IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 3 February 2023, accepted 10 February 2023, date of publication 14 February 2023, date of current version 23 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3244881

RESEARCH ARTICLE

Particle Swarm–Grey Wolf Cooperation Algorithm Based on Microservice Container Scheduling Problem

XINYING CHEN, YUEFENG WU, AND SIYI XIAO
School of Computer and Communication Engineering, Dalian Jiaotong University, Dalian, Liaoning 116600, China

Corresponding author: Siyi Xiao (xiaosiyi1997@163.com)

This work was supported in part by the Liaoning Natural Science Foundation Program Grant 2019-MS-036; in part by the Liaoning Provincial Science and Technology Department under Grant 1655706734383; and in part by the Basic Scientific Research Projects of Colleges and Universities of Liaoning Provincial Department of Education under Grant LJKMZ20220826.

ABSTRACT In recent years, microservices have been very widely used as a new application development technology in edge computing, IoT, and cloud computing. Application containerization technology is one of its core technologies, which allows multiple containers to be deployed within the same physical node. Then a single physical node could provide different services to user. How to rationally deploy containers on a cluster of physical nodes is one of the main research directions nowadays. Although a number of researchers have modeled the microservice container scheduling problem and proposed effective solutions, there are still shortcomings, for example, the slow speed of finding the optimal solution and the tendency of the algorithm to fall into local optimality. This paper propose a Particle Swarm - Grey Wolf Cooperation Algorithm based on Microservice Container Scheduling Problem (PS-GWCA) by using particle swarm optimization algorithm (PSO) and grey wolf algorithm (GWO) in a multi-core parallel way, which enables the two algorithms to complement each other in the whole search process through the information exchange between populations. In the early of the search stage, the GWO can use its global search capability to guide the PSO to jump out of the local optimum to avoid premature convergence, and in the late of the search stage, the PSO can enhance the search capability of the GWO on the pareto optimal frontier. The experimental results show that compared with the other three algorithms, the algorithm optimizes 18.07% in network transmission cost, 14.67% in local load balancing, 20.66% in global load balancing, and 7.5% in search speed, and 5.69% in service reliability.

INDEX TERMS Intelligent optimization algorithm, PSO, microservice container scheduling, GWO, pareto optimal.

I. INTRODUCTION

In recent years, as a new application development technology, microservice has been widely used in edge computing, Internet of things, cloud computing and so on. Microservice is a cloud architecture method, which advocates dividing a complete application into a group of small services. These services can cooperate with each other to complete users' requests and provide users with the final results [1]. An application designed based on micro service architecture is composed of a group of independent, fine-grained and modular services. Each service performs different tasks, and uses light-weight communication mechanisms to transfer information between different services. Every user's request can always find a group of microservices to complete together.

Virtualization technology has become a widely recognized way of server resource sharing. It can provide great flexibility for system administrators in the process of building operating system instances on demand [2]. Because the hypervisor virtualization technology still has some problems in performance and resource utilization efficiency, a new virtualization technology called container has emerged to help solve these problems. Application containerization technology is one of the many technologies to realize microservice architecture. It is a method to realize operating system virtualization. Each container shares the underlying operating system kernel

The associate editor coordinating the review of this manuscript and approving it for publication was Hongwei Du.

and hardware resources, but they can set CPU computing power, memory size, hard disk resources, network bandwidth, IP address and the other aspects independently. At the same time, each container can be run or closed independently without affecting each other. This feature enables containers in the same physical node to realize services with different functions. In the current mainstream container cluster management tool Docker Swarm [3], there are three commonly used scheduling strategies: spread, binpack and random.

PSO and GWO are both excellent swarm intelligence algorithms. GWO algorithm is a meta-heuristic algorithm based on the hunting behavior of the gray wolf group, which achieves the purpose of optimization based on the mechanism of the Wolf group cooperation. GWO algorithm has the characteristics of simple structure, few parameters to adjust, and easy to implement. Among them, there are convergence factors and information feedback mechanism that can perform local search and global search. Therefore, the algorithm has good performance in the accuracy of the solution and early convergence speed of the problem. PSO algorithm is based on the bird group foraging behavior of a meta-heuristic algorithm, its core idea is to use the individual in the group to the optimal information sharing makes the movement of the whole group in the problem solution space evolution process from disorder to orderly, so as to obtain the feasible solution to the problem, PSO algorithm is easy to implement and not too much need to adjust the parameters. These two algorithms have the advantages of less parameters, simple implementation and fast optimization speed, and are also widely used in various fields. However, the microservice container scheduling problem is a multi-objective optimization problem. Its solution space is large and complex. The search ability of a single algorithm is not enough to find a suitable solution in the solution space. And both the algorithms also have the disadvantages of easy to fall into local optimization Liu et al. [4] in this paper pointed out that the GWO is a classic group intelligence algorithm, but it has slow convergence, in some problems easily into the local optimal disadvantages, Yu et al. [5] and Chamaani et al. [6]mentioned PSO algorithm often exist premature convergence problem, easy to fall into the local optimal. So, this paper proposed a new collaborative optimization algorithm PS-GWCA. Through research, this paper finds that although particle swarm optimization and gray wolf algorithm are both vulnerable to local optimization, the reasons and stages of these two algorithms are different. Using the search characteristics of these two algorithms to conduct collaborative optimization can effectively reduce the impact of local optimization on the algorithm and improve the optimization efficiency and stability of the algorithm.

The main contributions of this paper are as follows:

(1) Firstly, this article established three optimization target model, including total network overhead model, load balance model and service reliability model, total network overhead model and service reliability model consists of physical node container two parts inside and outside, load balancing model consists of local load balance and global load balance

(2) Secondly, this paper proposes a container scheduling multi-objective optimization model for optimizing the total network overhead with the computation, memory and storage resource size of physical nodes, with load balancing and service reliability as constraints.

(3) Thirdly, the representation and update methods are improved according to the search characteristics of the PSO. This paper make the PSO applicable to the microservice container scheduling problem, analyze the GWO, and then improve the representation and update methods according to the search characteristics of the GWO. The combination of GWO and pareto optimization theory improves the ability of GWO to solve multi-objective optimization problems and makes GWO suitable for the microservice container scheduling problem combining the advantages of the two algorithms proposes the Particle Swarm - Grey Wolf Cooperation Algorithm based on Microservice Container Scheduling Problem (PS-GWCA) is proposed. In this algorithm, PSO and GWO are used for collaborative optimization through multi-core parallel, and the two algorithms can complement each other in the whole optimization process through information exchange between populations, because is the memory information exchange, populations only exchange Pareto frontier, so the communication overhead is usually not considered, which improves the stability and global optimization ability of the algorithm. In the early of search stage, GWO can use its global optimization ability to guide PSO to jump out of local optimization and avoid premature convergence. In the late of search stage, PSO can enhance the optimization ability of GWO on the frontier of pareto optimization.

(4) Finally, it is difficult to obtain the optimal solution of a multi-objective optimization problem, and this paper proposed PS-GWCA algorithm, which only achieves better results than other algorithms the PS-GWCA algorithm is evaluated through a large number of experiments, and the experimental results are analyzed. The experimental results show that compared with the other three algorithms, the algorithm optimizes the network transmission cost by an average of 18.04%, the local load balancing by an average of 18.07%, the global load balancing by an average of 20.66%, the optimization speed by an average of 7.5%, and 5.69% in service reliability.

This paper is the following work of Multi-Objective and Parallel Particle Swarm Optimization Algorithm for Container-Based Microservice Scheduling on sensors, which mainly proposed the Grey wolf-particle swarm collaboration algorithm PS-GWCA, which can effectively avoid falling into local optima and has better optimization ability. The rest of this paper is organized as follows. The related work is in Section II. The related technologies is in Section III. The definition of microservice container scheduling problem is in Section IV. The proposed PS-GWCA algorithm is in Section V. In Section VI, is the results and analysis of our experiment. Finally in Section VII is the summary and prospect of this paper.

II. RELATED WORK

Currently, the commonly used microservice container scheduling algorithms are mainly divided into two categories: non-heuristic algorithms and heuristic algorithms. The non-heuristic algorithm regards the multi-objective optimization problem of microservice container scheduling as a general task scheduling problem and directly solves it, while the heuristic algorithm usually regards the multi-objective optimization problem of microservice container scheduling as an optimization problem and uses the corresponding heuristic algorithm to solve it [7]. Swarm intelligence algorithm is a kind of heuristic algorithm. At present, it has been widely used to deal with various problems, such as common differential evolution (DF) [8], non-dominated sorting genetic algorithm (NSGA-II) [9], ant colony optimization (ACO) [10]. Because this kind of algorithm can complete the container scheduling problem under complex constraints, it is more favored by researchers. Guerrero et al. [11] proposed a microservice container scheduling algorithm based on genetic algorithm. The algorithm is based on the NSGA-II to find the most suitable container scheduling scheme and solve the elastic scheduling problem when subsequent containers are added, but he did not use the Pareto theory to select the target solution in the Pareto front. Lin et al. [12] proposed a new multi-objective optimization model for microservice container scheduling and used ant colony algorithm to solve this model. However, did not consider the relationship of resource consumption between containers. Bin-Tao et al. [13] introduced the improved bacterial foraging algorithm and designed a new container cloud multi-dimensional resource balancing scheduling method, which improves the cluster resource utilization in the container cloud multi-dimensional resource scheduling process and ensures the scheduling load balance, Wei-Guo et al. [14] improved kubernetes scheduling model by combining ant colony algorithm and PSO. Compared with the original model, this algorithm considers the use of resource cost. Patel et al. [15] proposed a technology based on GWO to distribute the load in the container cloud and reduce the manufacturing time, but using GWO alone tends to fall into local optima.

PSO is one of the most commonly used scheduling algorithms in the field of resource scheduling. Pan and Chen [16] established a resource task allocation model and proposed an improved PSO algorithm to achieve resource load balancing in the cloud environment. Zhang and Yang [17] proposed a task scheduling algorithm based on improved PSO, which can effectively schedule tasks, shorten task completion time and improve resource utilization in cloud computing. Li et al. [18] proposed a docker platform container scheduling algorithm based on PSO to solve the problems of insufficient resource utilization and unbalanced load. The algorithm distributes the application containers on the docker host, balances the use of resources, and finally improves the application performance. Verma and Kaushal [19] proposed a hybrid PSO algorithm based on non-dominated sorting to deal with the multi-objective workflow scheduling problem in the cloud. Li et al. [20] proposed a docker platform container scheduling algorithm based on PSO to solve the problems of insufficient resource utilization and unbalanced load. The algorithm distributes the application containers on the docker host, balances the use of resources, and finally improves the application performance. Jordehi [21]proposed a Since, the basic version of binary particle swarm optimization (PSO) does not perform well in solving binary engineering optimization problems, in this paper a new binary particle swarm optimization with quadratic transfer function, named as quadratic binary PSO (QBPSO) is proposed for scheduling shiftable appliances in smart homes. Chou et al. [22] proposed a dynamic energy-saving resource allocation (DprA) mechanism based on PSO to improve the energy efficiency of cloud data centers, however he did not consider the reliability of the system. Jordehi [23] proposed an improved PSO variant, with enhanced leader, named as enhanced leader PSO (ELPSO) is used. In ELPSO, by enhancing the leader through a five-staged successive mutation strategy, the premature convergence problem is mitigated in a way that more accurate circuit model parameters are achieved in the PV cell/module parameter estimation problem. Liu et al. [24] proposed a container scheduling algorithm based on the PSO algorithm of simulated annealing algorithm. By using the simulated annealing optimized PSO algorithm to make it jump out of the local optimal situation at the initial stage of the algorithm, the algorithm performance is improved. Compared with the standard PSO algorithm and the dynamic inertia weight PSO algorithm, not only the convergence ability is significantly improved, but also the global optimal node hit rate is increased by nearly 60% compared with the standard PSO algorithm. Fan et al. [25] formulated container based microservice scheduling as a multi-objective optimization problem to optimize the network delay between microservices, the reliability of microservice applications and the load balancing of clusters, and proposed a delay, reliability and load balancing aware scheduling (lrlbas) algorithm to determine the container based microservice deployment in edge computing. Jordehi [26]proposed an attempt toward premature convergence mitigation in PSO, its personal acceleration coefficient is decreased during the course of run, while its social acceleration coefficient is increased. In this way, an appropriate tradeoff between explorative and exploitative capabilities of PSO is established during the course of run and premature convergence problem is significantly mitigated. Chen et al. [27] proposed MOPPSO-CMS algorithm, which can find a reasonable microservice container scheduling scheme in a short time. Jordehi [28] proposed a use mixed binary continuous particle swarm optimization (PSO) with V-shaped quadratic transfer for solving UC in demand response integrated MGs, while the uncertainties are considered. Wu and Xia [29] proposed a new container cloud environment cost calculation model in order to solve the container deployment of application tasks in the container

cloud environment at the lowest possible container deployment cost, and then proposed an improved PSO, namely PSO algorithm (cd-PSO), to provide the best solution for application task loading. Ma et al. [30] proposed a comprehensive improved SSA (cissa) based on the newly proposed salt swarm algorithm (SSA).

Although the above researchers have modeled the multi-objective optimization problem of microservice container scheduling and put forward effective solutions, they are still having the shortcoming of easy to fall into local optimization. In this paper, PS-GWCA algorithm is proposed, this paper proposed a algorithm, which cooperates PSO and GWO for optimization. In PS-GWCA the two algorithms can make use of each other's search characteristics in the process of optimization by means of parallel and inter process communication, so as to make up for the shortcoming that they are easy to fall into local optimization in the process of searching, which improve the global search ability and stability of the algorithm.

III. RELATED THEORY

This section will introduce the theories which used in this paper. Pareto optimization theory is one of the most used theories in multi-objective optimization problems. In this paper, it is used to compare the optimization results. In addition, the relevant theories of PSO and GWO will also be explained, this paper combines the advantages of PSO and GWO and conduct parallel computing according to different convergence periods and propose the PS-GWCA algorithm with better performance on multi-target container scheduling problems.

A. PARETO OPTIMALITY THEORY

Pareto optimality is an ideal state of resource allocation. The following section introduces some important concepts behind pareto optimality.

1) Pareto Dominance
For the function $f(x) = [f_1(x), \cdots, f_n(x)]$, if $\forall f_i(x) \leqslant f_i(v) \wedge \exists f_j(x) < f_j(v), i, j \in (1, \cdots, n)$, there is solution $\overrightarrow{x} = (x_1, \cdots, x_m)$ pareto-dominate solution $\overrightarrow{v} = (v_1, \cdots, v_m)$.
2) Pareto Non-inferior Solutions
For the function $f(x) = [f_1(x), \cdots, f_n(x)]$, if $\exists f_i(x) < f_i(v) \wedge \exists f_j(x) > f_j(v), i, j \in (1, \cdots, n)$, there is solution $\overrightarrow{x} = (x_1, \cdots, x_m)$ is pareto non-inferior to solution $\overrightarrow{v} = (v_1, \cdots, v_m)$.
3) Pareto Optimal Solution
For the function $f(x) = [f_1(x), \cdots, f_n(x)]$, if there is no solution in the solution set $X = [\overrightarrow{x}_1, \cdots, \overrightarrow{x}_k]$ that can pareto-dominate solution $\overrightarrow{x}_p, p \in (1, \cdots, k)$, then the solution $\overrightarrow{x}_p$ is the pareto-optimal solution. Multi-objective optimization problems usually have many pareto-optimal solutions and the solution set of pareto-optimal solutions is called the pareto-optimal front.

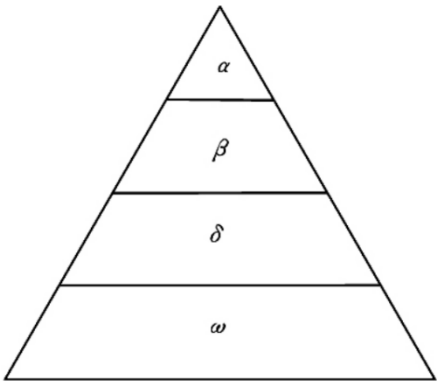


FIGURE 1. Grey wolf's social hierarchy.

B. PARTICLE SWARM OPTIMIZATION

In 1995, Eberhart and Kennedy first proposed the particle swarm optimization algorithm inspired by the predatory behavior of birds [31]. The standard PSO algorithm is detailed in the following equations:

$$V_i(k+1) = \omega \times V_i(k) + c_1 \times rand() \times (pbest_i - X_i) + c_2 \times rand() \times (gbest - X_i), \quad (1)$$

$$X_i(k+1) = X_i(k) + V_i(k+1), \quad (2)$$

where ω is the inertia weight, and c_1 and c_2 are learning weight, representing the influence degree of the external force on particles. The vector $X_i(k) = \{x_{i1}, x_{i2}, \cdots, x_{iN}\}$ represents the position of the kth iteration of particle i in the N-dimensional space. $V_i(k) = \{v_{i1}, v_{i2}, \cdots, v_{iN}\}$ represents the moving distance and direction of the kth iteration of particle i. Each particle is optimized independently in the solution space. In addition to its own inertia, particles also update themselves through two extreme values $pbest_i$ and $gbest$. $pbest_i$ is the historical optimal position in the process of particle iteration. $gbest$ is the historical optimal position found in the process of population optimization. The particles will always update their positions according to these two extreme values, until the optimal solution is found.

C. GREY WOLF OPTIMIZATION

Inspired by the predation behavior of grey wolves, mirjalili proposed a new swarm intelligence optimization algorithm, grey wolf optimization (GWO), in 2014. The algorithm simulates the predation behavior of grey wolf population to achieve the purpose of optimization [32]. In nature, most grey wolves like to live in groups, and they have a very strict hierarchy, as shown in Figure 1.

By modeling the hunting behavior of grey wolves, the following equation is proposed:

$$\overrightarrow{D} = |\overrightarrow{C} \cdot \overrightarrow{X}_p(t) - \overrightarrow{X}(t)| \quad (3)$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X}_p(t) - \overrightarrow{A} \cdot \overrightarrow{D} \quad (4)$$

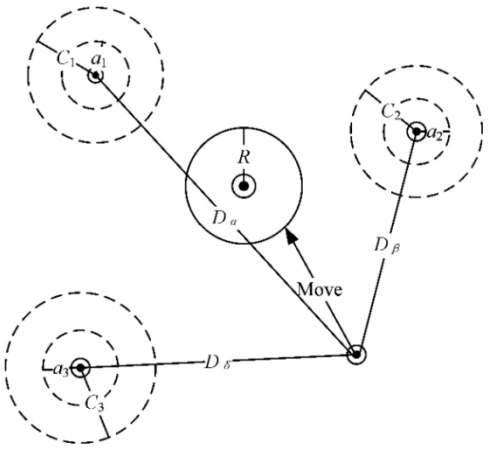


FIGURE 2. Schematic diagram of grey wolf position update in GWO algorithm.

where, t represents the current iteration times, $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, $\overrightarrow{X}_p(t)$ which are the position vectors of prey, and $\overrightarrow{X}$ represents the position vectors of grey wolves. Equation (3) calculates the distance between grey wolf and prey, and equation (4) calculates the position of grey wolf at $t+1$. Where $\overrightarrow{A}$ and $\overrightarrow{C}$ are calculated as follows:

$$\overrightarrow{A} = 2\overrightarrow{a} \cdot \overrightarrow{r_1} - \overrightarrow{a} \tag{5}$$

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r_2} \tag{6}$$

where $\overrightarrow{a}$ is the convergence factor, which decreases linearly from 2 to 0 with the number of iterations, and $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random numbers between [0,1].

Hunting is usually leader by α, β and δ. GWO save the three best solutions obtained so far in α, β and δ, and use these three solutions to guide other wolves to gradually find the best solution. The mechanism by which wolves search for prey is shown in Figure 2.

The position update formula of grey wolf individuals is as follows:

$$\overrightarrow{D}_\alpha = |\overrightarrow{C}_1 \cdot \overrightarrow{X}_\alpha - \overrightarrow{X}| \tag{7}$$

$$\overrightarrow{D}_\beta = |\overrightarrow{C}_2 \cdot \overrightarrow{X}_\beta - \overrightarrow{X}| \tag{8}$$

$$\overrightarrow{D}_\delta = |\overrightarrow{C}_3 \cdot \overrightarrow{X}_\delta - \overrightarrow{X}| \tag{9}$$

$$\overrightarrow{X}_1 = \overrightarrow{X}_\alpha - \overrightarrow{A}_1 \cdot \overrightarrow{D}_\alpha \tag{10}$$

$$\overrightarrow{X}_2 = \overrightarrow{X}_\beta - \overrightarrow{A}_2 \cdot \overrightarrow{D}_\beta \tag{11}$$

$$\overrightarrow{X}_3 = \overrightarrow{X}_\delta - \overrightarrow{A}_3 \cdot \overrightarrow{D}_\delta \tag{12}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X}_1 + \overrightarrow{X}_2 = \overrightarrow{X}_3}{3} \tag{13}$$

where $\overrightarrow{D}_\alpha$, $\overrightarrow{D}_\beta$ and $\overrightarrow{D}_\delta$ respectively represent the distance between α, β and δ and other individuals; $\overrightarrow{X}_\alpha$, $\overrightarrow{X}_\beta$ and $\overrightarrow{X}_\delta$ respectively represent the current position of α, β, δ; $\overrightarrow{C}_1$, $\overrightarrow{C}_2$ and $\overrightarrow{C}_3$ are random vectors. Equations (10) - (12) define the orientation of individuals in wolves α, β, δ For the step length and direction of advance, equation (14) determines the final position of grey wolf individuals.

IV. PROBLEM STATEMENT

A. RELATED PARAMETERS

An application based on a microservice architecture can be represented as a tuple $< MS\_SET, MS\_RELATION >$ [27], where MS_SET is the set of microservices related to the application and $MS\_RELATION$ is the set of utilization relationships among the microservices of the application. If a microservice completes a task and needs to use the result of other microservices, there is a utilization relationship between the two microservices. This relationship can be defined as $(ms_{cons}, ms_{prov}) \in MS\_RELATION$, where ms_{cons} represents the consumer and ms_{prov} represents the provider.

A microservice ms_i can be represented as a tuple $< Calc\_Req_i, Str\_Req_i, Mem\_Req_i, Fail_i, CONS\_SET_i, Link\_Thr_i, Scale_i >$ where $Calc\_Req_i, Str\_Req_i, Mem\_Req_i$ are the computing resources, storage resources, memory resources are the resource that a container instance of microservice ms_i required, when needs to process a unit of user requests. $Fail_i$ is the probability that microservice ms_i will fail when processing a request. $CONS\_SET_i$ is a set of predecessors microservice of microservice ms_i, the implementation of microservice ms_i requires the results of these microservices; $Link\_Thr_i$ is the maximum number of the request that can be processed by a single container instance of microservice ms_i. $Scale_i$ is the number of container instances of microservice ms_i. A microservice can have multiple container instances, but a container instance can only serve one microservice.

As mentioned above, there are utilization relationships between microservices and microservices, and their container instances. $Link(ms_i, ms_l)$ is the number of requests between microservice ms_i and ms_l. $Trans(ms_i, ms_l)$ is the amount of data required for a request between microservice ms_i and microservice ms_l. When the sender is a user or a client, the network transmission cost is ignored, only consider the number of requests in this paper.

A physical node, pm_j, can be represented as a tuple $< Calc\_Res_j, Str\_Res_j, Mem\_Res_j, Fail_j >$. Microservices are not mutually exclusive and there can be multiple container instances of microservices on a physical node. $Calc\_Res_j$, Str_Res_j, Mem_Res_j are the computing resources, storage resources, memory resources that physical node pm_j can provide respectively. The resources consumed by the container within the physical node must be less than the resources available on the physical node. $Fail_j$ represents the physical node failure rate, for physical nodes may cause downtime, denial of service, or computational exceptions due to software or hardware problems; as such. Physical nodes are connected by the network, and $Dist(pm_j, pm_{j'})$ is the network distance between each physical node. The time required for data transmission between two containers is represented as $PassTime(cont_i, cont_k)$. The closer the containers are, the shorter $Dist(pm_j, pm_{j'})$ and $PassTime(cont_i, cont_k)$ are. The relevant parameters and explanations of the

TABLE 1. Parameters relevant to the models discussed in this paper.

Element	Parameter	Description
Application	MS_SET	Microservice set of an application
	$\|MS\_SET\| = m$	Total number of microservices
	$MS\_RELATION$	utilization relationships between microservices
Microservice	$ms_i \in MS\_SET$	Microservice i
	$Calc\_Req_i$	Computing resources required by a container of microservice i
	Str_Req_i	Storage resources required for a container of microservice i
	Mem_Req_i	Memory resources required for a container of microservice i
	$Fail_i$	Failure rate of container
	$CONS\_SET_i$	Microservice set consumed by microservice i
	$Link\_Thr_i$	The request threshold that can be processed by a container of microservice i
	$Link_i$	Total requests received for microservice i
	$Scale_i$	The number of containers of microservice i in the cluster
	$(ms_i, ms_l) \in MS\_RELATION$	utilization relationships between microservice i and microservice l
	$Link(ms_i, ms_l)$	Total requests from microservice i to microservice l
	$Trans(ms_i, ms_l)$	Data transmission from microservice i to microservice l
Physical Node	$pm_j \in CLUSTER$	Physical node j
	$\|CLUSTER\| = n$	Total number of physical nodes
	$Calc\_Res_j$	Computing resources of physical nodes
	Str_Res_j	Storage resources of physical nodes
	Mem_Res_j	Memory resources of physical nodes
	$Fail_j$	Failure rate of physical nodes
Network	$Dist(pm_j, pm_{j'})$	Network distance between physical nodes
	$PassTime(pm_j, pm_{j'})$	Time required to transfer data between physical nodes

multi-objective optimization problem of microservice container scheduling are shown in Table 1.

B. NETWORK TRANSMISSION

Containers deployed in the same physical node have significantly faster transmission speed, shorter transmission distance and more reliable service requests than containers deployed in different physical nodes. The network transmission cost model used in this paper is as follows:

$$\begin{aligned} &Trans\_Consume(type) \\ &\quad = Link(cont_i, cont_k^{type}) Trans(cont_i, cont_k^{type}) \\ &\quad Dist(cont_i, cont_k^{type}) PassTime(cont_i, cont_k^{type}) \end{aligned} \tag{14}$$

$$Inner\_Consume = \frac{\sum_{i=1}^{n} \sum_{k \in CONS\_SET} Trans\_Consume(in)}{Scale_i} \tag{15}$$

$$Outer\_Consume = \frac{\sum_{i=1}^{n} \sum_{k \in CONS\_SET} Trans\_Consume(out)}{Scale_i} \tag{16}$$

$$Total\_Consume = Inner\_Consume + Outer\_Consume \tag{17}$$

The network transmission cost $Total\_Consume$, consists of $Inner\_Consume$ and $Outer\_Consume$. $Inner\_Consume$ is the network transmission cost between containers which assigned to the same physical node. $Outer\_Consume$ is the network transmission cost between containers which assigned to different physical nodes. $Trans\_Consume(type)$ is the calculation method of network transmission cost. Depending on whether the containers are assigned to the same physical node, the $type$ is divided into in and out. If the containers are assigned to the same physical node, the container, which is related to $cont_i$, is to represent as $cont_k^{in}$. If the container is assigned to different physical node, the container, which is related to $cont_i$, is represented as $cont_k^{out}$. $cont_i$ is the container instance of microservice ms_i. Container instances of microservices that have consumer relationships with microservice ms_i and are assigned to different physical nodes, represented as $cont_k^{out}$. This model focuses on the difference between the containers which are assigned to the same physical node and the containers which are assigned to different physical nodes.

C. LOAD BALANCING

Cluster load balancing and local load balancing make up global load balancing. Cluster load balancing is the load balancing of the physical node cluster. Local load balancing refers to the balanced use of resources within physical node. The purpose of global load balancing is to realize the load balancing of the entire physical node cluster and realize the rational utilization of the resources of the physical node. Where CB represents cluster load balancing, LLB represents local load balancing, and GLB represents global load. The model of load balancing is used as follows (18)–(23), shown at the bottom of the next page, where LLB records the differences between the three resources in the physical node. $CalcStrDif$ is the difference between calculting resource and storage resource. $StrMemDif$ is the difference between storage resource and memory resource. $MemCalcDif$ is the difference between memory resource and calculating resource. The larger the value is, the more unbalanced the resource usage of the physical node is. σ_{calc} represents the standard deviation of computing resources, σ_{str} represents the standard deviation of storage resources, σ_{mem} represent the standard deviation of memory resources. CB is the mean of three standard deviations. The larger the value is, the worse the load balance of the physical cluster is. GB is the mean of CB and LLB.

D. SERVICE RELIABILITY

The service reliability model used in this paper is as follows:

$$InnerFail = \frac{Link(cont_i, cont_k^{in})}{Scale_k} (fail_i + fail_k) \tag{24}$$

$$\begin{aligned} OuterFail =& \frac{Link(cont_i, cont_k^{out})}{Scale_k} \\ &\times [fail_j + (1 - fail_j)(fail_i + fail_k)] \end{aligned} \tag{25}$$

$$SystemFail = InnerFail + OuterFail \tag{26}$$

Similar to global load balancing, service reliability is divided into two cases *InnerFail* and *OuterFail*. *InnerFail* is the number of fails of the requests between container which are assigned to the same physical node, and *OuterFail* is the number of fails of the requests between container which are assigned to the different physical nodes. There is the difference between *InnerFail* and *OuterFail*. When requests are transmitted in same physical node, the probability of fail is only considering the container failure rate, the $fail_i$ and $fail_k$ in (24) and (25); But when requests are transmitted between different physical node, the probability of fail is consisted of the container failure rate and the physical failure rate, the $fail_j$ in (25).

E. MULTI-OBJECTIVE OPTIMIZATION MODEL

In order to solve the container-based microservice scheduling problem, a multi-objective optimization model based on the above three models was established under the constraints of physical node resources and the number of containers and we can know that the microservice container scheduling problem can be transformed into the solution to minimal problem.

$$minimize\ Total\_Consume(x) \tag{27}$$
$$minimize\ LoadBalancing(x) \tag{28}$$
$$minimize\ SystemFail(x) \tag{29}$$
$$s.t.\ Calc\_Req_j \le Calc\_Res_j \tag{30}$$
$$s.t.\ Str\_Req_j \le Str\_Res_j \tag{31}$$
$$s.t.\ Mem\_Req_j \le Mem\_Res_j \tag{32}$$

Function (27) – (29) represents three optimization objectives: minimizing network transmission costs, rationalizing load balancing, and minimizing the number of failed requests. The function (30) – (32) represents the computing, storage, and memory resource constraints of the physical node, respectively. The resources used by containers on physical nodes cannot exceed the resources available on physical nodes.

V. PS-GWCA FOR MICROSERVICE CONTAINER SCHEDULING PROBLEM

This paper has carried out experiments to illustrate the advantages and disadvantages of PSO and GWO during the search stage in the microservice container scheduling problem. The experiment environment and data are the same as those in Section VI. The number of *UserRequest* is 5, the *iternum* and the size of population are both 300. It can be seen from Figure 3 and Figure 4 that in the early stage of search, the PSO is easy to be limited to the local optimum, and its ability to jump out of the local optimum is weak, resulting in premature convergence. The GWO has a strong global optimization ability in the early stage of search because of its first divergence. With the help of this ability, the GWO has a smoother convergence curve and better results. However, it can be seen from Figure 5 and Figure 6 that in the later stage of the search, the global search ability of the GWO decreases with the increase of the number of iterations. Because the individuals of the GWO on the pareto optimal frontier attract each other, the local optimization range of the algorithm on the pareto optimal frontier gradually decreases, and finally converges to a certain point, Because of its search characteristics, PSO has a stronger ability of local optimization on the pareto optimal frontier. Therefore, according to this characteristic, this paper uses PSO and GWO to solve the multi-objective optimization problem of microservice container scheduling. Because the PSO and GWO are not directly applicable to the microservice container scheduling problem in this paper, it is necessary to improve the PSO and GWO for the microservice container scheduling problem.

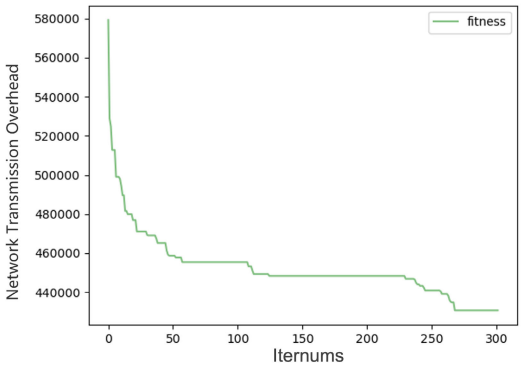


FIGURE 3. Convergence curve of PSO.

$$CalcStrDif_j = |\frac{Calc\_Req_j}{Calc\_Res_j} - \frac{Str\_Req_j}{Str\_Res_j}| \tag{18}$$
$$StrMemDif_j = |\frac{Str\_Req_j}{Str\_Res_j} - \frac{Mem\_Req_j}{Mem\_Res_j}| \tag{19}$$
$$MemCalcDif_j = |\frac{Mem\_Req_j}{Mem\_Res_j} - \frac{Calc\_Req_j}{Calc\_Res_j}| \tag{20}$$
$$LLB = \frac{\sum_{j=1}^{n}(CalcStrDif_j + StrMemDif_j + MemCalcDif_j)}{3n} \tag{21}$$
$$CB = \frac{\sigma_{clac} + \sigma_{str} + \sigma_{mem}}{3} \tag{22}$$
$$GLB = \frac{ClusterLoadBalancing + LocalLoadBalancing}{2} \tag{23}$$

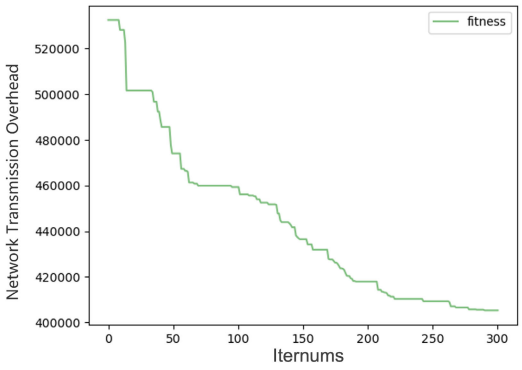


FIGURE 4. Convergence curve of GWO.

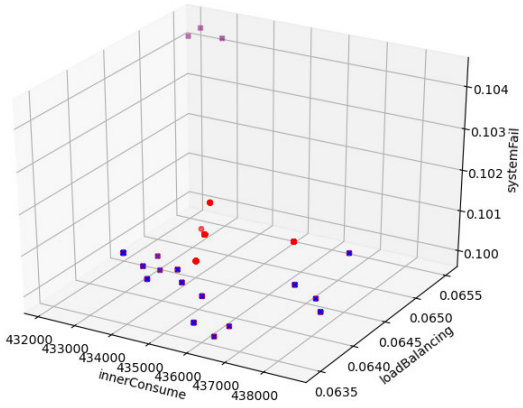


FIGURE 5. Pareto front of PSO.

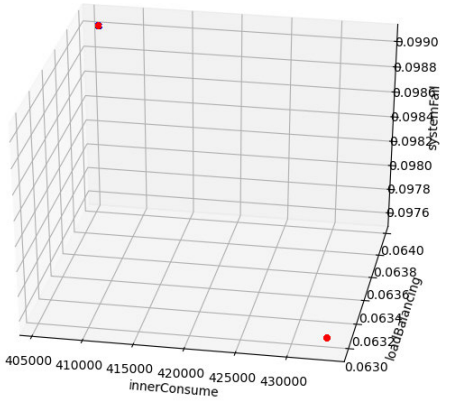


FIGURE 6. Pareto front of GWO.

A. REPRESENTATION OF INDIVIDUALS

The number-based scheduling scheme representation is used in this paper. A two-dimensional array is used to represent scheduling scheme, each row is a microservice ms_i in the application, and each is a physical node pm_j that can be assigned. The element (ms_i, pm_j) is the number of container instances of microservice ms_i that have been allocated to physical node pm_j. Figure 7 shows the example of the individual, which is randomly initialized by the PS-GWCA algorithm.

Physical Node pm_1				
$cont_1$ ms_1	$cont_1$ ms_2	$cont_1$ ms_3		
Physical Node pm_2				
$cont_2$ ms_2	$cont_3$ ms_2	$cont_2$ ms_3	$cont_1$ ms_5	$cont_2$ ms_5
Physical Node pm_3				
$cont_2$ ms_1	$cont_3$ ms_3	$cont_1$ ms_4		

(a) Initialize container allocation

	pm_1	pm_2	pm_3
ms_1	1	0	1
ms_2	1	2	0
ms_3	1	1	1
ms_4	0	0	1
ms_5	0	2	0

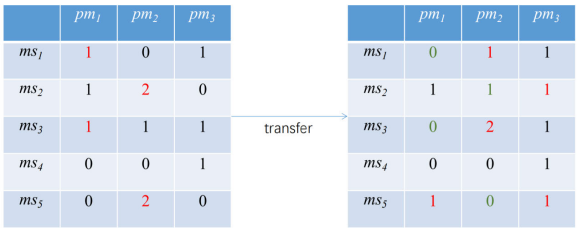
(b) Initialize microservice

FIGURE 7. Representation based on number of containers.

B. UPDATE METHOD

In PSO and GWO, there are two factors that affect the movement direction of individuals. The first is the individual optimization ability of the population, which ensures the diversity of the population and the global optimization ability of the algorithm in the solution space. If there is no influence of this factor, the population will fall into blind obedience, leading to premature algorithm and extremely easy to fall into local optimal. This factor corresponds to the influence of inertia on particle movement direction in PSO, and the roaming ability of wolves in the solution space in the early searching period in GWO. The second is the learning ability of individuals, which enables individuals to make use of the experience of the whole population to gradually approach the optimal solution until convergence. If there is no influence of this factor, the individuals in the population of the algorithm will be randomly scattered in the solution space, lose the ability of local optimization and will not converge. This factor corresponds to the ability to learn from individual and global extreme values in PSO, and the ability to approach and learn from three wolves in GWO. In PSO algorithm and the GWO individuals all can learn from the best individual within a population, but the service container scheduling problem is a multi-objective optimization problem, it is difficult to find the optimal solution, in order to solve this problem, the PSO the optimal solution *gbest* and α, β, and δ of GWO will be replace by pareto optimal front.

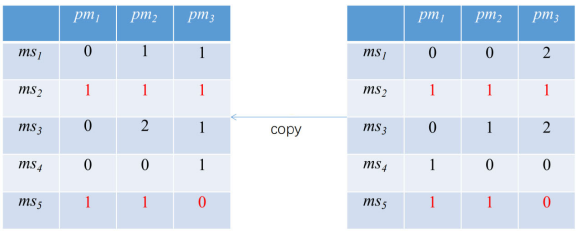
In order to ensure the self-optimization ability of individuals, this paper changes the position of individuals through transfer operation during algorithm optimization. Taking a particle as an example the particle transfer operation is shown in Figure 8.



	pm_1	pm_2	pm_3
ms_1	1	0	1
ms_2	1	2	0
ms_3	1	1	1
ms_4	0	0	1
ms_5	0	2	0

	pm_1	pm_2	pm_3
ms_1	0	1	1
ms_2	1	1	1
ms_3	0	2	1
ms_4	0	0	1
ms_5	1	0	1

FIGURE 8. Transfer operation.



	pm_1	pm_2	pm_3
ms_1	0	1	1
ms_2	1	1	1
ms_3	0	2	1
ms_4	0	0	1
ms_5	1	1	0

	pm_1	pm_2	pm_3
ms_1	0	0	2
ms_2	1	1	1
ms_3	0	1	2
ms_4	1	0	0
ms_5	1	1	0

FIGURE 9. Copy operation.

In Figure 8, the left side represents the particle state before the transfer operation, and the right side represents the particle state after the transfer operation. If transfer operation occurs at a location, any number of containers at that location will be randomly moved to other physical nodes. For example, a transfer occurs at (ms_1, pm_1), where the container instance is randomly transferred to the physical node pm_2, and at (ms_2, pm_2), where a container instance is transferred to the physical node pm_3, and at (ms_5, pm_2), the container instance distribution is transferred to physical node pm_1 and physical node pm_3.

In order to ensure the learning ability of individuals, the position of individuals is changed by copy operation during algorithm optimization. In the copy operation, the particle could copy the individual and global extremes according to the learning factors c_1 and c_2. Taking the above particle and its individual extreme value as an example, the copy operation of the particle is shown in Figure 9.

In Figure 9, the left side is the individual particle after the copy operation, and the right side is the individual extreme value of the particle. As can be seen from Figure 9, the transfer operation takes place at row ms_2 and row ms_5, which copies and overwrite the existing deployment of the same row within the individual extreme value, thus achieving the purpose of learning from the individual extreme value.

In GWO according to the factor $\vec{a}$ the wolves would spread in the solution space firstly, and the gradually converges the head wolf. In order to make PSO suitable for solving microservice container schedule problem, in this paper the factor $\vec{a}$ represents the rate of the number of rows performing transfer operations in the total number of rows. As iterations increase, the $\vec{a}$ will gradually decrease to 0. And $1 - \vec{a}$ represents the rate of the number of rows performing copy operations on pareto optimal front in the total number of rows.

According to the characteristics of PSO, the update of particle is affected by its own inertia, individual extreme value and global extreme value at the same time. ω, c_1 and c_2 respectively indicate the influence degree of these three factors on particle. In order to make PSO suitable for solving microservice container schedule problem, in this paper the sum of ω, c_1 and c_2 is 1. The position of particles update process in PSO is as follows:

	pm_1	pm_2	pm_3
ms_1	1	0	1
ms_2	1	2	0
ms_3	1	1	1
ms_4	0	0	1
ms_5	0	2	0

FIGURE 10. Particle after updating.

Firstly, ω times the number of microservices to determine the number of rows to perform the transfer operation, then randomly select the rows to do transfer operation. Secondly, use c_1 times number of microservices to determine the number of rows that will perform copy operation on *pbest*, the same as above, randomly select the remaining rows to do copy operation on *pbest*. Finally, the rest rows will perform copy operation on *gbest*. The effect of self-inertia(pink), individual extreme value(orange) and global extreme value(green) are shown in Figure 10.

C. METHOD OF COOPERATION BETWEEN POPULATIONS

In PS-GWCA, multi-algorithm and multi-population parallelism are realized through multi-core parallelism, and information exchange between different populations is realized through inter-process communication. At the end of each iteration, PSO and GWO will upload the best individuals within populations, namely the pareto optimal front of two algorithms, to the shared memory. Then the third party known as the "Factory" will download the two the pareto optimal front from shared memory, merge the two fronts, and do a pareto optimal sorting again. The elements within the two pareto optimal fronts will be compared to each other. If an element is dominated by another element pareto, the element will be removed and the remaining elements will form a new pareto optimal frontier, which will be re-uploaded to shared memory.

PSO and GWO will download the newly generated pareto optimal front from shared memory and replace the local pareto optimal front. At the next location update, PSO and GWO will be guided by the new pareto optimal front for optimization. At the same time, the PSO and GWO will find the worst individuals of population and replace it with in the pareto optimal front from the elements in the population. Because of the large search space, appropriately with the best individual to replace the worst individual can effectively improve the efficiency of convergence and optimization, but too much will lead to substitute algorithm in early fall into local optimum.

D. METHODS OF COMPETITION BETWEEN POPULATIONS

In PS-GWCA, in addition to the cooperative relationship mentioned above, competition also exists between populations. In the "Factory" there is a counter *count*. A counter records the number of times the algorithm performs badly. If this counter is above a certain threshold or below a certain threshold, said an algorithm optimization effect is too poor, have not conducive to problem solving, such as into a local optimum, and can't jump out from it, or the algorithm itself optimization ability weak does not apply to solve the problem, and so on. The Factory will then issue a command, and the algorithm receiving the command will be transformed. The better half individuals of the population will retain existing fitness value and the information such as scheduling scheme, the other poor individuals will be initialized again. After the transformation the update method of algorithm will become the same as the better algorithm. For example, if PSO has poor performance, it will be transformed to GWO. The fitness value and position of the better half individuals of the PSO will be saved. The system will initialize a new GWO, the saved individuals from PSO will be assigned to the new GWO and continue to optimization.

In Factory, $count = 0$ in its initial state. The Factory evaluates the remaining elements in the new pareto optimal frontier. The counter is increased by one if the worst-performing element comes from PSO, and decreased by one if the element comes from GWO. Transformation occurs when *count* is greater than or less than a certain threshold. According to the experiment in this paper, when the threshold value is too small, the transformation will take place prematurely, which is very unfavorable to some algorithms with strong optimization ability in the later period. If the threshold value is too large, the transformation will not take place, and algorithms with poor optimization ability will run until the end of iteration, which not only wastes computing resources, but may also have adverse effects on optimization. After experiments, it is suggested in this paper that the threshold value of $0.5\times$ iteration number is suitable. The transformation occurs when $count \geq +0.5\times$ iteration number, and the PSO transform to GWO and when $count \leq -0.5\times$ iteration number, GWO transform to PSO.

E. PARTICLE SWARM - GREY WOLF COOPERATION ALGORITHM

The framework of PS-GWCA proposed in this paper is shown in Figure 11. In the calculation process, PSO, GWO and Factory are simultaneously run in a parallel way with multiple processes. The calculation process of PS-GWCA can be summarized as follows:

Step1: Initialize PSO, GWO and Factory

Step2: Calculate the fitness of each particle and wolf and select the pareto optimal front in PSO and GWO respectively. And upload it to the shared memory.

Step3: The Factory downloads the pareto optimal front of PSO and GWO from shared memory. The Factory removes the used new pareto optimal frontier from shared memory.

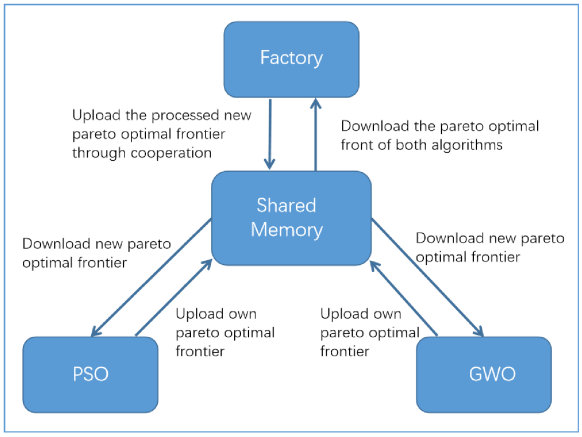


FIGURE 11. Framework of PS-GWCA.

Step4: The Factory merges the two pareto optimal fronts and reorder them to form a new pareto optimal front and upload them to the shared memory again.

Step5: The Factory evaluates the individuals in the new pareto optimal frontier and operates counters according to the population of the worst-performing individuals.

Step6: PSO and GWO download the new pareto optimal frontier from the shared memory, mark it with use, and upload it again.

Step7: PSO and GWO perform transfer operation according to probability of conducting autonomous optimization.

Step8: PSO and GWO take the new pareto optimal front as the *gbest* and pareto optimal front of the population respectively, and perform the copy operation on the new pareto optimal front according to the probability to guide the population optimization.

Step9: if the number of iterations reaches, output pareto optimum; otherwise, goes to Step2.

The pseudocodes of PS-GWCA are shown in Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4.

VI. EXPERIMENT AND ANALYSIS

A. EXPERIMENTAL DATA

In order to verify the effectiveness of PS-GWCA in solving the microservice container scheduling problem, this paper conducts experiments based on the Alibaba cluster trace V2018 cluster data set [33]. This data set contains about 4000 computers, including 8-day information, and consists of the following six tables:

1) machine_meta.csv: the meta info and event information of machines.
2) machine_usage.csv: the resource usage of each machine.
3) container_meta.csv: the meta info and event information of containers.
4) container_usage.csv: the resource usage of each container.
5) batch_instance.csv: information about instances in the batch workloads.

Algorithm 1 PS-GWCA (Main)

Input: Data set and related parameters
Output: Optimal solution (scheduling scheme)
Initialize the number of iterations
Initialize the number of user requests
Initialize container and physical node parameters
Initialize the container relationship set
Parallel PSO Algorithm, GWO Algorithm and ‘‘Factory"
Collect results
Output the pareto optimal frontier

Algorithm 2 PS-GWCA (PSO Part)

Input: Data set and related parameters
Output: Optimal solution (scheduling scheme)
Initialize the particle swarm
Calculate the fitness of particles
Initialize *pbest* and pareto optimal front
while i<iternum **do**
  Update own pareto optimal front to shared memory
  flag = 1
  while flag **do**
    Get pack from shared memory
    if pack comes from Factory **then**
      Replaces the own pareto optimality in the package with the new pareto optimality
      Update the pack to shared memory
    end if
    if *transform_flag* == 1 **then**
      PSO transform to GWO
    end if
    flag = 0
  end while
  Replace the individual with the worst fitness in the particle swarm with an element in the pareto front
  for each particle **do**
    Perform transfer operation
    Perform copy operation
  end for
  Calculate the particles' fitness
  Update the *pbest* and pareto optimal front
  i += 1
end while
Output pareto optimal frontier

Algorithm 3 PS-GWCA (GWO Part)

Input: Data set and related parameters
Output: Optimal solution (scheduling scheme)
Initialize the wolves
Calculate the fitness of wolves
Initialize pareto optimal front
while i<iternum **do**
  Update own pareto optimal front to shared memory
  flag = 1
  while flag **do**
    Get pack from shared memory
    if pack comes from Factory **then**
      Replaces the own pareto optimality in the package with the new pareto optimality
      Update the pack to shared memory
    end if
    if *transform_flag* == 2 **then**
      GWO transform to PSO
    end if
    flag = 0
  end while
  Replace the individual with the worst fitness in the particle swarm with an element in the pareto front
  for each wolf **do**
    c_2=i/*iternum*
    $c_1 = 1 - c_2$
    if random() < c_1 **then**
      Perform transfer operation to pareto optimal front
    end if
    if random() < c_2 **then**
      perform copy operation to
    end if
  end for
  Calculate the particles' fitness
  Update the *pbest* and pareto optimal front
  i += 1
end while
Output pareto optimal frontier

6) batch_task.csv: information about instances in the batch workloads. Note that the DAG information of each job's tasks is described in the task_name field.

Experimental data sets are shown in Table 2 and Table 3. Table 2 shows the utilization relationships between microservices, the data transferred and the number of requests each time the application receives a unit of user request. If a service is an entry microservice, that is, the first microservice started in the entire application, then the utilization relationships of the microservice is $(0, ms_i)$ in Table 2. Table 3 shows the related resources and failure rate required by microservices to complete a unit of user request.

B. EXPERIMENTAL ENVIRONMENT AND PARAMETER SETTING

In this paper, the experiment is carried out on a Windows11 system, the processor is Intel core i7-8750h, the memory is 16GB, the display card is NVIDIA RTX2070, the programming language is Python. In order to verify the effectiveness of PS-GWCA algorithm, this paper compared it with MOPPSO-CMS, GA-MOCA and Spread algorithm.

MOPPSO-CMS [27] algorithm is a multi-objective optimization algorithm based on parallel PSO to solve the scheduling problem of microservice containers. The parameter configuration of the algorithm is the same as the PSO

Algorithm 4 PS-GWCA (Factory Part)

Input: Pareto optimal front from GWO and PSO
Output: New pareto optimal front after processing
count = 0
transform_flag = 0
while i<iternum **do**
  flag = 1
  while flag **do**
    Get pareto optimal front form GWO and PSO
    Reorder the two pareto optimal frontiers to generate a new pareto optimal front
    flag = 0
  end while
  if *transform_flag*==0 **then**
    Evaluate algorithm efficiency
    if PSO is worse **then**
      count+ = 1
    end if
    if GWO is worse **then**
      count− = 1
    end if
  end if
  if *count* > *iternum*/2 **then**
    transform_flag = 1
  end if
  if *count* < −*iternum*/2 **then**
    transform_flag = 2
  end if
  Update the new pareto optimal front to shared memory
  i += 1
end while

parameter of the PS-GWCA algorithm proposed in this paper.

GA-MOCA [11] algorithm is a multi-objective optimization algorithm for micro-service container scheduling based on NSGA-II. The algorithm considers the threshold distance of the container, the load balance of the physical node computing resources, the reliability of the service and the communication cost between related microservices. Through experimental analysis, we observed that the number of populations in GA-MOCA is 300, and the number of iterations is 300 were suitable for the experimental evaluation of our solution. The crossover probability is fixed to 1.0, because the design of NSGA-II already considers the possibility of keeping fathers from one generation in the offspring. The mutation probability is established as 0.25, and the selection of the mutation is carried out uniformly.

Spread [3] is one of Docker Swarm's scheduling strategy. It selects physical nodes with the least container instances for deployment in each iteration, so that cluster resources can be used more evenly to reduce the load balancing of the algorithm.

TABLE 2. A unit of user requests the utilization relationships between microservices.

(ms_i, ms_j)	$Link_{i,j}$	$Trans_{i,j}$
$(0, ms_1)$	50	0
$(0, ms_3)$	70	0
$(0, ms_6)$	8	0
$(0, ms_7)$	30	0
$(0, ms_{10})$	100	0
$(0, ms_{13})$	30	0
(ms_1, ms_2)	20	4.6
(ms_1, ms_4)	10	3.1
(ms_1, ms_9)	20	4.0
(ms_2, ms_4)	10	3.5
(ms_2, ms_{12})	15	5.9
(ms_3, ms_{13})	60	1.8
(ms_4, ms_{15})	30	5.6
(ms_4, ms_{16})	8	5.7
(ms_5, ms_{15})	30	5.3
(ms_7, ms_2)	20	4.8
(ms_7, ms_{14})	10	4.1
(ms_8, ms_{14})	15	4.2
(ms_9, ms_5)	20	3.6
(ms_9, ms_{11})	20	4.7
(ms_{10}, ms_5)	20	3.4
(ms_{10}, ms_9)	25	4.4
(ms_{10}, ms_{11})	20	4.9
(ms_{11}, ms_2)	20	3.2
(ms_{12}, ms_8)	45	6.4
(ms_{13}, ms_2)	20	4.5
(ms_{13}, ms_8)	45	6.1
(ms_{13}, ms_{16})	8	5.5
(ms_{13}, ms_{17})	30	2.4
(ms_{15}, ms_{16})	8	5.2
(ms_{16}, ms_{14})	15	4.3
(ms_{17}, ms_{12})	15	6.2

TABLE 3. The resources required to complete a unit of user requests.

ms_i	CONS	Cal_i	Str_i	Mem_i	$Scale_i$	$Link_i$	Thr_i	$Fali_i$
1	2, 4, 9	2.1	1.4	2	5	10	50	0.04
2	4, 12	0.5	3.2	4	10	8	80	0.02
3	13	3.1	1.6	2	9	8	70	0.02
4	15, 16	4.7	0.2	2	4	5	20	0.02
5	15	1.8	3.1	2	10	8	40	0.002
6		2.5	5.1	2	2	4	8	0.02
7	2, 14	6.2	0.6	2	8	4	30	0.001
8	14	0.8	6.2	2	23	4	90	0.003
9	5, 11	3.9	2.3	2	9	5	45	0.001
10	5, 9, 11	0.2	4.8	4	25	4	100	0.006
11	2	2.8	2.6	2	5	8	40	0.02
12	8	5.3	0.9	2	8	4	30	0.003
13	2, 8, 16, 17	0.6	4.8	4	18	5	90	0.04
14		6.1	2.5	2	10	4	40	0.006
15	16	1.2	4.2	4	12	5	60	0.003
16	14	5.4	1.6	2	6	4	24	0.02
17	12	3.7	2.2	4	5	6	30	0.004

The specific parameters in the algorithm are set as follows:
(1) Physical node number: $|CLUSTER| = 100$.
(2) Physical nodes have three different computing capabilities: $Calc\_Resj = [100, 200, 400]$.
(3) Physical nodes have three different storage capacities: $Str\_Resj = [100, 200, 400]$.
(4) Physical nodes have three different memory capacities: $Mem\_Resj = [60, 120, 180]$.

TABLE 4. Performance comparison in terms of network transport cost.

UserRequest	PS-GWCA	MOPPSO	GA-MOCA	Spread
1	**22637**	25455	30764	36375
1.5	**49039**	55986	63660	77758
2	**87266**	96177	112181	127170
2.5	**126090**	141992	149693	179951
3	**180746**	197190	201388	249179
3.5	**228936**	250486	273723	319389
4	**285978**	315070	336937	371709
4.5	**348715**	380004	405690	440373
5	**402837**	442095	457704	520737

(5) The failure rate of physical nodes is a random number ranging from 0.001 to 0.01.

(6) Network distance between container transmission: $Dist(pm_j, pm_{j'}) = 1$ and $Dist(pm_j, pm_{j'}) = 4$ of different physical nodes.

(7) The time required for data transmission between containers is the same as $PassTime = 1$ in the same physical node; Different physical nodes $PassTime = 4$.

Related parameters of PS-GWCA algorithm proposed in this paper are set as follows: the number of iterations is 300, the number of population in PSO is 300, the inertia of PSO is $\omega = 0.45$, the individual optimal learning factor $c_1 = 0.1$, and the global optimal learning factor $c_2 = 0.45$. The population of GWO is 300, and the convergence factor $\vec{a}$ increases linearly from 0.2 to 0.8 with the number of iterations. The Factory counter $count = 0$ and the transformation occurs when $count > 150$ or $count < -150$.

C. EXPERIMENT RESULTS AND ANALYSES

The comparison of the average results obtained by the PS-GWCA algorithm in this paper and the other three algorithms after repeated operations under different units of user requests is shown in Table 4, 5, 6, 7 and 8 respectively, in the table MOPPSO-CMS will be abbreviated to MOPPSO. The better results in the table are shown in bold. According to the operation results of the four algorithms, the box plots obtained under the user request of one unit are shown in Figure 12, 13, 14 and 15 respectively.

As can be seen from Table 4, the PS-GWCA algorithm proposed in this paper has an average optimization of 9.69% compared with MOPPSO-CMS, 15.80% compared with GA-MOCA and 28.70% compared with Spread in terms of network transmission cost in terms of the optimal value of the four algorithms.

As can be seen from Table 5, the PS-GWCA algorithm proposed in this paper has an average optimization of 4.19% compared with MOPPSO-CMS, 18.22% compared with GA-MOCA and 21.61% compared with Spread in terms of global load balancing overhead in terms of the optimal value of the four algorithms.

As can be seen from Table 6, in terms of the optimal value of the four algorithms, the PS-GWCA algorithm proposed in this paper has an average optimization of 6.91% compared with MOPPSO-CMS, 24.22% compared with GA-MOCA and 30.86% compared with Spread in terms of local load balancing.

TABLE 5. Performance comparison in terms of global load balancing.

UserRequest	PS-GWCA	MOPPSO	GA-MOCA	Spread
1	**0.014**	0.015	0.020	0.021
1.5	**0.021**	0.022	0.027	0.028
2	**0.026**	0.028	0.035	0.035
2.5	**0.033**	0.034	0.040	0.042
3	**0.039**	0.041	0.045	0.049
3.5	**0.045**	0.047	0.053	0.055
4	**0.052**	0.053	0.059	0.062
4.5	**0.058**	0.060	0.065	0.070
5	**0.064**	0.066	0.072	0.076

TABLE 6. Performance comparison in terms of local load balancing.

UserRequest	PS-GWCA	MOPPSO	GA-MOCA	Spread
1	**0.017**	0.019	0.030	0.031
1.5	**0.028**	0.030	0.041	0.044
2	**0.037**	0.041	0.055	0.056
2.5	**0.046**	0.051	0.061	0.068
3	**0.057**	0.062	0.071	0.081
3.5	**0.068**	0.072	0.085	0.092
4	**0.080**	0.084	0.095	0.0105
4.5	**0.090**	0.094	0.105	0.119
5	**0.099**	0.105	0.116	0.130

TABLE 7. Performance comparison in terms of service reliability.

UserRequest	PS-GWCA	MOPPSO	GA-MOCA	Spread
1	**32.49**	32.96	35.13	36.07
1.5	**47.56**	48.61	51.49	52.79
2	**62.73**	63.95	69.09	68.54
2.5	**76.66**	78.89	82.73	83.50
3	**91.63**	93.13	99.24	99.33
3.5	**104.83**	107.02	113.05	113.54
4	**118.57**	120.92	126.10	126.52
4.5	**131.94**	134.21	142.96	140.18
5	**144.36**	147.22	155.49	151.99

TABLE 8. Performance comparison in terms of speed.

UserRequest	PS-GWCA	MOPPSO	GA-MOCA	Spread
1	8.05	3.9	519.58	1.72
1.5	8.11	4.34	556.22	2.52
2	10.44	4.16	589.94	3.33
2.5	7.54	5.71	611.03	4.20
3	8.28	8.86	626.36	4.88
3.5	7.38	7.9	663.00	5.93
4	7.81	7.9	681.74	6.56
4.5	7.591	5.7	779.46	7.24
5	7.76	6.06	1017.02	8.13

As can be seen from Table 7, in terms of the optimal value of the four algorithms, the PS-GWCA algorithm proposed in this paper has an average optimization of 1.91% compared with MOPPSO-CMS algorithm in terms of service reliability, 7.49% compared with GA-MOCA algorithm, and 7.68% compared with Spread algorithm.

As can be seen from Table 8, in terms of local load balancing, the PS-GWCA algorithm proposed in this paper

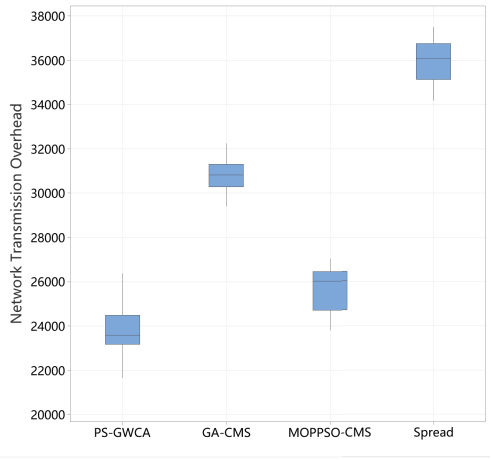


FIGURE 12. Box diagram of network transmission cost.

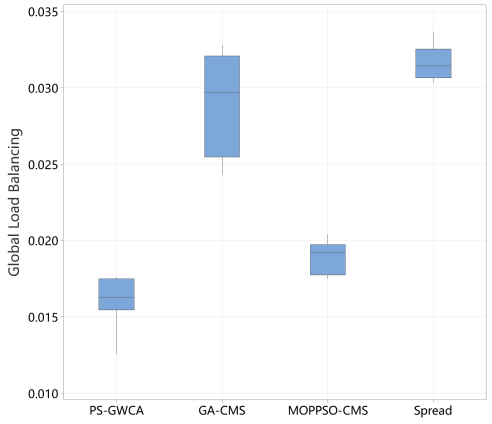


FIGURE 13. Box diagram of global load balancing.

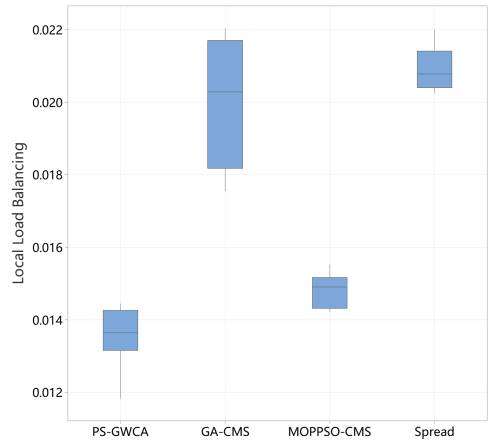


FIGURE 14. Box diagram of local load balancing.

is slightly worse than MOPPSO-CMS in terms of optimization time,because of PS-GWCA requires Pack Factory to synchronize and process different algorithms, while MOPPSO only shares the optimal solution among algorithms PS-GWCA is slightly slower than MOPPSO much better than GA-MOCA, and slightly worse than Spread.

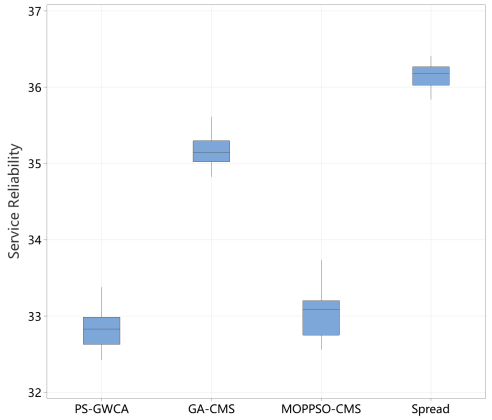


FIGURE 15. Box diagram of service reliability.

As can be seen from Figure 12-15, the PS-GWCA algorithm proposed in this paper has better performance in most of the time, and has good performance in network transmission cost, local load balancing, global load balancing and service reliability. The logic of Spread algorithm is simple, and it has the shortest optimization time in most cases. However, Spread algorithm cannot find suitable container scheduling scheme under the constraints of various conditions, and performs poorly in many aspects. Ga-MOCA algorithm is a multi-objective optimization algorithm based on NSGA-II. Due to the characteristics of crossover and variation of the algorithm, a large number of invalid solutions are generated under the condition of obvious constraints on the number of containers. In order to make up for these invalid solutions to reach the population size, GA-MOCA needs to regenerate the population and iterate again. This will cause the optimization of the algorithm to increase the time and reduce the efficiency of optimization. Therefore, GA-MOCA algorithm is superior to Spread algorithm in results, but compared with other algorithms, it has shortcomings such as longer searching time and unstable searching. MOPPSO-CMS algorithm is an algorithm based on PSO algorithm. Due to the fast search speed of PSO, MOPPSO-CMS algorithm can also find the appropriate scheduling scheme in an acceptable time. However, PSO is prone to fall into the local optimal solution. Although parallelism can alleviate this problem to some extent, it still has some shortcomings. The PS-GWCA algorithm proposed in this paper has a longer optimization time than MOPPSO-CMS algorithm and Spread algorithm, but it also finds a suitable scheduling scheme in an acceptable time. PS-GWCA algorithm can use grey wolves at the early stage of the search algorithm of the search features to guide the PSO to jump out of local optimal and enhance the global optimization ability of the algorithm, late in the search, the PS - GWCA algorithm make use of the search characteristics of PSO algorithm, the pareto optimal front sufficient local optimization, guide the algorithm to find a better solution. Therefore, the algorithm proposed in this paper has better and more stable performance in network transmission cost, global

load balancing, local load balancing, service reliability and other aspects, and has stronger comprehensive optimization ability.

VII. CONCLUSION AND PROSPECT

In order to solve the micro-service container scheduling problem, avoid the algorithm falling into local optimal, improve the global optimization ability of the algorithm, improve the speed of the algorithm, this paper proposes the PS-GWCA algorithm, which uses the global optimization ability of the early GWO to improve the ability of PSO to jump out of local optimal. The local optimization ability of PSO in the pareto front was used to make up for the weakness of GWO in the later period, and a new communication mechanism between populations was developed, so that the two algorithms could cooperate in the optimization, and the computational resources were timely allocated to the appropriate algorithm when an algorithm performed badly. Experiments show that the proposed algorithm has good performance in many aspects.

In the future, will consider more time to further improve the existing model and algorithm, there are many changes in the process of practical application, the user's request will not wait for all containers deployed before entering, in the face of a steady stream of user requests, how to solve the problem of rapidly changing dynamic scheduling will be the next step research direction.

REFERENCES

[1] T. Wen-Yu, "Design and implementation of SOA architecture oriented microservice security system," M.S. thesis, Dept. Softw., Nanjing Univ., Nanjing, China, 2016.

[2] L. Jian, "Design and implementation of deep learning platform based on container technology," M.S. thesis, Dept. Comput., Beijing Univ. Posts Telecommun., Beijing, China, 2020.

[3] A. Freeman, *Essential Docker for ASP.NET Core MVC*. Berkeley, CA, USA: Apress, 2017, pp. 1–6.

[4] J. Liu, X. Wei, and H. Huang, "An improved grey wolf optimization algorithm and its application in path planning," *IEEE Access*, vol. 9, pp. 121944–121956, 2021, doi: 10.1109/ACCESS.2021.3108973.

[5] Y. F. Yu, G. Li, and C. Xu, "An improved particle swarm optimization algorithm," *Applied Mechanics and Materials*, vol. 401. Zürich, Switzerland: Trans Tech Publications, 2013, pp. 1328–1335.

[6] S. Chamaani, S. A. Mirtaheri, M. Teshnehlab, and M. A. Shooredeli, "Modified multi-objective particle swarm optimization for electromagnetic absorber design," in *Proc. Asia–Pacific Conf. Appl. Electromagn.*, Dec. 2007, pp. 1–5, doi: 10.1109/APACE.2007.4603923.

[7] Z. Jing, D. Shou-Bing, and T. De-Yu, "Cloud computing scheduling algorithm based on intrusion tumor growth optimization," *Chin. J. Comput.*, vol. 41, no. 6, pp. 1140–1155, 2018.

[8] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[9] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," Presented at the Int. Conf. Parallel Problem Solving From Nature, Berlin, Germany, 2000.

[10] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[11] C. Guerrero, I. Lera, and C. Juiz, "Genetic algorithm for multi-objective optimization of container allocation in cloud architecture," *J. Grid Comput.*, vol. 16, no. 1, pp. 113–135, Mar. 2018.

[12] M. Lin, J. Xi, W. Bai, and J. Wu, "Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud," *IEEE Access*, vol. 7, pp. 83088–83100, 2019, doi: 10.1109/ACCESS.2019.2924414.

[13] D. Bin-Tao and X. Sheng-Chao, "Multidimensional resource balanced scheduling of container cloud based on improved bacterial foraging algorithm," *J. Yunnan Normal Univ., Natural Sci. Ed.*, vol. 41, no. 5, pp. 28–32, 2021.

[14] Z. Wei-Guo, M. Xi-Lin, and Z. Jin-Zhong, "Research on kubernetes' resource scheduling scheme," in *Proc. 8th Int. Conf. Commun. Netw. Secur.*, Qingdao, China, Nov. 2018, pp. 144–148.

[15] D. Patel, M. K. Patra, and B. Sahoo, "GWO based task allocation for load balancing in containerized cloud," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Wuhan, China, Feb. 2020, pp. 655–659.

[16] K. Pan and J. Chen, "Load balancing in cloud computing environment based on an improved particle swarm optimization," in *Proc. 6th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Beijing, China, Sep. 2015, pp. 595–598.

[17] Y. Zhang and R. Yang, "Cloud computing task scheduling policy based on improved particle swarm optimization," in *Proc. 43rd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Beijing, China, Aug. 2017, pp. 8768–8772.

[18] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," *Future Gener. Comput. Syst.*, vol. 65, pp. 140–152, Dec. 2016.

[19] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, Feb. 2017.

[20] L. Li, J. Chen, and W. Yan, "A particle swarm optimization-based container scheduling algorithm of Docker platform," in *Proc. 4th Int. Conf. Commun. Inf. Process.*, Qingdao, China, 2018, pp. 12–17.

[21] A. R. Jordehi, "Binary particle swarm optimisation with quadratic transfer function: A new binary optimisation algorithm for optimal scheduling of appliances in smart homes," *Appl. Soft Comput.*, vol. 78, pp. 465–480, May 2019.

[22] L.-D. Chou, H.-F. Chen, F.-H. Tseng, H.-C. Chao, and Y.-J. Chang, "DPRA: Dynamic power-saving resource allocation for cloud data center using particle swarm optimization," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1554–1565, Jun. 2018.

[23] A. R. Jordehi, "Enhanced leader particle swarm optimisation (ELPSO): An efficient algorithm for parameter estimation of photovoltaic (PV) cells and modules," *Sol. Energy*, vol. 159, pp. 78–87, Jan. 2018.

[24] Z. Liu, X. Lv, and C. Jiang, "Application of PSO based on simulated annealing algorithm in container scheduling," *Comput. Meas. control*, vol. 29, no. 12, pp. 177–183, 2021.

[25] G. Fan, L. Chen, H. Yu, and W. Qi, "Multi-objective optimization of container-based microservice scheduling in edge computing," *Comput. Sci. Inf. Syst.*, vol. 18, no. 1, pp. 23–42, 2021.

[26] A. R. Jordehi, "Time varying acceleration coefficients particle swarm optimisation (TVACPSO): A new optimisation algorithm for estimating parameters of PV cells and modules," *Energy Convers. Manage.*, vol. 129, pp. 262–274, Dec. 2016.

[27] X. Chen and S. Xiao, "Multi-objective and parallel particle swarm optimization algorithm for container-based microservice scheduling," *Sensors*, vol. 21, no. 18, p. 6212, 2021.

[28] A. Rezaee Jordehi, "A mixed binary-continuous particle swarm optimisation algorithm for unit commitment in microgrids considering uncertainties and emissions," *Int. Trans. Electr. Energy Syst.*, vol. 30, no. 11, Nov. 2020, Art. no. e12581.

[29] L. Wu and H. Xia, "Particle swarm optimization algorithm for container deployment," *J. Phys., Conf. Ser.*, vol. 1544, no. 1, 2020, Art. no. 012020.

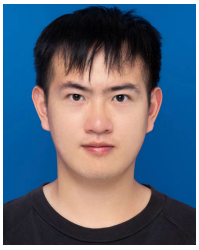
[30] B. Ma, H. Ni, X. Zhu, and R. Zhao, "A comprehensive improved salp swarm algorithm on redundant container deployment problem," *IEEE Access*, vol. 7, pp. 136452–136470, 2019, doi: 10.1109/ACCESS.2019.2933265.

[31] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, 1995, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[32] S. Mohanty, B. Subudhi, and P. K. Ray, "A new MPPT design using grey wolf optimization technique for photovoltaic system under partial shading conditions," *IEEE Trans. Sustain. Energy*, vol. 7, no. 1, pp. 181–188, Jan. 2016, doi: 10.1109/TSTE.2015.2482120.
[33] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in *Proc. IEEE/ACM 27th Int. Symp. Quality Service (IWQoS)*, Phoenix, AZ, USA, Jun. 2019, pp. 1–10.

XINYING CHEN received the B.E. degree in computer science and technology and the M.E. degree in computer software and theory from Jilin University, China, in 2002 and 2005, respectively, and the Ph.D. degree in computer application technology from Dalian Maritime University, China. She is currently an Associate Professor with the School of Computer and Communication Engineering, Dalian Jiaotong University. Her research interests include data mining, intelligent information processing, the Internet of Things, and the Semantic Web of Things.

YUEFENG WU received the B.E. degree from Dalian Jiaotong University, in 2020, where he is currently pursuing the M.E. degree with the School of Computer and Communication Engineering. His research interests include intelligent optimization algorithms, cloud computing, and microservice container scheduling.



SIYI XIAO received the B.E. degree from the City Institute, Dalian University of Technology, in 2019, and the M.E. degree from the School of Computer and Communication Engineering, Dalian Jiaotong University, in 2022. His research interests include intelligent optimization algorithms, the Internet of Things, and cloud computing.