

3.2 Power supply risk

When more servers are deployed for power oversubscription in a data center, there is a need to balance the power utilization and power supply risk. Quantitative metrics need to be used in this procedure to assess the power supply risk of the power nodes. Reference [18] used the overload probability as a risk evaluation metric, i.e., the percentage of time that a power overload occurs during the observed time. Although the overload probability can reflect the power supply risk partly, it does not consider the impact of the power overload magnitude. For example, as shown in Fig. 2, although the power curves of Figs. 2a and 2b have the same overload probability, the power curve of Fig. 2b poses a more serious power supply risk because of its greater overload magnitude. To describe both the impact of overload frequency and overload magnitude on the supply risk, we propose an evaluation metric



(a) Small overload magnitude (b) Large overload magnitude

Fig. 2 Two power curves with the same overload frequency but different overload amplitudes.

called WPSRisk based on the circuit breaker tripping curve.

The WPSRisk indicates that the overload current corresponding to the weight is calculated using the tripping curve of the circuit breaker. To highlight the significance of the overload magnitude on the risk to the power supply, a higher overload current corresponds to a greater weight. In Ref. [19], the authors stated that the ideal upper limit of the safe power oversubscription is the lower limit of the tolerance band of the circuit breaker tripping curve. Specifically, as shown in Fig. 3, the tripping curve is a tolerance band consisting of two curves that describe how long a circuit breaker will trip at different magnitudes of current for a continuous period of time , where the horizontal axis is the ratio of the overload current to the rated current of the circuit breaker, and the vertical axis is the tripping time. We consider the front half of the tripping curve, corresponding to the thermal tripping for overload protection, which works on the principle that the overcurrent causes the conductor

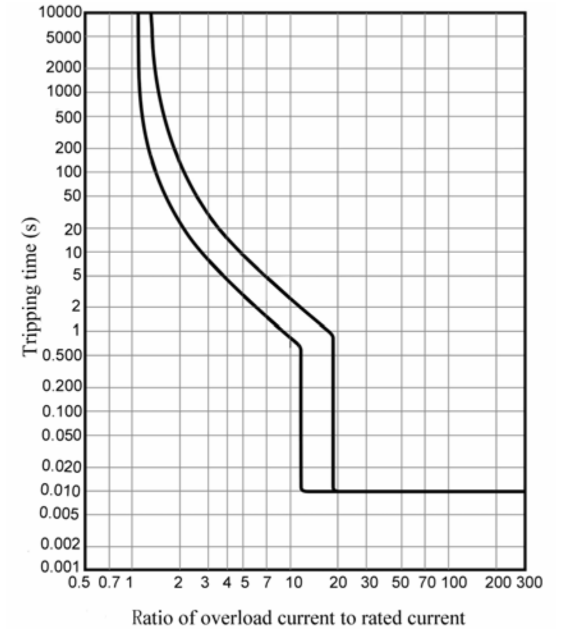


Fig. 3 Schneider EasyPact CVS100E (20A) tripping curve[28] (both the horizontal and vertical axes are logarithmic axes).

to heat up, so that the bimetallic strip is deformed by heat, and when the deformation reaches a certain level it will trigger a trip. We use the curve corresponding to the lower limit of the tolerance band to calculate the weight of the overload current.

In the actual data center production environment, the overload current of the power supply node generally does not surpass the rated current I_r of the circuit breaker by a significant degree. In this article, the current of the rack is limited to less than BI_r when performing the optimization of server placement (B is a manually set parameter). Therefore, we only consider the tripping curve below BI_r when performing the risk evaluation. To obtain a function of the lower limit curve of the tolerance band, a linear interpolation can be used between the horizontal axis 1.05 and B to obtain a function $C(\eta)$ of the lower limit of the tripping time with respect to η (where η is the ratio of overload current to rated current). We set the weight corresponding to η as the reciprocal of the lower limit of the tripping time $C(\eta)$, indicating the rate of heat accumulation in the circuit breaker at that overload current, with a larger value implying a significant contribution to the power supply risk. According to IEC 60947-2, the conventional non-tripping current of a circuit breaker is 1.05 times its rated current[29]. In other words, when the current remains below this threshold ($\eta = 1.05$), no tripping is guaranteed for at least two hours. Therefore, when $\eta < 1.05$, the current can be seen as not causing a risk to the supply, corresponding to a weight of 0. The calculation of the weight $K(\eta)$ can be expressed as the following segmentation function:

$$K(\eta) = \begin{cases} 0, & 0 \leqslant \eta \leqslant 1.05; \\ \frac{1}{C(\eta)}, & 1.05 \leqslant \eta \leqslant B \end{cases} \tag{2}$$

Based on the above weight calculation, we use the power supply risk metric WPSRisk as shown in Eq. (3), whose magnitude range is [0, 1], with larger values representing higher risk,

$$risk_i = \frac{\int_0^{+\infty} P_i(z) \times K(\frac{z}{limit_i})\mathrm{d}z}{K(B)} \tag{3}$$

where $risk_i$ is the risk metric of $rack_i$, $P_i(z)$ is the Probability Distribution Function (PDF) of the power of $rack_i$, which can be calculated from the historical power data, and $limit_i$ is the rated power of the circuit breaker at $rack_i$.

3.3 Optimization problem definition

The server placement optimization problem can be formulated as follows: at the rack level, there are n racks and there are currently m servers to be deployed to each rack. Let $R = \{rack_1, rack_2, \ldots, rack_n\}$ be the set of all racks, $S = \{server_1, server_2, \ldots, server_m\}$ be the set of all servers, and X be a solution to this placement optimization problem, which is a zero-one matrix of dimension (m, n), and each element x_{ij} inside the matrix represents whether $server_i$ is allocated to $rack_j$, as shown below:

$$x_{ij} = \begin{cases} 1, & \text{if } server_i \text{ is assigned to } rack_j; \\ 0, & \text{otherwise,} \end{cases}$$
$$\forall i \in \{1, 2, \ldots, m\} \text{ and } \forall j \in \{1, 2, \ldots, n\} \tag{4}$$

The server placement method proposed in this article takes into account both power utilization and power supply risk in a bi-objective optimization, to minimize both the sum of peaks and the maximum value of the WPSRisk of the racks WPSRisk$_{\text{max}}$. The former minimizing the sum of peaks means that the power utilization of racks is improved. The latter using WPSRisk$_{\text{max}}$ instead of the average value WPSRisk$_{\text{avg}}$ is to avoid the case, which most of the racks with small risks while a small number of racks have extremely high risks. In addition, to reduce the effect of the numerical size of power values, we convert the minimized sum of peaks into the minimized average peak power ratio. The peak power ratio of $rack_i$ is calculated in Eq. (5) and the average peak power ratio is calculated in Eq. (6),

$$r_i = \frac{peak(rP_i)}{limit_i} \tag{5}$$

$$f_1(X) = \frac{\sum_{i=1}^{n} r_i}{n} \tag{6}$$

where rP_i is the power of $rack_i$, $limit_i$ is the rated power of the circuit breaker at $rack_i$, X is the server placement solution, and r_i is the peak power ratio of $rack_i$. Since the rack current is limited to be less than BI_r, the value of r_i is taken to be in the range [0, B]. As another optimization objective, the WPSRisk$_{\max}$ is expressed as

$$f_2(X) = \max(risk_i),\ \ \forall i \in \{1, 2, \ldots, n\} \tag{7}$$

There are three main constraints to be considered when performing server placement. (1) Each server is need to be placed, (2) the peak power of each rack must not exceed BI_r, and (3) the number of servers placed on each rack must not exceed the maximum number. In summary, the server placement optimization problem can be defined as follows:

$$\text{minimize } F(X) = f_1(X) + c \times f_2(X) \tag{8}$$

$$\text{s.t., } \sum_{j=1}^{n} x_{ij} = 1,\ \ \forall i \in \{1, 2, \ldots, m\} \tag{9}$$

$$peak\ (rP_i) \leqslant B \times limit_i,\quad \forall i \in \{1, 2, \ldots, n\} \tag{10}$$

$$serverCount_i \leqslant NumLimit_i,\quad \forall i \in \{1, 2, \ldots, n\} \tag{11}$$

where c is the power supply risk penalty factor, $serverCount_i$ and $NumLimit_i$ are the number of servers and the maximum number of servers that can be placed in $rack_i$, respectively, and B represents the upper limit of the power overrun level, which indicates the maximum allowed ratio of the peak current of the rack to the rated current of the circuit breaker.

4 HGAACS-Based Server Placement Optimization

4.1 Algorithm design

In this article, we propose HGAACS algorithm that combines ACS and GA for the server placement problem. To improve the performance of the solution, HGAACS enhances the global search ability at the later stage by adding selection, mutation, and crossover to the ACS, thus avoiding the search from stalling to some extent. Moreover, the positive feedback mechanism of ACS also improves the problem of weak local search ability in GA. In addition, to improve the blindness of mutation, we use priori knowledge to modify the mutation to accommodate the server placement optimization problem.

In the initialization phase of HGAACS, the pheromone of each server-pair needs to be initialized to τ_0 and an initial server is randomly set for each rack. After the initialization is completed, Algorithm 1 starts to iterate, and Fig. 4 shows the flow chart of the HGAACS algorithm in one iteration. At the G-th iteration, the initial solutions are first constructed by ants and saved to $Swarm_G$. After that, tournament selection is performed on $Swarm_G$, and the selected population is mutated to get a new population and saved to $tSwarm1_G$. Next, the crossover is performed on $tSwarm1_G$, and the population after the crossover is amended to obtain $tSwarm2_G$. Finally, the optimal solution in $Swarm_G$, $tSwarm1_G$, and $tSwarm2_G$ is taken as the contemporary optimal solution X^{ibest}, and if this solution is better than the current global optimal solution, X^{gbest} is updated. In addition, the global pheromone update is performed at the end of each iteration. HGAACS needs to keep repeating the above iterative process until the termination condition is triggered. Algorithm 1 is the general flow pseudo-code of the HGAACS-based server placement optimization algorithm.

Algorithm 1 HGAACS workload placement

```
Input: $sP, m, n$
Output: $Solution$
1:  for each $(a, b) \in \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ do
2:      $\tau_{a,b} = \tau_0$;
3:  end for
4:  for $k$ from 1 to $n$ do
5:      Choose an unplaced server $server_w$;
6:      $initDistribution_k = w$;
7:  end for
8:  for $G$ from 1 to $K$ do
9:      $Swarm_G \leftarrow$ConstructAnts $(initDistribution)$;
10:     Update $F(Swarm_G^t)$ of $Swarm_G^t$ by Eq. (9) for $t \in \{1, 2, \ldots, L\}$;
11:     $tSwarm1_G \leftarrow$ Selection($Swarm_G$);
12:     $tSwarm1_G \leftarrow$ Mutation($tSwarm1_G$);
13:     $tSwarm2_G \leftarrow$ Crossover($tSwarm1_G$);
14:     Update $F(tSwarm1_G^t)$ of $tSwarm1_G^t$ for
        $t \in \{1, 2, \ldots, L\}$;
15:     Update $F(tSwarm2_G^t)$ of $tSwarm2_G^t$ for
        $t \in \{1, 2, \ldots, L\}$;
16:     Update $X^{ibest}$ and $X^{gbest}$ by
        $(Swarm_G, tSwarm1_G, tSwarm2_G)_{best}$;
17:     Apply global pheromone update rule;
18: end for
19: return $X^{gbest}$
```

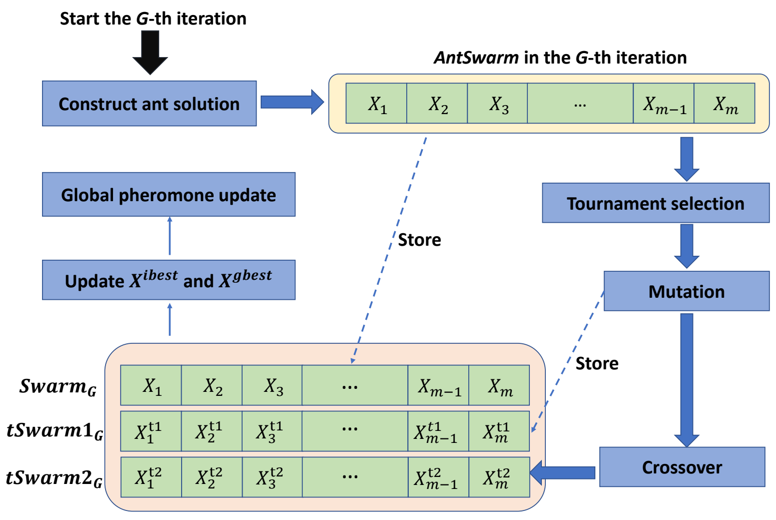


Fig. 4 Iteration process of HGAACS.

4.2 ACS

In the HGAACS-based server placement algorithm proposed in this work, the solution is first constructed initially by ACS at each iteration. This section will describe how the solution to the server placement problem is constructed by ACS.

4.2.1 Pheromone structure

For the server placement optimization problem, since each ant constructing a solution needs to assign each server to an appropriate rack based on the historical experience provided by the pheromone, a simple approach is to create pheromone for the mapping between servers and racks. However, for this problem, the individual rack can be viewed as being homogeneous, and it is not the racks themselves that affect the quality of the distribution, but the set of servers that already exist within each rack. Therefore, we adopt a pheromone structure similar to that used in Refs. [30, 31], which changes the direct recording of preferences between servers and individual racks to the recording preferences between servers and server sets. We use an $m \times m$ matrix to record the pheromones among individual servers (where m is the number of servers). When calculating the pheromone between $server_k$ and $rack_i$, the average of the pheromones between the servers already deployed in $rack_i$ and $server_k$ is used, as shown below:

$$T_{k,i} = \frac{\sum\limits_{j \in M_i} \tau_{k,j}}{|M_i|} \tag{12}$$

where $T_{k,i}$ is the pheromone between $server_k$ and $rack_i$, and $\tau_{k,j}$ is the pheromone between $server_k$ and $server_j$.

4.2.2 Constructing solutions

In ACS, ants search for solutions based on pheromones as well as heuristic information. For the server placement optimization problem, each ant needs to select an appropriate rack for each server by using state transfer rules when building the solution. ACS uses a method called pseudo-random-proportional-rule for state transfer as follows: When making the assignment of $server_k$, the probability p_k assigned to each rack is first calculated based on the rules using pheromones as well as heuristic information firstly. Then, a uniform random number q in range [0, 1] is generated. When $q \leqslant q_0$, ACS performs exploration: assign $server_k$ to $rack_t$ with the largest $p_{k,t}$. Otherwise, biased exploration is performed: The roulette wheel selection is used to select which rack to assign based on the probability distribution p_k. The state transfer rule can be expressed as follows:

$$s = \begin{cases} \arg\max\limits_{i \in \{1,2,\dots,n\}} p_{k,i}, & \text{if } q < q_0; \\ S, & \text{otherwise} \end{cases} \tag{13}$$

where s is the selected rack, q_0 is the parameter, S refers to the rack determined by roulette wheel selection, and $p_{k,i}$ is the probability that $server_k$ is assigned to $rack_i$, which is calculated as follows:

$$w_{k,i} = \begin{cases} T_{k,i}^{\alpha} \times \eta_{k,i}^{\beta}, & \text{if } server_k \text{ can be} \\ & \text{assigned to } rack_i; \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$p_{k,i} = \frac{w_{k,i}}{\sum\limits_{j=1}^{n} w_{k,j}} \tag{15}$$

In Eq. (14), $T_{k,i}$ is the pheromone between $server_k$ and $rack_i$, which is calculated by Eq. (12), $\eta_{k,i}$ is the heuristic value between $server_k$ and $rack_i$, α and β are parameters that together determine the relative importance of historical experience and heuristic value for ants to make decisions.

In this work, the asynchronous scoring defined in Eq. (1) is used as heuristic value, so that servers are assigned to racks with lower peak interval overlap in priority. The calculation of the heuristic value is as follows:

$$\eta_{k,i} = \Gamma(\{M_i, k\}) = \frac{peak(rP_i) + peak(sP_k)}{peak(rP_i + sP_k)} \quad (16)$$

It should be noted that, as shown in Eq. (14), during the process of server placement, it is necessary to check whether the Formulas (10) and (11) are violated after assigning this server to each rack. If there exists $rack_v$ that does not meet the constraints in Formulas (10) and (11), then $w_{k,v}$ will be set to 0 in order to let $p_{k,v}$ become 0 as well, thus avoiding the $rack_v$ from being selected.

During the iterative process of ACS, the state transfer rules in Eq. (13) are used to assign each server, so as to construct the solution for each ant. The solutions constructed by the ants is an $L \times m$ matrix (where L is the population size), and each row in the matrix is the solution of an individual ant, and each element represents the serial number of the rack to which the server is assigned and takes an integer in the range $[1, n]$. The pseudo code in Algorithm 2 describes the process of constructing solutions by ants.

Algorithm 2 Construct ants

```
Input: $initDistribution$
Output: $AntSwarm$
1:  for $i$ from 1 to $L$ do
2:      $ServerDistribution \leftarrow \{-1, -1, \ldots, -1\}_m$;
3:      for $k$ from 1 to $n$ do
4:          $C \leftarrow initDistribution_k$;
5:          $ServerDistribution_C \leftarrow k$;
6:      end for
7:      for $j$ from 1 to $m$ do
8:          if $ServerDistribution_j == -1$ then
9:              for $r$ from 1 to $n$ do
10:                 Computer $p_{j,r}$ by Eq. (15);
11:             end for
12:             Generate a uniform random number $q$ in [0, 1];
13:             if $q < q_0$ then
14:                 $s \leftarrow argmax_{r \in \{1,2,\ldots,n\}} p_{j,r}$;
15:             else
16:                 Randomly choose a rack $rack_{target}$ with probability $p_j$;
17:                 $s \leftarrow target$;
18:             end if
19:             $ServerDistribution_j \leftarrow s$;
20:         end if
21:     end for
22:     $AntSwarm_i \leftarrow ServerDistribution$;
23:     Apply local pheromone update rule in Eq. (17);
24: end for
25: return $AntSwarm$
```

4.2.3 Pheromone update

There are two pheromone update modes in ACS: local pheromone update and global pheromone update.

(1) Local pheromone update: In ACS, each ant needs to perform local pheromone update after constructing a solution: update the pheromone between each server-pair $(server_a, server_b)$ on the same rack. The update rule is as follows:

$$\tau_{a,b} = (1-\rho) \times \tau_{a,b} + \rho \times \tau_0 \quad (17)$$

where ρ the local pheromone evaporation coefficient, and τ_0 is the initial value of pheromone.

(2) Global pheromone update: After all ants have constructed their solutions, HGAACS needs to perform selection, mutation, and crossover on the population in turn. After completing these operations, the algorithm starts the global pheromone update, which has the feature of only performing pheromone updates on the contemporary optimal solution. The update rule is as follows:

$$\tau_{a,b} = \begin{cases} (1-\mu) \times \tau_{a,b} + \mu \times \Delta\tau_i, & \text{if } a, b \in M_i \text{ and } M_i \in X^{ibest}; \\ \tau_{a,b}, & \text{otherwise} \end{cases} \quad (18)$$

where X^{ibest} is the contemporary optimal solution, μ is the global pheromone evaporation coefficient, and $\Delta\tau_i$ is the increment of pheromone among the servers in $rack_i$, which is calculated as follows:

$$\Delta\tau_i = \frac{1}{F(X^{ibest})} + A_i \quad (19)$$

where $F(X^{ibest})$ is the value of the objective function of the optimal individual, and A_i is the asynchronous score defined in Eq. (1) and takes different values for each rack, depending on the degree of power pattern asynchrony among its servers. The higher the degree of asynchrony, the larger the pheromone increment among the servers on the rack.

4.3 GA

After the ants construct the solutions, HGAACS needs to apply GA-related operations to the solutions: selection, mutation, and crossover. This section explains the encoding form of the population when performing these operations, and the changes to the mutation operator and crossover operator in our work.

4.3.1 Encoding scheme

In GA, the individual encoding scheme of the population is the same as that of the solutions constructed by ants, and the integer coding is used. The code length of an individual is the number of servers m, and an integer in the individual code corresponds to the index of the rack assigned to the server. For example, in individual x, $x_i = k$ represents the assignment of $server_i$ to $rack_k$. Figure 5 is an example of applying this encoding scheme to represent a server placement solution.

4.3.2 Mutation operator

In order to improve the search efficiency of the algorithm, we use a priori knowledge to modify the mutation operator in order to make the algorithm more adaptable to the server placement problem. In the modified mutation operator, there are two different modes of mutation. Mutation mode 1: The operator randomly select a server from the rack for migration, and then randomly select another rack which can satisfy the server placement constraints as the destination rack. Mutation mode 2: A server is randomly selected and the asynchronous score among the server and other racks is calculated. Then the racks are sorted in descending order according to the asynchronous score, and the server is migrated to the first suitable rack.

When the population performs the mutation, for each rack of each chromosome, there are probabilities of Pm_1 and Pm_2 to perform mutation mode 1 and mutation mode 2, respectively. Additional, there is a probability of $1 - Pm_1 - Pm_2$ for no operation to be performed. We use the Roulette Wheel method to determine which operation to perform for each rack.

4.3.3 Crossover operator

The crossover operator in HGAACS uses two-point crossover, as shown in Fig. 6. It should be noted that after the crossover operation, the solutions that do not meet the constraints may be generated, so it is necessary to check the feasibility of each individual after the crossover. For the infeasible individuals, the amending operation is as follows: A server is randomly selected from that illegitimate rack, and then migrated to another rack using a method similar to mutation mode 2 without violating the constraints. If there is no other rack capable of accommodating that server, then another server is randomly selected to try to migrate. The above operation will be repeated until the rack meets the constraints.

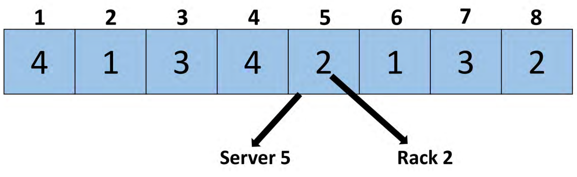


Fig. 5 Individual encoding scheme.

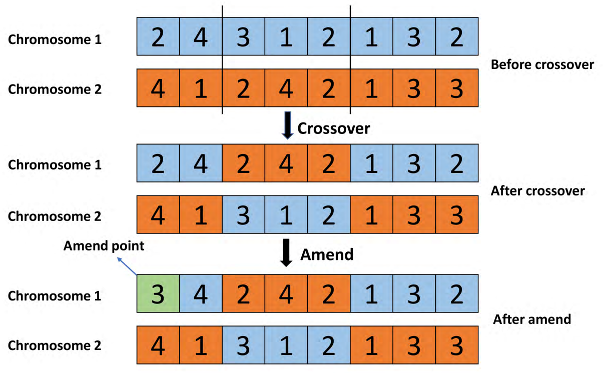


Fig. 6 Example of crossover. The maximum number of servers in *rack*$_2$ is 3. After the completion of the crossover operation, Chromosome 1 violates the constraint specified in Formula (11). Therefore, in the amending operation, *server*$_1$ is migrated from *rack*$_2$ to *rack*$_3$ for amending.

4.3.4 Time complexity analysis

The time complexity of HGAACS primarily depends on three components: ACS, GA, and fitness computation. We define the maximum number of iterations as K, the population size as L, the number of racks as n, the number of servers as m, and the length of the server history power consumption data sequence as T. Then the time complexity of the fitness calculation is $O(K \times L \times T \times (m + n))$. The time complexity of the ACS depends on the process of constructing the ant solutions, which is $O(K \times L \times m \times (m + n \times T))$. The time complexity of the GA depends on the tournament selection, mutation, and crossover, and since each rack after crossover needs to be amended in the worst case, the time complexity of the GA is $O(K \times L \times (L + n^2 \times (\log n + T)))$. To sum up, the total time complexity of the HGAACS is $O(K \times L \times (m^2 + m \times n \times T + L + n^2 \times (\log n + T)))$.

5 Experiment

5.1 Datasets and data pre-processing

The source of the original dataset for the experiments is the Alibaba cluster dataset[32], which contains about 4000 machines. Since the input of the algorithm is time series data of power consumption per machine, the data used by in our experiments are mainly the following three fields of the machine_usage table in the dataset: machine ID, timestamp, and CPU utilization. Since most of the machines in this dataset are missing some data from the second day, we select data for the experiments with a total of 864 timestamps for six days, from the third to the eighth day.

Since there is no information about the power consumption of the machines and their models in the dataset, we selected five server models from SPECpower for power modeling to construct the data centers with heterogeneous servers. Figure 7 shows the variation of power with CPU utilization for these five types of servers. Since CPU utilization is a major influence on server power[3, 33], server power can be modeled using the power data provided by SPECpower at different load levels. In our experiments, we employ linear interpolation to model server power sP as a segmented function of CPU utilization u. This approach is commonly used for power modeling in various works, including CloudSim[34] and DCSim[35]. This power modeling method can be expressed as follows:

$$P(u)=\begin{cases} k_1u+b_1, & 0\leqslant u<0.1;\\ k_2u+b_2, & 0.1\leqslant u<0.2;\\ k_3u+b_3, & 0.2\leqslant u<0.3;\\ & \vdots \\ k_{10}u+b_{10}, & 0.9\leqslant u<1\end{cases} \tag{20}$$

In addition, most of the servers in the original dataset have the same power consumption pattern, and this power consumption pattern is noted as P_1. In order to obtain a dataset with more complex and diverse power consumption patterns, we perform the following five operations on the original data to get different server power consumption patterns: rolling 50 timestamps, rolling 100 timestamps, rolling 700 timestamps, inverting the sequence, and rolling 300 timestamps after inverting the sequence. We denote the five different server power consumption patterns obtained as P_2, P_3, P_4, P_5, and P_6.

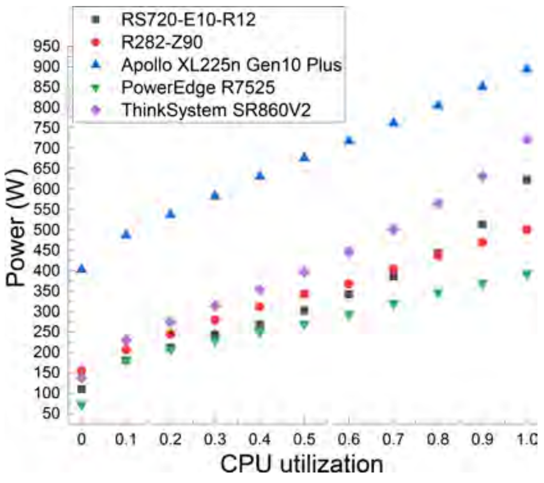


Fig. 7 Power of the five types of servers for different CPU utilization.

To test the performance of the proposed algorithm in data centers of different sizes, we set up three different sizes of data center entity types in our experiments, containing 120, 240, and 360 servers, respectively. Two test entities are set up for each size of data center, one of which has three power consumption patterns, P_1, P_2, and P_3, and the number ratio between the three power consumption patterns is 1:1:1. The other entity is a combination of servers with six power consumption patterns, P_1, P_2, P_3, P_4, P_5, and P_6, and the number of the six patterns is the same. Table 2 shows the information of all data center entities used in the experiments. It's worth noting that in each data center entity, there is an equal number of servers for each type of hardware.

5.2 Experimental setup

In our experiments, we compare the performance of HGAACS with the following five server placement algorithms: Random Placement, First Fit Decreasing (FFD), SmoothOperator, GA-based server placement algorithm, and ACS-based server placement algorithm. A brief description of these algorithms is given below.

(1) Random Placement. It shuffles the order of all servers and assigns them evenly to each rack.

(2) FFD[36]. It is a heuristic algorithm that assigns servers by first sorting the racks in descending order by available power budget and then deploying the server in the first rack that can accommodate it.

(3) SmoothOperator[26]. It uses the k-means algorithm to classify servers into several classes according to the difference in power consumption patterns, and then uses Round-Robin to evenly distribute servers of the same class to each rack.

(4) GA-based server placement algorithm[27]. It uses the same genetic operators as in the basic GA.

Table 2 Information about data center entities.

Data center	Number of servers	Number of racks	Power consumption pattern
$D1$	120	12	P_1, P_2, P_3
$D2$	180	18	P_1, P_2, P_3
$D3$	240	24	P_1, P_2, P_3
$D4$	120	12	$P_1, P_2, P_3, P_4, P_5, P_6$
$D5$	180	18	$P_1, P_2, P_3, P_4, P_5, P_6$
$D6$	240	24	$P_1, P_2, P_3, P_4, P_5, P_6$

(5) ACS-based server placement algorithm[31]. ACS is applied to the VM Placement problem (VMP) in Ref. [31], and we modify it to apply to the server placement problem.

In order to verify the performance of the above algorithms, each algorithm was experimented under several different data center entities. Each algorithm is run 10 times on each experimental setup, and the algorithm performance is averaged over 10 runs. In the experiments, the rack power threshold is 4 kW, the maximum number of servers on a rack is 20, and the upper limit of the power oversubscription level B is 1.5. In addition, we set all the circuit breakers used in the racks to Schneider EasyPact CVS100E (20A), whose tripping curve is shown in Fig. 3. Finally, the algorithm performance and runtime of GA, ACS, and HGAACS are closely related to the algorithm parameters. Table 3 shows the main parameter settings that enable each algorithm to achieve a high level of performance.

5.3 Experimental results

There are two algorithm performance evaluation metrics for the experiments: the sum of peaks and $\text{WPSRisk}_{\text{max}}$. The former represents power utilization and the latter represents power supply risk. The value of the risk penalty weight c in Eq. (8) has a significant impact on the performance of GA, ACS, and HGAACS, so in our experiments we tested 11 values of c from 0 to 1 in steps of 0.1. Figure 8 shows the effect of different values of c on the two optimization objectives in the experiments under the data center $D4$ (the result is the average of ten runs).

According to Fig. 8, when $c = 0$, $\text{WPSRisk}_{\text{max}}$ is large for all three algorithms, while the sum of peaks is small, due to its equivalent to performing a single-objective server placement optimization with the goal of minimizing the sum of peaks. When $c > 0$, a dual-objective server placement optimization with the objective of minimizing the sum of peaks as well as minimizing $\text{WPSRisk}_{\text{max}}$ is performed. Comparing the case of $c = 0$, $\text{WPSRisk}_{\text{max}}$ of all three algorithms decreases significantly (using HGAACS as an example, $\text{WPSRisk}_{\text{max}}$ at $c = 0.1$ is only about 0.8% of that at $c = 0$), while the sum of peaks increases only slightly (also using HGAACS as an example, the sum of peaks at $c = 0.1$ is only about 0.097% higher than that at $c = 0$), and as c increases, $\text{WPSRisk}_{\text{max}}$ generally shows a decreasing trend, while the change of the sum of peaks is irregular and small.

The results in Fig. 9 illustrate that all three algorithms are able to get $\text{WPSRisk}_{\text{max}}$ significantly lower at a small cost (a slight increase in the sum of peaks) by setting c to be greater than 0. Figure 9 shows the power over time for all racks for one run of HGAACS in data center $D4$ when c is set to 0, 0.2, and 0.5, respectively. The values of power supply risk in Figs. 9b and 9c are significantly lower than those in Fig. 9a, which further illustrates that the algorithms can significantly reduce the power supply risk when setting c to a number greater than 0.

Tables 4 and 5 record the sum of peaks and $\text{WPSRisk}_{\text{max}}$ of the six algorithms under each data center entity, respectively, and the data in this two tables are the mean and standard deviation (std) of ten runs (the value of c is set to 0.4).

Analysis of Table 4 shows that HGAACS outperforms the other five algorithms for all six data center entities in sum of peaks. the average sum of peaks of HGAACS is reduced by 4.18% to 4.95% under the six data center

Table 3 Algorithm parameter.

Algorithm	L	K	P_c	P_m	Pm_1	Pm_2	α	β	ρ	μ	q_0	τ_0
GA	200	100	0.85	0.1	–	–	–	–	–	–	–	–
ACS	100	100	–	–	–	–	4	2	0.01	0.02	0.8	0.1
HGAACS	100	100	0.7	–	0.1	0.05	4	2	0.01	0.02	0.8	0.1

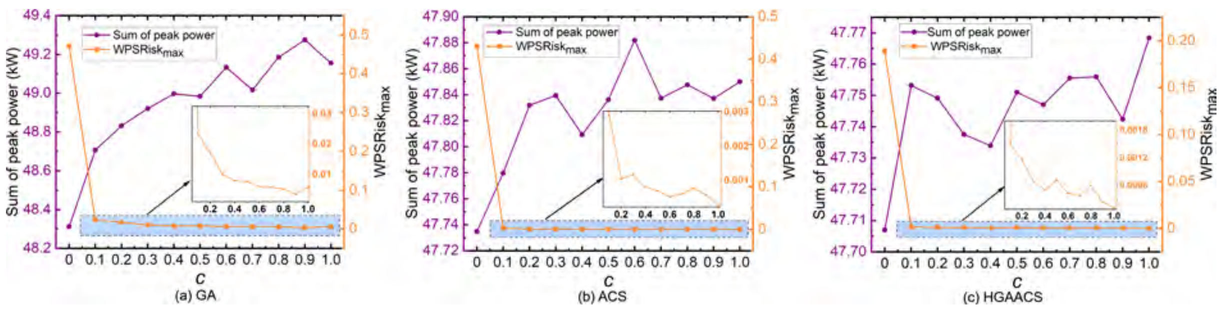


Fig. 8 Sum of peaks and $\text{WPSRisk}_{\text{max}}$ of the three algorithms for different values of c.

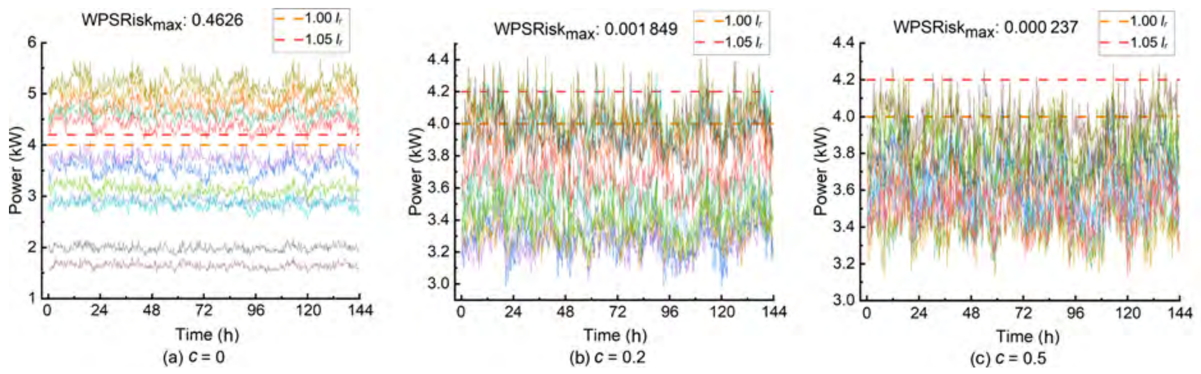


Fig. 9 Power curves for all racks in *D*4 obtained by HGAACS at different *c* value settings.

Table 4 Comparison of the sum of peaks for the six algorithms (mean ± std). (kW)

Data center	Random placement	FFD	SmoothOperator	GA	ACS	HGAACS
$D1$	52.48±0.32	52.34±0.44	51.12±0.28	51.14±0.15	50.00±0.024	**49.98±0.017**
$D2$	78.73±0.52	78.79±0.42	76.73±0.30	77.34±0.33	74.92±0.049	**74.83±0.036**
$D3$	104.97±0.50	105.18±0.48	102.84±0.74	104.62±0.45	100.61±0.032	**100.58±0.013**
$D4$	50.10±0.33	49.93±0.54	48.71±0.19	49.00±0.11	47.84±0.052	**47.73±0.035**
$D5$	75.16±0.42	75.69±0.62	73.07±0.18	74.26±0.38	71.82±0.033	**71.75±0.040**
$D6$	100.16±0.47	100.05±0.47	97.85±0.29	99.52±0.45	95.88±0.057	**95.78±0.050**

Table 5 Comparison of WPSRisk$_{max}$ for the six algorithms (mean ± std).

Data center	Random placement	FFD	SmoothOperator	GA	ACS	HGAACS
$D1$	0.071±0.043	0.0037±0.002	0.043±0.053	0.011±0.004	0.0049±0.0060	**0.0023±0.0020**
$D2$	0.072±0.035	0.0093±0.010	0.032±0.059	0.019±0.009	0.0031±0.0010	**0.0020±0.0007**
$D3$	0.120±0.060	0.0078±0.004	0.022±0.020	0.042±0.009	0.0040±0.0050	**0.0012±0.0006**
$D4$	0.110±0.092	0.0023±0.002	0.099±0.095	0.008±0.005	0.0008±0.0007	**0.0005±0.0004**
$D5$	0.140±0.010	0.0028±0.002	0.026±0.053	0.018±0.009	0.0010±0.0010	**0.0004±0.0003**
$D6$	0.190±0.047	0.0037±0.003	0.052±0.048	0.045±0.015	0.0011±0.0007	**0.0006±0.0004**

entities compared to Random Placement. Compared to FFD, the reduction is 4.27% – 5.21%. Compared to SmoothOperator, the reduction is 1.81% – 2.48%. Compared to GA, the reduction is 2.27% – 3.86%. Compared with ACS, the average sum of peaks also has a slight decrease, with a reduction of 0.03% – 0.23%. Random Placement, FFD, SmoothOperator, and GA, all four algorithms have a large standard deviation of results, especially Random Placement and FFD, which have a more unstable performance. In contrast, ACS and HGAACS exhibit relatively small standard deviations, which indicates that they are more stable.

Based on the analysis of the data in Table 5, it is clear that HGAACS shows remarkable improvement in all six data center entities for WPSRisk$_{max}$ compared to the other five algorithms. Comparing to Random Placement, the average WPSRisk$_{max}$ decreases from 96.76% to 99.71% under the six data center entities. Comparing to FFD, the decrease is 37.84% – 85.71%. Comparing to SmoothOperator, it decreases from 93.75% to 99.49%. The main reason for such a significant improvement is that SmoothOperator only considers the asynchrony score among servers, and does not consider the relative magnitude of power among them, which may assign asynchronous but high-powered servers to the same rack, resulting in individual peak power of the rack is too high. Comparing to GA, the decrease is 79.09% – 98.67%. Comparing to ACS, the WPSRisk$_{max}$ average can also have a significant decrease from 37.5% to 70%. Both Random Placement and SmoothOperator have large standard deviations, proving that they perform very erratically on this metric. In contrast, HGAACS demonstrates small standard deviations under each data center entity, reflecting its ability to consistently achieve stable and favorable results for WPSRisk$_{max}$.

By comprehensively analyzing the performance of each algorithm in terms of sum of peaks and WPSRisk$_{max}$, it can be seen that the HGAACS proposed in this article has a great advantage over the other five algorithms. Compared with Random Placement, FFD, SmoothOperator, and GA, HGAACS has a significant improvement in both peak power sum and $\text{WPSRisk}_{\text{max}}$. Compared to ACS, despite only a slight improvement in the sum of peaks, it achieves a significant enhancement in $\text{WPSRisk}_{\text{max}}$.

To further compare the convergence ability of the three swarm intelligence algorithms, GA, ACS, and HGAACS, we collected data on the variation of the fitness of these three algorithms for one run under the six data center entities when setting c to 0.4, which is shown in Fig. 10. Based on the analysis of Fig. 10, it is evident that HGAACS outperforms GA and ACS in terms of the number of iterations required to find highly fit solutions. Additionally, HGAACS consistently achieves the best fitness across data center entities of all sizes and types, indicating its superior convergence speed and search ability compared to GA and ACS. This is because the combination of ACS and GA brings together the strengths of both algorithms. ACS contributes its excellent local search ability, while genetic operators, like mutation and crossover, introduce perturbations that enhance the algorithm's global search ability. The use of genetic operators prevents the search process from getting stuck in later stages and empowers the algorithm to explore and find better solutions effectively.

6 Conclusion and Outlook

In this article, we first analyze the demand for increased power utilization in data centers and the potential power supply risk issues associated with it. Then we propose a power supply risk evaluation metric WPSRisk, and a server placement algorithm HGAACS that can improve the power utilization of the data center while taking into account power supply security. We compare HGAACS with five other algorithms, and the results show that HGAACS is able to reduce the sum of peaks by 4.18% to 4.95% and $\text{WPSRisk}_{\text{max}}$ by 96.78% to 99.72% compared to random server placement in data centers of different sizes and types. These results indicate an increase in power utilization and the ability of the data center to handle a larger number of workloads. Meanwhile, compared with the placement method SmoothOperator proposed in the existing work, the sum of peaks and $\text{WPSRisk}_{\text{max}}$ can be reduced by at least 1.81% and 93.75%, respectively, which reflects the performance advantage of the HGAACS algorithm. Finally, server placement optimization does not conflict with the risk management measures commonly used in power oversubscription data centers. Moreover, utilizing the methods proposed in this paper for server placement can effectively reduce power supply risks, thus mitigating

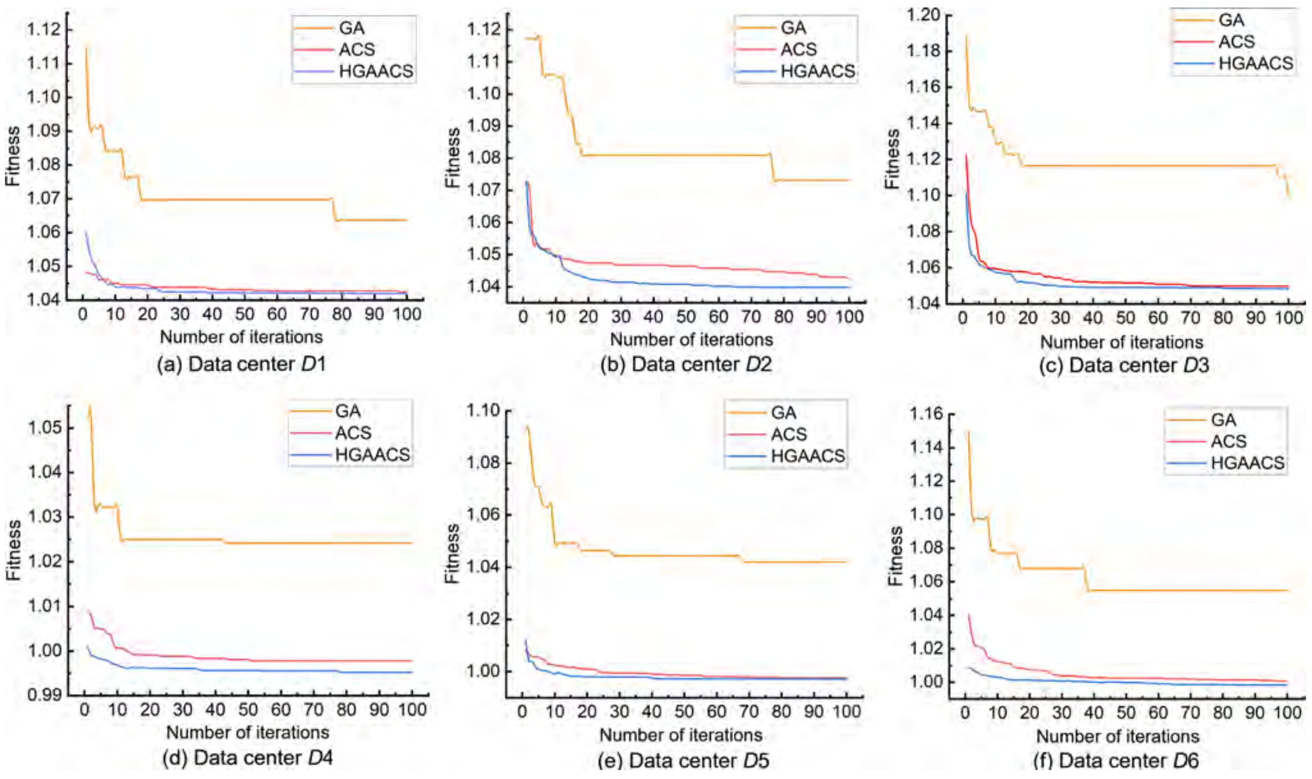


Fig. 10 Convergence analysis of GA, ACS, and HGAACS under each data centers entity (c = 0.4).

the negative impact of other risk response measures, such as performance degradation and SLA due to power capping.

Current research granularity in this article is the placement of each server in the data center. In the future, we plan to integrate it with workload scheduling to further improve power utilization as well as to reduce power supply risks.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62072187), the Guangdong Major Project of Basic and Applied Basic Research (No. 2019B030302002), the Guangzhou Science and Technology Program Key Projects (No. 202007040002), the Guangdong Marine Economic Development Special Fund Project (No. GDNRC[2022]17), and the Guangzhou Development Zone Science and Technology Project (Nos. 2021GH10 and 2020GH10).

References

[1] L. A. Barroso, U. Hölzle, and P. Ranganathan, *The Datacenter as A Computer*: *Designing Warehouse-Scale Machines*. 3rd ed. Switzerland: Springer, 2019, p. 189.

[2] Gartner forecasts worldwide it spending to exceed $4 trillion in 2022, https://www.gartner.com/en/newsroom/press-releases/2021-10-20-gartner-forecasts-worldwide-it-spending-toexceed-4-trillion-in-2022, 2021.

[3] X. Fan, W. D. Weber, and L. A. Barroso, Power provisioning for a warehouse-sized computer, in *Proc. 34th Ann. Int. Symp. Computer Architecture*, San Diego, CA, USA, 2007, pp. 13–23.

[4] M. A. Islam, S. Ren, and A. Wierman, Exploiting a thermal side channel for power attacks in multi-tenant data centers, in *Proc. 2017 ACM SIGSAC Conf. on Computer and Communications Security*, Dallas, TX, USA, 2017, pp. 1079–1094.

[5] C. Li, Z. Wang, X. Hou, H. Chen, X. Liang, and M. Guo, Power attack defense: Securing battery-backed data centers, in *Proc. 2016 ACM/IEEE 43rd Int. Symp. Computer Architecture*, Seoul, Republic of Korea, 2016, pp. 493–505.

[6] X. Hou, C. Li, J. Yang, W. Zheng, X. Liang, and M. Guo, Integrated power anomaly defense: Towards oversubscription-safe data centers, *IEEE Trans. Cloud Comput*., vol. 10, no. 3, pp. 1875–1887, 2022.

[7] Cost of data center outages, https://www.vertiv.com/globalassets/documents/reports/2016-cost-of-data-center-outages-11-11511901.pdf, 2016.

[8] B. Rountree, D. H. Ahn, B. R. De Supinski, D. K. Lowenthal, and M. Schulz, Beyond DVFS: A first look at performance under a hardware-enforced power bound, in *Proc. 2012 IEEE 26th Int. Parallel and Distributed Processing Symp. Workshops & PhD Forum*, Shanghai, China, 2012, pp. 947–953.

[9] S. Li, X. Wang, X. Zhang, V. Kontorinis, S. Kodakara, D. Lo, and P. Ranganathan, Thunderbolt: Throughput-optimized, quality-of-service-aware power capping at scale, in *Proc. 14th USENIX Conf. Operating Systems Design and Implementation* (*OSDI 20*), Vritual Event, 2020, pp. 1241–1255.

[10] S. Malla and K. Christensen, Reducing power use and enabling oversubscription in multi-tenant data centers using local price, in *Proc. 2017 IEEE Int. Conf. Autonomic Computing* (*ICAC*), Columbus, OH, USA, 2017, pp. 161–166.

[11] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, Benefits and limitations of tapping into stored energy for datacenters, in *Proc. 2011 38th Ann. Int. Symp. Computer Architecture* (*ISCA*), San Jose, CA, USA, 2011, pp. 341–351.

[12] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar, Leveraging stored energy for handling power emergencies in aggressively provisioned datacenters, in *Proc. Seventeenth Int. Conf. Architectural Support for Programming Languages and Operating Systems*, London, UK, 2012, pp. 75–86.

[13] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. M. Tullsen, and T. S. Rosing, Managing distributed ups energy for effective power capping in data centers, in *Proc. 2012 39th Ann. Int. Symp. Computer Architecture* (*ISCA*), Portland, OR, USA, 2012, pp. 488–499.

[14] B. Aksanli, E. Pettis, and T. Rosing, Architecting efficient peak power shaving using batteries in data centers, in *Proc. 2013 IEEE 21st Int. Symp. Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, USA, 2013, pp. 242–253.

[15] S. Malla, Q. Deng, Z. Ebrahimzadeh, J. Gasperetti, S. Jain, P. Kondety, T. Ortiz, and D. Vieira, Coordinated priority-aware charging of distributed batteries in oversubscribed data centers, in *Proc. 2020 53rd Ann. IEEE/ACM Int. Symp. Microarchitecture* (*MICRO*), Athens, Greece, 2020, pp. 839–851.

[16] G. Wang, S. Wang, B. Luo, W. Shi, Y. Zhu, W. Yang, D. Hu, L. Huang, X. Jin, and W. Xu, Increasing large-scale data center capacity by statistical power control, in *Proc. Eleventh European Conf. Computer Systems*, London, UK, 2016, p. 8.

[17] A. G. Kumbhare, R. Azimi, I. Manousakis, A. Bonde, F. V. Frujeri, N. Mahalingam, P. A. Misra, S. A. Javadi, B. Schroeder, M. Fontoura, and R. Bianchini, Prediction-based power oversubscription in cloud platforms, in *Proc. 2021 USENIX Ann. Technical Conf.*, *USENIX ATC 21*, Virtual Event, 2021, pp. 473–487.

[18] S. Malla and K. Christensen, The effect of server energy proportionality on data center power oversubscription, *Future Generat. Computer Syst*., vol. 104, pp. 119–130, 2020.

[19] X. Fu, X. Wang, and C. Lefurgy, How much power oversubscription is safe and allowed in data centers, in *Proc. 8th ACM Int. Conf. Autonomic Computing*, Karlsruhe, Germany, 2011, pp. 21–30.

[20] Q. Wu, Q. Deng, L. Ganesh, C. H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, and Y. J. Song, Dynamo: Facebook's data center-wide power management system, in *Proc. 2016 ACM/IEEE 43rd Ann. Int. Symp. Computer Architecture*, Seoul, Republic of Korea, 2016, pp. 469–480.

[21] Y. Li, C. R. Lefurgy, K. Rajamani, M. S. Allen-Ware, G. J. Silva, D. D. Heimsoth, S. Ghose, and O. Mutlu, A scalable priority-aware approach to managing data center server power, in *Proc. 2019 IEEE Int. Symp. High Performance Computer Architecture* (*HPCA*), Washington, DC, USA, 2019, pp. 701–714.

[22] Y. Jiang, Z. Huang, and D. H. K. Tsang, On power-peak-aware scheduling for large-scale shared clusters, *IEEE Trans. Big Data*, vol. 6, no. 2, pp. 412–426, 2020.

[23] C. Zhang, A. G. Kumbhare, I. Manousakis, D. Zhang, P. A. Misra, R. Assis, K. Woolcock, N. Mahalingam, B. Warrier, D. Gauthier, L. Kunnath, S. Solomon, O. Morales, M. Fontoura, and R. Bianchini, Flex: High-availability datacenters with zero reserved power, in *Proc. 2021 ACM/IEEE 48th Ann. Int. Symp. Computer Architecture* (*ISCA*), Valencia, Spain, 2021, pp. 319–332.

[24] H. Sun, P. Stolf, J. M. Pierson, and G. Da Costa, Multi-objective scheduling for heterogeneous server systems with machine placement, in *Proc. 2014 14th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing*, Chicago, IL, USA, 2014, pp. 334–343.

[25] M. H. Jamal, M. T. Chaudhry, U. Tahir, F. Rustam, S. Hur, and I. Ashraf, Hotspot-aware workload scheduling and server placement for heterogeneous cloud data centers, *Energies*, vol. 15, no. 7, p. 2541, 2022.

[26] C. H. Hsu, Q. Deng, J. Mars, and L. Tang, SmoothOperator: Reducing power fragmentation and improving power utilization in large-scale datacenters, in *Proc. Twenty-Third Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Williamsburg, VA, USA, 2018, pp. 535–548.

[27] L. Yan, W. Liu, and D. Bai, Temperature and power aware server placement optimization for enterprise data center, in *Proc. 2018 IEEE 24th Int. Conf. Parallel and Distributed Systems* (*ICPADS*), Singapore, 2018, pp. 433–440.

[28] EasyPact CVS catalog, https://www.schneider-electric.cn/zh/product-range/61052-easypact-cvs/#documents, 2022.

[29] IEC 60947–2: 2016-low-voltage switchgear and controlgear–part 2: Circuit-breakers, https://webstore.iec.ch/publication/25040, 2016.

[30] X. F. Liu, Z. H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, An energy efficient ant colony system for virtual machine placement in cloud computing, *IEEE Trans. Evolut. Comput.*, vol. 22, no. 1, pp. 113–128, 2018.

[31] H. Tabrizchi and M. K. Rafsanjani, Energy refining balance with ant colony system for cloud placement machines, *J. Grid Comput.*, vol. 19, no. 1, p. 7, 2021.

[32] https://github.com/alibaba/clusterdata/tree/master/cluster-trace-v2018, 2018.

[33] W. Lin, G. Wu, X. Wang, and K. Li, An artificial neural network approach to power consumption model construction for servers in cloud data centers, *IEEE Trans. Sustainable Comput.*, vol. 5, no. 3, pp. 329–340, 2020.

[34] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

[35] M. Tighe, G. Keller, M. Bauer, and H. Lutfiyya, DCSim: A data centre simulation tool for evaluating dynamic virtualized resource management, in *Proc. 2012 8th Int. Conf. Network and Service Management* (*CNSM*) *and 2012 Workshop on Systems Virtualiztion Management* (*SVM*), Las Vegas, NV, USA, 2012, pp. 385–392.

[36] A. Alahmadi, A. Alnowiser, M. M. Zhu, D. R. Che, and P. Ghodous, Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud, in *Proc. 2014 Int. Conf. Computational Science and Computational Intelligence*, Las Vegas, NV, USA, 2014, pp. 69–74.

Rui Chen received the BEng degree in computer science from South China University of Technology, China in 2022. He is currently a master student in computer science at South China University of Technology. His research interests include cloud computing and big data.

Weiwei Lin received the BEng and MEng degrees from Nanchang University, China in 2001 and 2004, respectively, and the PhD degree in computer application from South China University of Technology, China in 2007. He was a visiting scholar at Clemson University, USA from 2016 to 2017. Currently, he is a professor at the School of Computer Science and Engineering, South China University of Technology, China. His research interests include distributed systems, cloud computing, big data computing, and AI application technologies. He has published more than 150 papers in refereed journals and conference proceedings. He is the reviewer for many international journals, including $TC, TCYB, TSC, TCC$, etc. He is a senior member of CCF and a member of the IEEE.

Huikang Huang received the BEng degree in computer science and technology from Hanshan Normal University, China in 2018, and the MEng degree in computer science and theory from South China Agricultural University, China in 2021. Now, he is a PhD candidate at the School of Computer Science and Engineering, South China University of Technology, China. His research interests mainly focus on cloud computing.

Xiaoxuan Luo received the BEng degree in computer science from South China University of Technology, China in 2022, where he is currently a PhD candidate. His research interests include cloud computing and energy-efficiency optimization.