A Multi-Swarm PSO Approach to Large-Scale Task Scheduling in a Sustainable Supply Chain Datacenter

Qi Liu †, *Senior Member, IEEE,* Lei Zeng †, *Member, IEEE,* Houbing Song, *Senior Member, IEEE,* Muhammad Bilal *, *Senior Member, IEEE,* Xiaodong Liu, *Senior Member, IEEE,* Yonghong Zhang, Xuefei Cao

Abstract—With the development of cloud computing and data intelligence, datacenters have become an important part of ensuring service quality and production efficiency in intelligent applications. However, datacenters are also facing increasingly complex and heavy task processing requirements currently, and more efficient scheduling methods are urgently needed. Therefore, this paper proposes a multi-swarm particle swarm optimization task scheduling method based on load balancing, aiming at reducing the maximum completion time (makespan) and response time in task scheduling. The proposed method designs a new fitness function for particles, and promotes the load balance of the cluster during the scheduling process by optimizing the combination of makespan and machine completion time variance. And a novel inertia weight is designed to dynamically adjust the particle search performance. The new initialization method and multi-swarm search design are used to improve the quality and diversity of solutions and avoid particles falling into local optimum. Finally, the proposed algorithm is verified experimentally using the task dataset released by Alibaba datacenter, and compared with other benchmark algorithms. The results show that the algorithm can improve the task scheduling performance of datacenters in supply chain management when dealing with different workloads and changes in the number of machines.

Index Terms—Particle Swarm Optimization, Supply Chain, Datacenter Management, Sustainable Task Scheduling, Load Balancing.

1 Introduction

In today's globalized and competitive business environment, supply chain management has become an important strategic function that can significantly impact the profit of companies. As the government and enterprises pay more and more attention to modern manufacturing systems, digitization has gradually become the trend of future development. Currently, manufacturing companies use a top-down decision-making approach to allocate production demand from overall equipment efficiency [1]. This approach ignores machine conditions such as excess inventory or unplanned downtime. At the same time, due to the lack of connection between factories, the efficiency of operation management is low. Establishing data-driven decision-making services is an important part of driving supply chain management intelligence, which can help reduce risk and improve efficiency. Therefore, effective supply chain management can provide a competitive advantage by improving operational efficiency, reducing costs, increasing customer satisfaction, and fostering innovation [2].

The development of cloud computing and Internet of Things technology has accelerated the pace of modern intelligent application construction. Data and computing are two important elements of intelligence, and both of them rely on datacenters to provide storage and computing capabilities [3]. Therefore, the datacenter has become an important underlying resource supply chain, providing computing and storage resource services for many applications. The salient feature of the datacenter is to ensure the access capability of building services and the controllability of service bandwidth, and to provide large-scale service scheduling capabilities. With the construction of data intelligence, the scale of datacenters and the complexity of task processing continue to increase, and the difficulty of supply chain resource management in data centers also increases [4]. Therefore, designing a reasonable scheduling algorithm is helpful to improve the efficiency of datacenter task scheduling and resource management, and realize the centralized management and dynamic use of physical server resources and virtual resources. This is of great help to enterprises in building a dynamic, flexible, and adaptable infrastructure to support business growth and sustainable development [5].

Datacenter task scheduling is a typical scenario of supply chain management, and it is an important way to ensure

- *†Both Authors Contribute equally to this paper.*
- ** Corresponding Author*

- *Qi Liu. Engineering Research Center of Digital Forensics, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: qi.liu@nuist.edu.cn*
- *Lei Zeng. School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: 20201220002@nuist.edu.cn*
- *Houbing Song. Security and Optimization for Networked Globe Laboratory (SONG Lab), Embry-Riddle Aeronautical University, Daytona Beach, FL 32114 USA. E-mail: h.song@ieee.org*
- *Muhammad Bilal. Department of Computer and Electronics Systems Engineering, Hankuk University of Foreign Studies, Gyeonggi-do, 17035, Republic of Korea. E-mail: m.bilal@ieee.org*
- *Xiaodong Liu. School of Computing, Edinburgh Napier University, Edinburgh, EH10 5DT, UK. E-mail: x.liu@napier.ac.uk*
- *Yonghong Zhang. School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: zyh@nuist.edu.cn*
- *Xuefei Cao. school of Cyber Engineering, Xidian University, No. 2 South Taibai Rd, Xi'an 710071, China. E-mail: xfcao@xidian.edu.cn*

Manuscript

the safe and efficient execution of user tasks. On the one hand, cloud service providers need to control costs. On the other hand, downstream consumers in the supply chain need to ensure stable and reliable services. Due to the complexity of task scheduling, the computing power resources of the datacenter are often not fully utilized, which will also lead to a decline in service quality [6]. The datacenter environment is highly dynamic, the machine resources are heterogeneous, and the types of tasks are complex and diverse. The datacenter task scheduler is a key component of execution scheduling. It matches corresponding resources for tasks and ensures that tasks can be completed accurately. At the same time, the scheduler should also consider the efficient allocation of cluster resources to maximize resource utilization [7].

The development of data intelligence has promoted the continuous expansion of cloud computing market demand. The types of tasks are more diverse, and the number of tasks is growing rapidly. This also prompts the scale of the datacenter to be further expanded to meet more service demands. Therefore, how to allocate resources reasonably and flexibly to meet the different requirements of tasks has become a very challenging problem in task scheduling of sustainable supply chain datacenters [8]. When scheduling tasks, first sort the tasks according to certain rules to form an executable task sequence. When the optimization goals are different, the constraints used for sorting are also different. Task scheduling methods can be divided into static task scheduling and dynamic task scheduling [9]. The scheduling effect of static task scheduling will decrease when the number of tasks increases. Dynamic task scheduling can be adaptively adjusted according to the state of the cluster. Computing the optimal schedule is an NP-hard problem, therefore, a feasible approach is to find its approximate optimal solution [10]. The meta-heuristic algorithm belongs to the dynamic task scheduling method, which combines the random algorithm and the local search algorithm, and can dynamically search for the optimal solution in the problem solution space. For example, the particle swarm optimization algorithm can use a group of particles to quickly search for the optimal solution in the solution space of the problem, and in the search process, evaluate the quality of the solution through the fitness function, and finally converge to a stable value [11].

Datacenter task scheduling research has made a lot of progress, but there are still deficiencies that need to be improved [12]. First-come-first-served (FCFS) schedules tasks in the order they arrive, without considering the characteristics of the tasks and the load of the cluster machines. The principle of the shortest job first (SJF) is to prioritize the scheduling of shorter tasks. This increases the priority of short jobs, but it is easy to cause long tasks to be blocked or even starved to death. Therefore, the scheduling effect of the SJF task scheduling method is not good when the number of tasks increases. In order to obtain a better scheduling effect, it is not possible to simply use these fixed rules for task scheduling. The dynamics and complexity of the cloud environment put forward higher requirements for task scheduling. The particle swarm optimization algorithm can obtain the optimal solution [13] of the task scheduling sequence through the dynamic exploration of the particles in the solution space. Zhao et al. [14] proposed a task scheduling based on particle swarm optimization, which allocates computing elements to individual tasks to minimize the total time cost of processing requested tasks. However, the particle swarm optimization algorithm also has some disadvantages, such as easily falling into local optimum, insufficient stability, etc. Moreover, the results of the particle swarm optimization method are also very dependent on the quality of the initial solution, and different initial solutions often lead to different final solutions. The simulated annealing particle swarm optimization algorithm adopts particle jumping with a certain probability, which can alleviate the problem of particle swarm falling into local optimum [15]. But it does not guarantee stability and lacks the constraints of a load-balancing mechanism. The motivation for this study is the desire to improve service quality and productivity in sustainable supply chain datacenter management. Scheduling tasks to appropriate machines according to their characteristics is an important way to improve the efficiency and quality of datacenter supply chain management services [16]. Setting a load-balancing constraint mechanism can help reduce the response time of tasks of different lengths and avoid certain types of tasks from being blocked.

This paper analyzes the task characteristics and machine resource utilization of the Alibaba datacenter, and summarizes the existing problems. Aiming at optimizing the makespan and response time in datacenter task scheduling, a multi-swarm particle swarm optimization algorithm based on load balancing is proposed. The main contributions of this paper can be summarized as follows. First, a new particle swarm optimization fitness function is designed, which combines the optimization of the completion time and the variance of the machine completion time to improve the cluster load balance. Second, a novel adaptive inertia weight is designed to dynamically tune the particle search performance. Third, a new initialization method and multi-swarm design are used to improve the quality and diversity of solutions and avoid falling into local optimum. Finally, the proposed method is verified experimentally under different environments using the Alibaba datacenter task dataset. The results show that the proposed method can achieve better scheduling results than other baseline methods under different workloads and elastically varying machine numbers.

The related work of task scheduling in sustainable supply chain datacenter is reviewed in Section 2. The workloads and resource characteristics of the Alibaba datacenter are analyzed in Section 3. The proposed multi-swarm PSO approach is described in Section 4, and the performance evaluation is presented in Section 5. Section 6 summarizes the research work and looks forward to future directions.

2 RELATED WORKS

As cloud infrastructure has gradually become the basic supply chain of many industries, it carries diverse and massive service demands. The stability and efficiency of the basic supply chain is an important part to ensure the service quality and production efficiency of many other industries [17]. Due to the rapid growth of the number of users and the diversification of service requirements, the scale of data centers is getting larger and the structure is becoming more and more complex. In the cloud environment, there is randomness in the task arrival rate, task duration and resource requirements. Efficient task scheduling is conducive to improving datacenter service quality and resource utilization efficiency. Therefore, how to efficiently and reasonably schedule tasks to be executed on suitable machines in a random environment has been widely concerned by researchers [18].

Mahmud et al. [19] proposed an adaptive hyperheuristic multi-objective optimization method for integrated supply chain scheduling problems, where supplier, manufacturer and batch decisions are simultaneously optimized to respond to heterogeneous customer demands with time window constraints. Gao et al. [20] proposed a distributed shortest remaining time first scheduling for datacenter networks. they utilize the estimated remaining stream size and available bandwidth to determine the remaining time for each stream and to prioritize them. Ajayi et al. [21] designed a carry-on weighted Round Robin approach in cloud workloads scheduling, which takes multiple workloads classes into account. Shirvani et al. [22] proposed a novel hybrid heuristic-based list scheduling algorithm for the task-dependent scheduling problem in heterogeneous cloud computing environments. Elmougy et al. [23] proposed a hybrid task scheduling algorithm (SRDQ) that combines Shortest Job First (SJF) and Round Robin Scheduling (RR). It takes into account dynamically variable task volumes and aims to reduce task response time and starvation issues. These methods are improvements to the heuristic algorithm. However, when dealing with complex changes in the cloud environment, heuristic algorithms often cannot achieve better results due to fixed and single rules.

The meta-heuristic algorithm combines random algorithms and local search, so it can dynamically search for optimal solutions [24]. Wu et al. [25] introduced a selection operator to improve standard PSO, aiming to solve the inefficiency of current methods. Cloudsim is used for verification, and the experimental results show that the improved PSO has higher search performance and convergence speed than the original PSO. However, the load balancing rate of this method is low, and further improvements are needed in terms of fitness function and inertia weight parameters. Kumar et al. [26] proposed a resource allocation model named PSO-cogent based on particle swarm optimization for efficient processing of applications. Compared with PSO and min-min algorithms, the optimization of PSO-cogent algorithm in resource utilization and throughput has been improved. Alsaidy et al. [27] proposed an improved initialization for PSO using heuristics, which initializes the PSO with the longest job to the fastest processor and shortest completion time algorithm. Dubey et al. [28] designed a new multi-objective CR-PSO task scheduling algorithm with deadline constraints. It was improved on the basis of traditional chemical reaction optimization and particle swarm optimization and combines the characteristics of optimal scheduling order of hybridization. Jiang et al. [29] exploited Alibaba as a case to study the characteristics of the collaborative distribution of batch jobs and online services in sustainable supply chain datacenter management.

Meziani et al. [30] proposed a hybrid algorithm combining particle swarm optimization and simulated annealing (SA-PSO) for the two-machine flowshop scheduling problem with coupled operations. Simulated annealing PSO accepts the current solution of the particle with a certain probability, so it can probabilistically jump out of the local optimal solution and find the global optimal. Liu et al. [31] introduced the simulated annealing algorithm in the adaptive PSO algorithm and proposed an adaptive simulated annealing particle swarm optimization algorithm. It aims to overcome the shortcoming that PSO is easy to fall into local extreme points and achieves certain results in terms of convergence speed. Wu et al. [32] described a new cluster scheduler, Aladdin. It adopts a multi-dimensional nonlinear capacity function to support constraint expressions and uses an optimized maximum flow algorithm to improve resource utilization. Wang et al. [33] used the multi-objective planning theory to establish a multi-objective optimal allocation model aiming at the sustainable utilization of regional water resources.

However, the task scheduling algorithm based on particle swarm optimization still needs to be optimized. For example, the particle swarm optimization algorithm is easy to fall into the local optimal solution, and the solution quality of random initialization of particles is unstable. The movement of particles is affected by local and global optimal solutions, and if particles are guided to a local optimal solution, it is difficult to escape from this local optimal solution. In the particle swarm optimization algorithm, the movement of particles will be affected by other particles, so there may be too many similar particles, resulting in a lack of diversity in the algorithm.

3 Analysis of workloads and machines

As a well-known cloud computing service provider, Alibaba not only boasts large amounts of cloud-based infrastructures but also supplies various cloud computing service needs for many industries around the world. It occupies an important position in the management of sustainable supply chain datacenters, and provides various services, such as computing and storage services, to consumers around the world every day. Therefore, the dataset of the real production scenario is suitable for the analysis and optimization of scheduling algorithms in a cloud environment. The operation data disclosed by Alibaba datacenter was selected as the objection of research for this paper. cluster-trace-v2017 is a dataset made public by Alibaba in 2017, and it contains data from about 1300 machines in 12 hours. By researching the state of the cluster, we can gain a better understanding of how resources are utilized and design better algorithms to distribute workloads to machines to achieve a higher quality of service.

As one of the most important attributes of a task, time can convey a lot of characteristic information about the task, such as the arrival time, duration, and deadline of the task. Firstly, the distribution characteristics of the tasks in the task dataset are analyzed. The statistical analysis results of the length distribution of the tasks are shown in Fig. 1. The distribution of the number of tasks in different durations is counted, the vertical axis represents the number

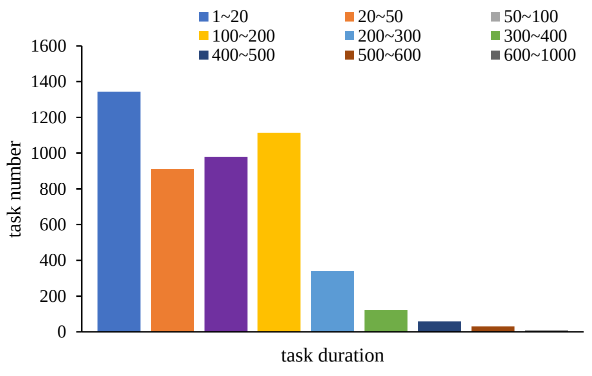

Fig. 1. Distribution of submission time interval of adjacent tasks.

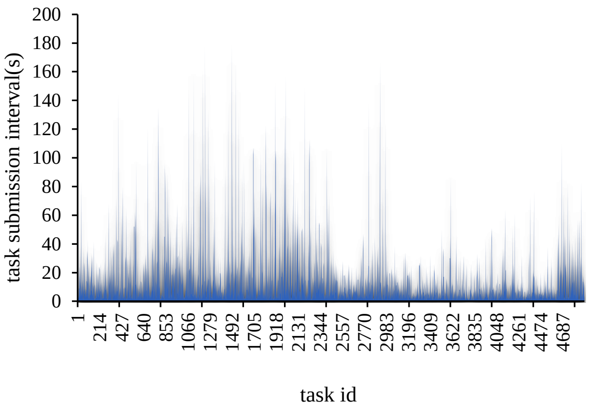

Fig. 2. Task duration proportion among all the tasks.

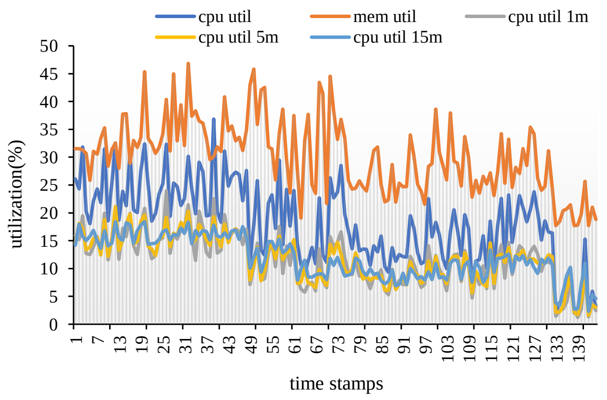

Fig. 3. Resource utilization of single machine through all timestamps.

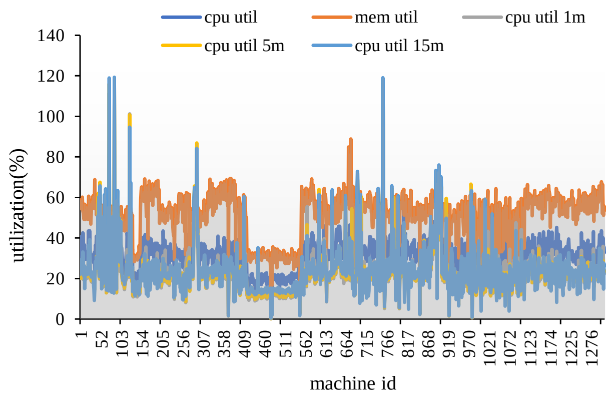

Fig. 4. Resource utilization of all machines at one timestamp.

of tasks, and the horizontal axis represents the duration of tasks. The duration of tasks in this dataset is all within 1000s, most of them are within 200s, accounting for 88.7%, tasks within 200-300s account for 7%, tasks within 300-400s account for 2.1%, and tasks exceeding 400s account for 2.2%. The number of tasks whose duration exceeds 200s gradually decreases. It reflects that the duration distribution of data center tasks is scattered and dynamic. Fig. 2 describes the distribution of the submission intervals of adjacent tasks, in which 83.7% of the job submission intervals are within 20s, and 13.7% are between 20-50s. Only 2.6% of tasks are submitted with a time interval of more than 50s. It reflects that the number of tasks arriving at the data center is relatively dense, but the task traffic is also relatively stable, with neither explosive growth nor cliff-like decline.

Resource utilization is the best indicator to reflect the load balance of the cluster, which may contribute to the design of the constraints of load balance. Therefore, the machine utilization table is also selected to analyze the cluster load characteristics, which contains the data of 187,963 machines. It selects 144 timestamps to record the resource utilization data of 1313 machines. These operation data include CPU utilization, memory utilization, hard disk utilization, and average CPU utilization in 15 minutes, 5 minutes, and 1 minute. It mainly focuses on CPU and memory utilization data, and analyzes resource utilization from the perspective of a single machine and the entire cluster. Fig. 3 shows the resource utilization of a single machine at all time stamps. It can be seen from the figure that the resource utilization of the machines has roughly the same change trend. The CPU utilization is low, whether it is the average utilization of 15 minutes, 5 minutes, and 1 minute, or the real-time utilization, most of the time is below 30%. Moreover, the utilization rate fluctuates greatly, and the load is unbalanced in each time period. Memory utilization is significantly higher than CPU utilization and varies with CPU utilization. Fig. 4 shows the resource utilization distribution of all machines in the cluster at the same timestamp. From the perspective of the entire cluster, the CPU utilization of all machines is still in a low state, most of which are 30% and below. The memory utilization of the machine is higher than the CPU utilization and remains around 60%. In addition, the resource utilization of machines numbered 415-533 is significantly lower than that of other machines, which indicates that the load is imbalanced in the cluster.

4 Proposed Multi-Swarm PSO Approach

Aiming at the problems of long completion time and slow task response in the task scheduling of the supply chain datacenter, this section proposes a multi-swarm particle swarm optimization method to optimize the task scheduling process of the data center. This section contains two subsections. The first part is problem formulation and system model in supply chain data center management, which introduces the characteristics of the supply chain datacenter task delivery and scheduling process. The second part introduces the design and improvement of the proposed multi-swarm particle swarm optimization algorithm based on load balancing.

4.1 Problem formulation and System model

Task scheduling is a mapping process between tasks and computing resources, and the scheduler will perform task scheduling and resource allocation in the cloud environment supply chain according to constraints. The task scheduler compares the requirements of the task with the characteristics of the resource, assigns the task submitted by the user to the most suitable machine for execution, and returns the result. The mathematical form of task scheduling can be described as follows.

Datacenter broker plays the role of the scheduler, which receives n number of tasks request which is independent of each other. The task list can be defined as $T = \{t_1, t_2, t_3, \cdots, t_n\}$, where n denotes the number of tasks in T and t_i represents the ith task. Each task t_i can be described as a vector $t_i = (l_i, m_i, p_i, d_i)$, which l_i represents the length of t_i in MIPS, and m_i , p_i , d_i denotes the memory, processor and deadline required by t_i, respectively. The cluster contains the m number of heterogeneous cloud resources which can be defined as $R = \{r_1, r_2, r_3, \cdots, r_m\}$, which are different in terms of MIPS speed, the core numbers, RAM, hypervisor, etc. Each virtual machine r_j can also be described as a vector $r_j = (mips_j, core_j, ram_j, hyper_j)$. The scheduler will match the corresponding machine according to the resource requirements of the task to ensure that the task is completed on time. Since the number of tasks in a cloud data center is usually far greater than the number of machine resources, the scheduler needs to take into account both the task completion time and response time.

The time from the start of execution to the end of a task on a particular machine is called the execution time, which can be formulated in Eq. (1), l_i and $mips_j$ represent the length of task t_i and MIPS speed of the virtual machine r_j, respectively.

$$ET_i = \frac{l_i}{\text{mips}_j} \tag{1}$$

Makespan is one of the most commonly used evaluation indicators in task scheduling, which represents the maximum completion time of all machines. The makespan is calculated in Eq. (2), where $CT_j = \gamma_j - start(r_j)$, γ_j represents the time when all the tasks assigned to jth virtual machine have been accomplished and $start(r_j)$ is the time instance when jth virtual machine starts its execution. CT_j represents the completion time of all tasks assigned to the jth virtual machine. The maximum completion time is taken as makespan (MS). The average completion time of all machines is also calculated as Eq. (3) during each iteration for the fitness function to evaluate the load balancing of the cluster.

$$MS = Max\{CT_1, CT_2, ..., CT_m\} \tag{2}$$

$$MS\_AVG = \frac{\sum_{j=1}^{m} CT_j}{m} \tag{3}$$

The response time of a task is the time to wait between the task arriving and getting executed. $RT_(t_i) = \vartheta_(t_i) - \theta_(t_i)$, where $\vartheta_(t_i)$is the time when task t_i start to execute and $\theta_(t_i)$ denotes the arrival time of task t_i. However, Since the number of tasks is large, it is not appropriate to collect the response time of each task and exploit it to evaluate the scheduling efficiency. Therefore, the average response time is usually chosen in scheduling performance evaluation. The average response time of virtual machine r_j is calculated in Eq. (4), where K represents the number of tasks allocated to r_j. In this paper, not only the average response time of the assigned tasks on each virtual machine is calculated, but also the variance of the average response time of all machines. So that the load balancing of different machines can be presented clearly. The global average response time can be calculated through Eq. (5), where m is the number of virtual machines mentioned in the cloud resource above.

$$AVG\_RT_{r_j} = \frac{\sum_{i=1}^{K} RT_{t_i}}{K} \tag{4}$$

$$AVG\_RT_{global} = \frac{\sum_{j=1}^{m} AVG\_RT_{r_j}}{m} \tag{5}$$

The tasks processed by the datacenter are various, and the number of tasks is far greater than the number of computing nodes. Cloud computing virtualizes the physical machines of the datacenter into more virtual resources through virtualization technology. Each virtual resource has certain attributes, such as CPU, storage capacity, network bandwidth, etc., and can process tasks independently without affecting each other. In this case, each node needs to execute multiple tasks in queued order. The task scheduling optimization model refers to the optimization problem-solving model established for the computing needs of massive tasks in the datacenter. The process includes building models, generating strategies, scheduling adjustment and optimization, and applying scheduling strategies, so that tasks are reasonably assigned to appropriate nodes for execution. The scheduler will perform task scheduling and resource allocation in the datacenter according to constraints, and assign tasks submitted by users to the most suitable machines for execution.

The task delivery and scheduling model in supply chain datacenter management is shown in Fig. 5, which is mainly divided into four stages. In the first step, users submit tasks to the datacenter through the client network, forming a huge task pool, which concentrates the tasks waiting to be scheduled for execution. In the second step, the scheduler checks the current resource situation and allocates the corresponding virtual machine resources to the tasks. The work of the datacenter task scheduler is relatively complicated,

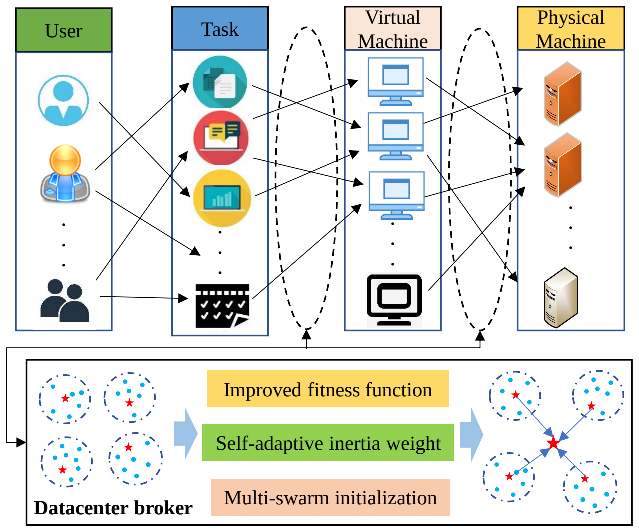
User
Task
Virtual Machine
Physical Machine
Improved fitness function
Self-adaptive inertia weight
Multi-swarm initialization
Datacenter broker

Fig. 5. Cloud datacenter task delivery and scheduling model.

and it needs to properly classify and sort the tasks in the task pool. At the same time, the scheduler also needs to observe the load of the virtual machine cluster, and then schedule tasks and allocate resources based on the information it has. In this process, the transmission of tasks and data is carried out in the internal network of the datacenter. The third step is to map the obtained virtual machine to the physical host to start executing the task. If the migration of virtual machines is involved in the process, there will be communication and latency overhead. Because it is necessary to shut down the virtual machine and transfer related data. The last step is the delivery phase of the task results, and the results are fed back to the user through the network. The datacenter scheduler is a key module of task scheduling, which schedules a large number of tasks submitted by users to appropriate virtual machines.

4.2 Multi-Swarm PSO Algorithm for Load Balancing

According to the results of workload and machine analysis, the load of the cluster is unbalanced in time and space. The load balancing of cluster machines is closely related to task response time and completion time. The PSO algorithm can dynamically search the solution space to find the optimal solution. However, the optimal solution finally searched by PSO depends heavily on its initialization. If the random initialization gives a poor solution, the final solution will also be of poor quality. These issues are closely related to scheduling efficiency, therefore, we need to avoid premature convergence of the algorithm to a local optimum. Meanwhile, it is necessary to design a more reasonable and effective fitness function to evaluate the search effect of particles.

In the design of the PSO algorithm, The particle has only two attributes: velocity and position. Velocity and position are varied according to algorithm rules during the execution of the search until the optimal solution is obtained. Depending on the dimension of the solution to the problem, the velocity and position of the particle are described as vectors of the corresponding dimension, respectively. Suppose there are N particles, and ϕ represents the dimension of the solution to the problem. Velocity represents the speed and direction of particle movement, which can be defined as a vector $v_i = (v_i^1, v_i^2, v_i^3, \cdots, v_i^\phi)$. Each scalar in the velocity vector represents the distance the particle travels in that dimension. The position is the current position of the particle, which is usually initialized with a set of random values and represents a solution to the problem even if not the best. Correspondingly, the position of each particle can be defined as a vector $x_i = (x_i^1, x_i^2, x_i^3, \cdots, x_i^\phi)$. In each round of iteration, the particle moves independently in the solution space and explores better solutions, and uses p_{best} to mark the optimal solution obtained by itself. The optimal solution of the whole swarm is obtained by comparing the optimal solutions of all particles. All particles in the group continuously adjust their speed and position during the search progress. Particles adjust their speed and position according to their individual extrema (p_{best}) and the global optimum (g_{best}). The PSO algorithm makes each particle obtain an initial velocity and position through random initialization and then finds the optimal solution through iteration and update. The updated formulas for the velocity and position of the particles are Eqs. (6) and (7), respectively.

$$v_i = v_i + c_1 \cdot \mu \cdot (p_{best} - x_i) + c_2 \cdot \mu \cdot (g_{best} - x_i) \quad (6)$$

$$x_i = x_i + v_i \quad (7)$$

where $i = 1, 2, ..., N$. N is the total number of particles in the swarm. x_i and v_i are the position and velocity of the particle, respectively. The μ is a random number between (0,1), which aims to increase the move randomness of particles. c_1 and c_2 are used to control the effect of individual and global extremes. The Eq. (6) is the update process of particle velocity, which includes three parts, namely, the memory item, the self-recognition item, and the group cognition item. Each of these three parts has a different role. They determine the movement of particles from the inertia of the particle's own motion, the knowledge acquired by its own exploration, and the globally shared knowledge. In this way, it can not only maintains the independence of particle exploration itself but also ensures the sharing of information between different particles, thereby improving efficiency and speeding up. After obtaining the update of the velocity, the update of the position is as shown in Eq. (7). The updated formula for particle velocity is improved as shown in Eq. (8).

$$v_i = w \cdot v_i + c_1 \cdot \mu \cdot (p_{best} - x_i) + c_2 \cdot \mu \cdot (g_{best} - x_i) \quad (8)$$

where w represents the weight of inertia, which is an important parameter. The parameter w greatly improves the optimization ability and generalization ability of the PSO algorithm. The global and local search capabilities of particles can be adjusted according to actual problem needs, which also enables the PSO algorithm to be applied to a wider range of practical problems. When the value of w is large, it has a good effect on improving the global search ability of the algorithm, and on the contrary, it enhances the local search ability of PSO. A dynamic value of w often has better results than a fixed value, and it can be dynamically adjusted according to the change of the particle search state in the current swarm, thereby ensuring a better result. In order to improve the ability of particles to explore the optimal solution in the process of task scheduling, a new nonlinear dynamic adaptive inertia weight is designed in this paper. As shown in Eq. (9).

$$w = \begin{cases} w_{\min} + (w_{\max} - w_{\min})\frac{(f - f_{\min})}{(f_{avg} - f_{\min})}, f \leq f_{avg} \\ w_{\max}, f > f_{avg} \end{cases} \tag{9}$$

where w_{max} and w_{min} represent the predetermined minimum and maximum inertia coefficients w in the algorithm. and f is the value of the current objective function of the particle. $f_{avg} = \sum_{i=1}^{n} x_i/n$ is the current average value of the objective function of all particles, and f_{min}=min$\{f(x_1), f(x_2), f(x_3), ..., f(x_n)\}$ represents the minimum value of the objective function of all particles. Linearly changing inertia weights refer to changing the value of the weights at regular intervals according to time. But this method is not very effective, because it does not consider the relationship between the current solution situation and the search capability of particles. When the objective function values of particles tend to be consistent, it indicates that they may fall into the local optimum, so increasing the relationship weight can expand the global search ability of particles. When the objective function value is scattered, it indicates the lack of refined search, and reducing the inertia weight can improve the particle's local exploration ability. The new inertia weight design changes the way of linear change, and instead adjusts according to the relationship between particles and the whole world. If the particle objective function value is smaller than the average value, the value of the inertia weight will be smaller to ensure that it continues to explore the local optimal solution. Instead, the particle's inertial weight increases, allowing it to explore other possibilities to avoid lingering around bad solutions. The new inertia weights are related to the fitness of each particle. The smaller the fitness, the closer to the optimal solution, requiring local search. The greater the fitness, the farther away from the optimal solution, requiring a global search.

The load balancing of the machine cluster has a great impact on the completion time and response time in the task scheduling process. Because if the load varies greatly between machines, some machines will be overloaded, which will increase both completion time and response time. According to the analysis results of workload and machine resource utilization, the load of the cluster is unbalanced in time and space. Therefore, it is not enough to optimize only makespan in the fitness function, because it is difficult to consider the load balancing of all machines in the entire cluster. By calculating the completion time of each machine after the execution of different scheduling algorithms, it is found that there are large differences between different machines. Among them, the proportion of the difference between the maximum completion time and the minimum completion time of all virtual machines in the FCFS scheduling method reaches 566%, the SJF scheduling algorithm is 185%, and the PSO algorithm is 385%.

Therefore, in the design of the fitness function, the method of evaluating only the makespan value is discarded. The objective function will evaluate the combinatorial optimality of makespan and the variance of the completion time among all machines during each iteration. In this way, the load balancing constraints between machines in the task scheduling process are realized. Therefore, in the design of the fitness function, the method of only evaluating the value of the completion period is abandoned. The new fitness function designs the combinatorial optimization of the makespan and the variance of the maketimes of all machines. On the one hand, it restricts the growth of makespan. On the other hand, the load variance among machines is reduced by constraining the variance of the completion time of cluster machines. When the load of the cluster machines is more balanced, the makespan and response time will be further reduced. The objective optimization function of fitness evaluation is shown in Eq. (10).

$$f = \delta \cdot MS + (1 - \delta)\frac{\sum_{j=1}^{m}(CT_j - MS\_AVG)^2}{m} \tag{10}$$

Where CT_j represents the calculated completion time of each machine, MS_AVG is the average completion time of all machines, and m represents the total number of virtual machines. The MS is the maximum of all the machines' completion time during each iteration. f is the optimization objective, and the purpose is to minimize its value. The δ is the optimization weight ratio, which is used to flexibly adjust the proportion of the optimization target.

By observing each complete search process of the particle swarm optimization algorithm, it is found that the optimal solution finally searched by the particle swarm optimization algorithm depends to a large extent on its initialization state. If random initialization gives a poor solution, then the quality of the final solution will also be relatively poor. Therefore, this section improves the particle initialization method to ensure that there is at least one solution with better quality in the particle community. So as to alleviate the possible poor results brought by randomness, and then search for a better solution in the solution space on this basis. By prioritizing both short and long tasks, the largest jobs are mapped to the best-performing machines, and the smallest jobs are mapped to the lowest-performing machines. Then map according to the sequence number and shrink towards the middle. After mapping a round of virtual machines, the loop continues until all tasks have been mapped.

The phenomenon that the particle swarm falls into the local optimum is that the number of iterations has not yet ended, and the particles have gathered near the local optimum solution and no longer move. When a particle in the particle swarm gets a better solution, other particles will move toward it. If the optimal particle is captured by the local optimal solution, the speed of other particles will gradually decrease until they lose their activity. Since a particle swarm can only generate one g_{best}, the information of other particles will be lost. Therefore, in the process of iteration, the search space will gradually shrink, and eventually all particles will fall into the local optimal position. Aiming at the problem that particle swarm may fall into local opti-

TABLE 1
Physical machine configuration.

PM	Mips	cores	Memory/GB	BD/(MB/s)
Host	3000	4	16	1200

TABLE 2
Virtual machine configuration.

VM	Small	Medium	Large
Processor/Mips	1000	1500	2000
Core numbers	1	1	1
Ram/GB	1	2	2
BD(MB/S)	100	100	100

mum, this paper adopts a new multi-swarm optimization design. Therefore, multiple particle swarms are designed to jointly search in the solution space, so that multiple g_{best} can be generated. Then the optimal particle is selected as the result to alleviate the problem that all particles fall into the local optimum. The advantage of this design is that the particles in different swarms are independent of each other and will not fall into local optimum together. Moreover, even if some swarms fall into the local optimum, the optimal solution can be selected among multiple swarms.

5 PERFORMANCE EVALUATION

This section contains two summaries, the experimental setup and the experimental results. Section 5.1 introduces the experimental environment, experimental settings and datasets. Section 5.2 conducts experiments to verify the performance of the proposed method in various experimental environments, and compares the performance with other scheduling methods.

5.1 Experimental Setup

In this paper, the proposed method load balancing multi-group PSO (LBMSPSO) is tested in the CloudSim simulation environment. First, a data center with 12 physical machines (PM) was built. The specific configuration parameters of the physical host are shown in the table 1. Then three types of virtual machines are established, each with different resource types, the specific configuration is shown in the table 2. The task dataset from the Alibaba data center is used as the task input for the experiments. The experiments assume that each task's memory and CPU requirements remain constant during its execution.

5.2 Experiment Result and Comparison

In this section, the proposed task scheduling method is experimentally verified and compared with existing heuristic and meta-heuristic methods. Algorithms chosen for comparison include FCFS, SJF, PSO, and simulated annealing PSO (SAPSO). The selected comparison metrics include makespan and task average response time. In addition, the variance of the completion time of the machine and the variance of the average response time of the machine are compared to evaluate the load balance of the cluster during task scheduling.

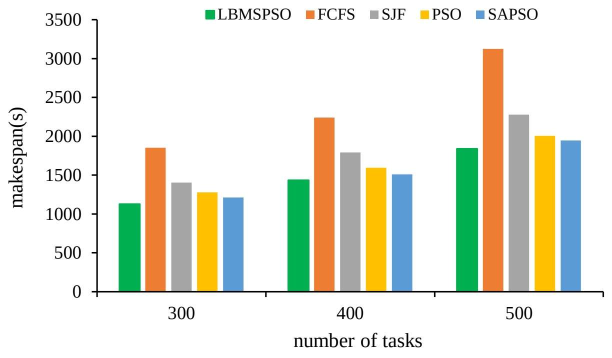

Fig. 6. Comparison of makespan under varying workload.

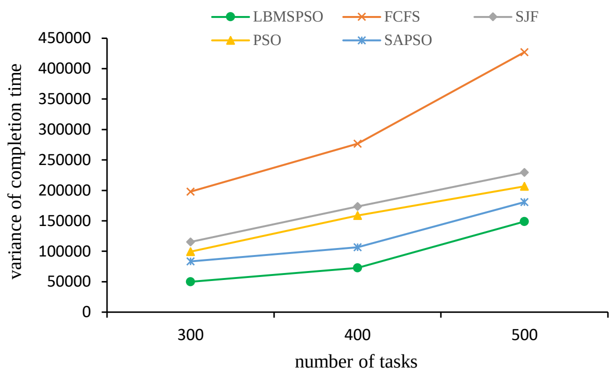

Fig. 7. Comparison of machine completion time Variances under varying Workloads.

The scheduling effect of the algorithm on changing workloads under a fixed number of machines is verified firstly. A heterogeneous virtual machine cluster is created, including 12 small, 8 medium and 6 large virtual machines, a total of 26 sets. Then change the number of tasks in the workload to test the scheduling effect of the proposed method. The results of makespan are shown in Fig. 6, and the number of tasks is set to 300, 400 and 500, respectively. The scheduling effect of FCFS is the worst, and as the number of tasks increases, its makespan grows the fastest. Therefore, FCFS has poor adaptability to task scheduling in the cloud environment. In the results of makespan, the method proposed in this paper can improve by an average of 39% under different loads. The scheduling principle of SJF is to schedule the shortest task first, which easily leads to long task blocking, which is not conducive to fair scheduling. As the number of tasks increases, the scheduling efficiency of SJF decreases, and the response time for long tasks increases. Compared with SJF, LBMSPSO can improve by an average of 20.4% under different workloads. The method proposed in this paper increases the constraints of the load balancing mechanism in the scheduling process, which can reduce the load difference between different machines. Moreover, new initialization methods and multi-community designs can improve the quality and diversity of solutions. Therefore,

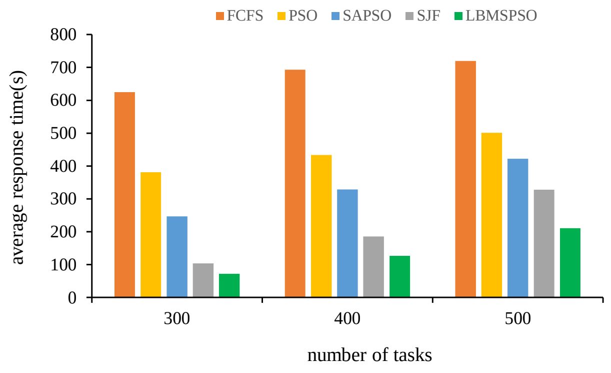

Fig. 8. Comparison of average task response time under different workloads.

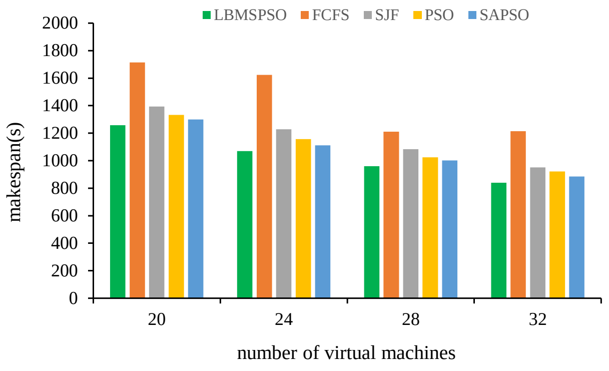

Fig. 9. Comparison of makespan under the varying number of virtual machines.

compared with PSO and SAPSO, LBMSPSO can improve the results of makespan by an average of 9.7% and 6.3%, respectively.

By recording the completion time of each virtual machine and calculating the variance of the machine completion time, it can reflect the load balance of the cluster during the task scheduling process. Therefore, the comparison of the variance of machine completion time in different workload task scheduling experiments is shown in Fig. 7. The load variance of FCFS is always at the maximum, and the variance of machine completion time grows rapidly when the number of tasks increases. This also explains the rapid increase in the results of makespan in FCFS scheduling. The first-come-first-served scheduling method does not consider the load balancing of machines. Since the proposed method adds a constraint on the cluster machine load difference in the fitness function, the load difference of LBMSPSO is always kept at the lowest.

The results of task average response time under different workloads are shown in Fig. 8. Corresponding to the result of makespan, the task average response time of FCFS is still the highest among all methods. However, unlike the results in makespan, the average task response speed of SJF surpasses PSO and SAPSO. This is due to the scheduling mechanism of SJF. Because SJF schedules short tasks first, and short tasks are executed in a faster time, so that the average task response time can be reduced. Therefore, even though the makespan values of PSO and SAPSO are lower than those of SJF, the average task response time is larger than that of SJF. LBMSPSO makes up for this defect. It uses an improved particle initialization method to obtain better results and avoid the influence of randomness. Therefore, compared with other baseline methods, LBMSPSO can further reduce the average response time of tasks on the basis of reducing makespan.

As the workload increases, both the task completion time and response time increase accordingly. When the workload is heavy and affects the quality of service, it is necessary to elastically increase the number of machines to relieve system pressure. Therefore, this section also tests the scheduling performance of the proposed method under an elastically varying number of virtual machines. A homogeneous virtual machine cluster containing 20 large virtual machines is created. Under the workload of 500 tasks, elastically increase the number of virtual machines for experimental comparison.

The results of makespan are shown in Fig. 9. Except for FCFS, as the number of virtual machines increases, the results of makespan for other scheduling methods decrease. The scheduling effect of FCFS is not stable. Increasing the number of virtual machines does not necessarily reduce the value of its makespan, and may even increase it. It shows that it is not effective in dealing with elastic changes in the number of virtual machines. Under the constraints of the load balancing mechanism, LBMSPSO can quickly and evenly distribute tasks to machines with lighter loads to reduce the makespan when the number of virtual machines increases. It can be seen from the figure that as the number of virtual machines grows elastically, LBMSPSO can achieve a better makespan than other benchmark methods. Therefore, LBMSPSO can effectively cope with the elastic change of the number of virtual machines when the workload is heavy in the task scheduling process.

The comparison of the variance of the machine completion time under different numbers of virtual machines is shown in Fig. 10. From the variance of the machine completion time, it can be seen that the variance of the machine completion time of FCFS is still the largest, and it fluctuates with the increase of the number of machines. The value of the variance of the FCFS machine make time also reflects the reason for the change of its makespan value. It shows that the machine load difference of FCFS task scheduling is the largest, and it is difficult to cope with the elastic change of the number of machines in the cloud environment. Under a stable load, when the number of virtual machines is elastically increased, SAPSO, PSO, SJF, and LBMSPSO can all maintain the stability of cluster machine load differences, and LBMSPSO performs optimally in maintaining load balancing.

The results of the average task response time are shown in Fig. 11. Under a stable workload, increasing the number of virtual machines can reduce system pressure and reduce task response time. Among them, the average task response time of FCFS is still the highest, but as the number of virtual

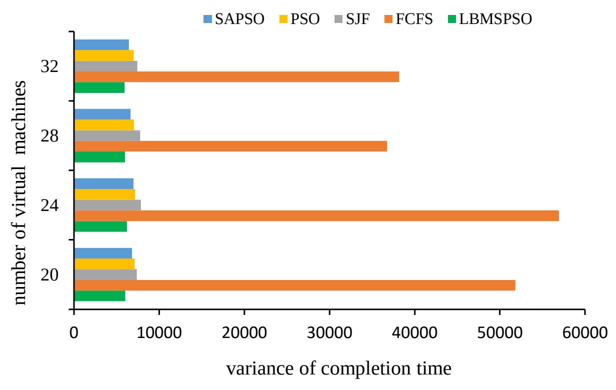

Fig. 10. Comparison of machine completion time Variances under the varying number of virtual machines.

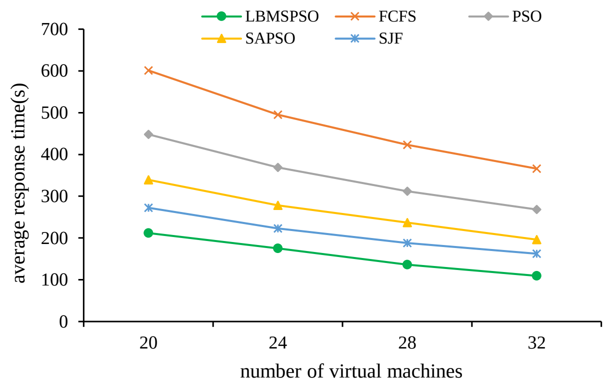

Fig. 11. Comparison of average task response time under the varying number of virtual machines.

machines increases, the response time gradually decreases. SJF lowers the average task response time because it prioritizes short tasks. Due to the impact of randomness and the lack of load balancing constraints, PSO and SAPSO do not perform well in the average response time of tasks. LBMSPSO can achieve the lowest average task response time, and when the number of virtual machines increases elastically, it can also maintain system load balance and further reduce response time.

6 CONCLUSION AND FUTURE WORK

Aiming at the task scheduling problem of sustainable supply chain datacenter, a task scheduling approach based on multi-swarm particle swarm optimization is proposed. The task and cluster machine data sets released by the Alibaba datacenter are analyzed, and the data analysis shows that there is a problem of uneven load in the cluster. By designing a new adaptive inertia weight to automatically adjust the local and global search capabilities of particles, particle search efficiency can be improved. The newly designed fitness function can more effectively evaluate the quality of particle solutions, so that the cluster can be optimized toward load balancing. The design of multi-swarm and new initialization method can promote the diversity and quality of the initial solution of particle swarm, and alleviate the problem that particle swarm is easy to fall into local optimum. The proposed method is experimentally tested and compared in a diverse set of environments. Experimental results show that the method proposed in this paper can maintain stable scheduling performance when the datacenter workload increases or the number of virtual machines changes elastically. And it outperforms other methods in makespan and task average response time.

However, task scheduling in sustainable supply chain data centers still needs to be optimized. How to achieve more coordinated and intelligent scheduling between batch jobs and online services to provide a higher quality of service and improve the throughput of batch jobs. How to deal with workflow scheduling with dependencies among tasks to improve system resource utilization efficiency. In future work, we plan to study artificial intelligence-driven sustainable supply chain data center task scheduling to further improve the intelligence level of datacenter management.

ACKNOWLEDGMENTS

This work has received funding from the Key Laboratory Foundation of National Defence Technology under Grant 61424010208, National Natural Science Foundation of China (No. 62002276, 41911530242 and 41975142), 5150 Spring Specialists (05492018012 and 05762018039), Major Program of the National Social Science Fund of China (Grant No. 17ZDA092), 333 High-Level Talent Cultivation Project of Jiangsu Province (BRA2018332), Royal Society of Edinburgh, UK and China Natural Science Foundation Council (RSE Reference: 62967_Liu_2018_2) under their Joint International Projects funding scheme and basic Research Programs (Natural Science Foundation) of Jiangsu Province (BK20191398 and BK20180794).

REFERENCES

[1] B. Li, R.-S. Chen, and C.-Y. Liu, "Using intelligent technology and real-time feedback algorithm to improve manufacturing process in iot semiconductor industry," *The Journal of Supercomputing*, vol. 77, no. 5, pp. 4639–4658, 2021.
[2] F. Bouhannana and A. Elkorchi, "Trade-offs among lean, green and agile concepts in supply chain management: Literature review," in *2020 IEEE 13th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA)*. IEEE, 2020, pp. 1–5.
[3] A. Dweekat and R. Al-Aomar, "An iot-enabled framework for dynamic supply chain performance management," in *2018 IEEE Technology and Engineering Management Conference (TEMSCON)*. IEEE, 2018, pp. 1–5.
[4] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, vol. 62, p. 100841, 2021.
[5] H. Shan, Y. Li, and J. Shi, "Influence of supply chain collaborative innovation on sustainable development of supply chain: a study on chinese enterprises," *Sustainability*, vol. 12, no. 7, p. 2978, 2020.
[6] M. Hussain, L.-F. Wei, A. Lakhan, S. Wali, S. Ali, and A. Hussain, "Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100517, 2021.
[7] M. S. Sangari and A. Mashatan, "A data-driven, comparative review of the academic literature and news media on blockchain-enabled supply chain management: Trends, gaps, and research needs," *Computers in Industry*, vol. 143, p. 103769, 2022.

[8] H. Yuan, J. Bi, and M. Zhou, "Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green iaas clouds," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5404–5412, 2019.

[9] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, "Artificial intelligence for edge service optimization in internet of vehicles: A survey," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 270–287, 2021.

[10] T. Ujazdowski and R. Piotrowski, "Task scheduling–review of algorithms and analysis of potential use in a biological wastewater treatment plant," *IEEE Access*, vol. 10, pp. 45 230–45 240, 2022.

[11] X. Zhu, M. Hussain, and X. Li, "Energy-efficient independent task scheduling in cloud computing," in *International Conference on Human Centered Computing*. Springer, 2018, pp. 428–439.

[12] W. Shu, K. Cai, and N. N. Xiong, "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing," *Future Generation Computer Systems*, vol. 124, pp. 12–20, 2021.

[13] B. Shen, X. Xu, L. Qi, X. Zhang, and G. Srivastava, "Dynamic server placement in edge computing toward internet of vehicles," *Computer Communications*, vol. 178, pp. 114–123, 2021.

[14] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597–633, 2019.

[15] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 2019.

[16] F. Abazari, M. Analoui, H. Takabi, and S. Fu, "Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simulation Modelling Practice and Theory*, vol. 93, pp. 119–132, 2019.

[17] A. Hmioui, B. Bentalha, and L. Alla, "Service supply chain: A prospective analysis of sustainable management for global performance," in *2020 IEEE 13th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA)*. IEEE, 2020, pp. 1–7.

[18] L. Versluis and A. Iosup, "A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies," *Future Generation Computer Systems*, vol. 123, pp. 156–177, 2021.

[19] S. Mahmud, A. Abbasi, R. K. Chakrabortty, and M. J. Ryan, "A self-adaptive hyper-heuristic based multi-objective optimization approach for integrated supply chain scheduling problems," *Knowledge-Based Systems*, p. 109190, 2022.

[20] C. Gao, V. C. Lee, and K. Li, "D-srtf: Distributed shortest remaining time first scheduling for data center networks," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 562–575, 2018.

[21] O. Ajayi, F. Oladeji, C. Uwadia, and A. Omosowun, "Scheduling cloud workloads using carry-on weighted round robin," in *International Conference on e-Infrastructure and e-Services for Developing Countries*. Springer, 2017, pp. 60–71.

[22] M. H. Shirvani and R. N. Talouki, "A novel hybrid heuristic-based list scheduling algorithm in heterogeneous cloud computing environment for makespan optimization," *Parallel Computing*, vol. 108, p. 102828, 2021.

[23] S. Elmougy, S. Sarhan, and M. Joundy, "A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique," *Journal of Cloud computing*, vol. 6, no. 1, pp. 1–12, 2017.

[24] N. Raman, A. B. Wahab, and S. Chandrasekaran, "Computation of workflow scheduling using backpropagation neural network in cloud computing: a virtual machine placement approach," *The Journal of Supercomputing*, vol. 77, no. 9, pp. 9454–9473, 2021.

[25] D. Wu, "Cloud computing task scheduling policy based on improved particle swarm optimization," in *2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*. IEEE, 2018, pp. 99–101.

[26] M. Kumar and S. C. Sharma, "Pso-cogent: Cost and energy efficient scheduling in cloud environment with deadline constraint," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 147–164, 2018.

[27] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of pso task scheduling algorithm in cloud computing," *Journal of King Saud University-Computer and Information Sciences*, 2020.

[28] K. Dubey and S. C. Sharma, "A novel multi-objective cr-pso task scheduling algorithm with deadline constraint in cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 32, p. 100605, 2021.

[29] C. Jiang, G. Han, J. Lin, G. Jia, W. Shi, and J. Wan, "Characteristics of co-allocated online services and batch jobs in internet data centers: a case study from alibaba cloud," *IEEE Access*, vol. 7, pp. 22 495–22 508, 2019.

[30] N. Meziani, M. Boudhar, and A. Oulamara, "Pso and simulated annealing for the two-machine flowshop scheduling problem with coupled-operations," *European Journal of Industrial Engineering*, vol. 12, no. 1, pp. 43–66, 2018.

[31] S. Liu, W. Liu, F. Huang, Y. Yin, B. Yan, and T. Zhang, "Multitarget allocation strategy based on adaptive sa-pso algorithm," *The Aeronautical Journal*, pp. 1–13, 2022.

[32] H. Wu, W. Zhang, Y. Xu, H. Xiang, T. Huang, H. Ding, and Z. Zhang, "Aladdin: Optimized maximum flow management for shared production clusters," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2019, pp. 696–707.

[33] Z. Wang, J. Tian, and K. Feng, "Optimal allocation of regional water resources based on simulated annealing particle swarm optimization algorithm," *Energy Reports*, vol. 8, pp. 9119–9126, 2022.