Journal of Applied Analysis and Computation
Volume 14, Number 1, February 2024, 436–457
Website:http://www.jaac-online.com
DOI:10.11948/20230266

RESEARCH ON SCHEDULING OF TWO TYPES OF TASKS IN MULTI-CLOUD ENVIRONMENT BASED ON MULTI-TASK OPTIMIZATION ALGORITHM*

Cuiyan Yi[1], Tianhao Zhao[1], Xingjuan Cai[1,2,†] and Jinjun Chen[3]

Abstract The multi-cloud environment (MCE) tasks can be classified as CPU-intensive or I/O-intensive. Using a single model to handle two tasks often results in system performance issues due to mismatches between task requirements and resource demands, caused by differing data characteristics. In this paper, a multi-task multi-objective optimization (MTMO) model is constructed. A multi-objective evolutionary algorithm with quadratic crossover is used to simultaneously schedule two types of tasks. This improves scheduling efficiency. First, according to the different data characteristics of tasks in MCE, tasks are separated into CPU-intensive tasks with large amounts of computation and high demand for CPU resources and I/O-intensive tasks that require frequent memory access. Different multi-objective optimization models are constructed according to the characteristics of per-task. Secondly, each multi-objective optimization model is constructed as a sub-task in a multi-task environment to build a MTMO model. Then, a multi-objective multi-factor evolutionary algorithm based on quadratic crossover, I-MOMFEA-II, is proposed to schedule the two types of tasks simultaneously. Finally, the proposed algorithm in this paper improved cost, time, and energy consumption for CPU-intensive tasks by 7.6%, 20.1%, and 16.1% respectively, for I/O-intensive tasks, it improved cost, time, and VM throughput by 10%, 17.7%, and 36.5% respectively. The experimental results from simulations confirm the effectiveness of I-MOMFEA-II in elevating task scheduling productivity.

Keywords Multi-tasking evolutionary algorithm, multi-objective optimization, task scheduling, multi-cloud.

MSC(2010) 68M20, 68W99, 93A30.

†The corresponding author.
[1]Shanxi Key Laboratory of Big Data Analysis and Parallel Computing, Taiyuan University of Science and Technology, Taiyuan, Shanxi 030024, China
[2]School of State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
[3]Department of Computing Technologies, Swinburne University of Technology, Hawthorn, VIC 3122, Australia
*The authors were supported by the National Natural Science Foundation of China (No.61806138) and China University Industry-University-Research Collaborative Innovation Fund (2021FNA04014).
Email: 17862078736@163.com(C. Yi), zhaotianhao1015@163.com(T. Zhao), xingjuancai@163.com(X. Cai), jinjun.chen@gmail.com(J. Chen)

1. Introduction

Cloud computing allows computing power and applications to be accessed as services through the Internet as needed. It has received considerable limelight from businesses and academics. This is courtesy of the budget computing capabilities and versatility of virtual machine technologies [2]. As Internet of Things (IoT) technology has grown and advanced swiftly, the massive growth in data volume has led to greater demand for computing resources [5]. However, the constraints inherent in traditional single-cloud environments make them insufficient for meeting user needs. Therefore, MCEs have gained significant attention in both industry and academia due to their numerous benefits [29]. They offer diverse virtual resources, improve data security and quality of service, and avoid single points of failure and vendor lock-in. This gives users more choices and better satisfies their demand for quality of service(QoS) [1]. Optimizing task scheduling on MCE is a significant research area. The goal is to map tasks and jobs to appropriate resources while meeting user requirements [19]. These requirements include cost, deadline, quality of service, and system throughput [32].

Scheduling issues are common. A good scheduling strategy impacts the performance of cloud resources [30]. It also affects cost issues for users and cloud providers who provide the infrastructure [25]. If a suitable and effective scheduling scheme is not available, not only will it lengthen the time for task execution and cause suboptimal utilization of cloud resources, but it will also lead to reduced availability and scalability of the system [9, 18]. To improve the scheduling performance, scholars have thoroughly investigated multi-cloud task scheduling from various angles [35]. In the past few years, numerous task scheduling schemes have been proposed in the literature that are based on MCE and efficient scheduling algorithms. For instance, the improved pigeon-inspired optimization algorithm and the enhanced multi-objective particle swarm optimization algorithm have been widely used in multi-cloud computing to schedule constrained cloud tasks in MCE [8]. These algorithms aim to balance multiple optimization objectives in cloud task scheduling in MCE [16].

However, previous studies have not considered the characteristics and demands of CPU-intensive and I/O-intensive tasks, and only one model is built to process the tasks. Tasks with different demands can be combined to optimize scheduling towards a common goal. However, this can lead to a mismatch between the task type and the required resource type. Existing scheduling methods are not capable of processing multiple tasks concurrently which leads to longer task execution times, especially when there is a large volume of tasks.

To solve these problems, a multitasking optimization algorithm is used in this paper. While traditional optimization algorithms usually treat each task as an independent problem to be solved, multi-task optimization algorithms enable collaborative optimization of multiple concurrent tasks [7, 17]. By facilitating interactions and knowledge sharing between the tasks, overall performance can be enhanced. Multi-task optimization algorithms have a wide range of applications. They have the potential to improve system performance. This is achieved by enhancing computational efficiency and accelerating the learning process. Consequently, this paper opts for an MTMO algorithm to optimize the scheduling of two task types in a multi-cloud environment. Through multitask evolutionary optimization, these two tasks can learn from each other's experience, learn useful knowledge, and adjust

their evolutionary process as a way to improve scheduling performance.

The main contributions of this paper are as follows:

1. Aimed at addressing the issue of multi-task scheduling optimization in multi-cloud environments, this paper constructs a multi-task multi-objective model. According to the characteristics of CPU-intensive and I/O-intensive tasks, a multi-objective optimization model is constructed for each of the two different types of tasks by considering multiple optimization objectives, and the two multi-objective optimization models are used as subtasks to construct a scheduling model of multi-task multi-objective on MCE;

2. To augment the competency of the multi-objective optimization strategy, a quadratic crossover-based multi-objective multi-factor evolutionary algorithm I-MOMFEA-II is introduced in this paper. The algorithm can retain the superior gene bits in the parent in each crossover, thus improving the convergence and performance of the algorithm. The I-MOMFEA-II algorithm shows better performance on the CEC17-MTMO test set by conducting comparative experiments with other algorithms;

3. The simulation experiments conducted using the I-MOMFEA-II algorithm to optimize the multi-task multi-objective scheduling model proposed in this paper demonstrate its effectiveness in a multi-cloud environment. The results indicate that I-MOMFEA-II can significantly enhance the scheduling efficiency of concurrently managing the two types of tasks in MCE.

The remainder of this paper is organized as follows: Section 2 reviews relevant literature and existing methodologies; Section 3 delineates the scheduling models for two types of tasks within a multi-cloud environment and provides a detailed description of the quadratic crossover-based multi-objective multi-factor evolutionary algorithm I-MOMFEA-II; Next, Section 4 validates the efficacy of I-MOMFEA-II through experimental analyses and benchmarking; Finally, Section 5 concludes this paper and discussing potential avenues for future research.

2. Related work

This section begins by reviewing pertinent research on task scheduling in multi-cloud environments. We then delineate the evolutionary multitasking algorithm utilized herein, along with its real-world applications.

2.1. Task scheduling in multi-cloud environments

In recent years, task scheduling for multi-cloud environments has garnered significant attention from scholars. These researchers have explored task scheduling problems in multi-cloud environments from various perspectives. For instance, to optimize for security and reliability, Zhu et al. [36] to optimize for security and reliability. This approach takes into account task priority sequences and aims to meet related needs. Additionally, a multi-round allocation strategy is employed to enhance scheduling performance and minimize the time objective. Experimental results demonstrate the algorithm's effectiveness in reducing completion time and cost savings. Kang et al. [15] introduced a segmented multi-round scheduling method to address the goals of achieving high utilization of computing nodes and load balancing in multi-cloud environments, considering both static and dynamic [28] aspects. The method was validated to be effective through simulated

evaluations. Hubert Shanthan et al. [14] partitioned the multi-cloud scheduling process into two divisions: the scheduling portion and the rescheduling portion. The scheduling stage utilizes a cyclic scheduling algorithm, while the two-stage scheduling achieves optimization of execution costs. SAMBIT et al. [20] proposed an energy-aware multi-cloud network task scheduling algorithm. In the first stage of execution, it selects virtual machines based on the expected completion time (ETC) matrix using deadlines. In the second stage, through the energy consumption (EC) matrix, it selects a VM that has enough resources to execute the task and uses fewer resources when the task is executed among all available VMs, this optimizes the schedule to minimize total energy consumption while reducing system overhead. Roy et al. [24] concluded that large-scale tasks are better handled using multi-cloud environments. They proposed a scheduling algorithm that combines task priorities to minimize maximum completion time, task execution cost, energy consumption, and delay as optimization objectives. The algorithm demonstrates satisfactory capabilities based on simulation experiments. Mazen Farid et al. [10] proposed a weighted adaptive inertia algorithm for optimizing multiple objectives during scheduling in a multi-cloud environment. They use the minimum weight algorithm to derive the Pareto frontier. They generate alternative weights using the weighted adaptive inertia algorithm to determine the inertia weights, achieving the effect of selecting the best trade-off arrangement among multiple conflicting objectives. Cai et al. [4] constructed a multi-objective distributed scheduling model based on the IoT, multi-cloud, and task scheduling. Accordingly, they developed a sine function-based multi-objective algorithm to implement the model. Experiments indicate the algorithm can schedule tasks with good efficiency. To enhance multi-cloud scheduling efficiency, Hao et al. [13] developed a model to process parallel tasks in multi-cloud environments. As they predicted more parallel tasks with increased virtual machines, they proposed a multi-objective algorithm optimizing job wait times across queues. Experimental results have validated the efficacy of this method.

However, in the above research, some researchers only consider the optimization objectives from the perspective of users or service providers, without taking into account the interests of both parties. In addition, these studies have low parallelism in processing tasks and cannot handle multiple tasks simultaneously. Furthermore, most of the above research ignores the differences in task data characteristics and resource demand differences in a multi-cloud environment. This means that regardless of the differences in data characteristics between tasks or the differences in resource requirements, a unified model is established to handle tasks. However, when the resource requirements between tasks differ greatly, system performance will be affected.

2.2. Multi-tasking evolutionary algorithm

In 2016, Gupta et al. [11] originally conceived the notion of multi-task evolution. Its essential goal is to concurrently solve multiple optimization challenges by exploiting their implicit parallelism, thereby reducing execution times and improving performance across heterogeneous environments. The unified search space containing multiple task design spaces is used to achieve implicit parallel processing. On this basis, Gupta et al. [11] pioneered the development of the Multi-Factor Evolutionary Algorithm (MFEA). Subsequently, many improved and derived algorithms

based on the MFEA algorithm have been proposed and applied in various research fields. As an example, in the constrained vehicle routing problem with combination optimization, Zhou et al. [34] developed a PMFEA algorithm integrating a unified permutation-based representation and separation-driven decoding. By improving the information sharing between tasks and the decoding operation, this algorithm improves the performance of multi-task evolutionary algorithms in solving vehicle routing problems. ThiThanh et al. [27] proposed a multi-task evolutionary algorithm to solve the shortest path tree clustering problem and introduced a new decoding scheme. This scheme can decode the solutions in the unified space to the corresponding tasks to obtain better solutions. For classification problems, Tang et al. [26] proposed MTM-ELM, a multi-task evolutionary algorithm. It trains multi-pole learning models with different numbers of hidden neurons to improve classification accuracy. Liang et al. [22] developed an algorithm called MFEA-VMP to solve the virtual machine placement problem in cloud computing environments. They introduced greedy placement operators as well as remigration and merger algorithms to solve the virtual machine placement problem, thereby improving resource utilization efficiency in cloud computing environments. In power dispatch, Liu et al. [23] introduced a multi-objective multi-factor evolutionary algorithm and combined it with the characteristics of the power system to propose a multi-task power dispatch method. This method can handle multiple objectives of multiple tasks in parallel to improve dispatch efficiency. In addition, Zhao et al. [33] modeled the multi-objective cloud task scheduling problem of the cloud environment and introduced multi-factor optimization algorithms to handle large-scale multi-objective cloud task scheduling problems.

These research works show that multi-task evolutionary algorithms have shown wide application and potential in optimization problems in various fields. By utilizing the inter-task information sharing and knowledge transfer mechanisms in multi-task optimization algorithms, the two types of tasks within a multi-cloud environment can learn useful knowledge from each other, thereby accelerating the scheduling efficiency of CPU-intensive tasks and I/O-intensive tasks, and improving the system performance across the multi-cloud environment. In summary, multi-task optimization algorithms can provide better solutions and optimization results for solving task scheduling problems in the MCE.

3. The proposed method

In the MCE, users can execute submitted tasks through multiple cloud providers. MCE mitigates vendor lock-in and strengthens data security while expanding user choices and reducing costs. However, due to the different data characteristics of tasks within the multi-cloud environment, the requirements of tasks will differ. Ignoring these differences will cause the resources allocated to tasks to mismatch the resources they need when there are large differences in data characteristics between tasks. This will not only cause unnecessary waste of resources but also reduce system performance. This section proposes a multi-task multi-objective scheduling model in MCE. According to the different data characteristics, the multi-cloud environment divides the tasks into I/O-intensive tasks with higher demands for I/O resources and CPU-intensive tasks with higher demands for CPU through Logistic regression. According to the requirements of these two types of tasks, two multi-objective optimization models are constructed. Then, these two multi-objective optimization

models are used as two sub-tasks in the multi-task optimization environment, and a one-time scheduling is performed using the multi-task optimization algorithm.

3.1. Classification method

In this paper, the task is classified using the logistic regression algorithm, which is a common classification algorithm used to solve binary classification problems. It estimates the relationship between input variables and output variables by building a logistic regression model. The essential idea of logistic regression is to map a linear combination of input features by a nonlinear function called a "sigmoid function" to a probability value that denotes the chance of a sample getting designated to a specific category. Samples with probabilities greater than or equal to a set threshold are usually classified as positive classes and those with probabilities less than the threshold are classified as negative classes.

The sigmoid function for logistic regression is expressed as follows:

$$\sigma(z) = (1 + e^{-z})^{-1}. \tag{3.1}$$

It is known from previous studies that the two factors affecting task classifications are the task data size and computational complexity. Thus, the linear combination z of input features in the sigmoid function can be expressed as:

$$z = w_0 DataSize + w_1 Complexity. \tag{3.2}$$

The probability of a classification result of 1 (I/O-intensive task) and 0 (CPU-intensive task) can be respectively:

$$\begin{aligned} P(y = 1|x; w) &= hw(x), \\ P(y = 0|x; w) &= 1 - hw(x). \end{aligned} \tag{3.3}$$

Where hw(x) is the value of a linear regression obtained by regression calculation after inputting the data size and computational complexity of the task, and then the logistic regression result is obtained by mapping the activation function, and hw(x) represents the value of this result.

In logistic regression, optimization is driven by a log-likelihood loss function and is represented by the following log-likelihood function: where M denotes the total of tasks to be classified and y(i) denotes the current classification result of logistic regression:

$$f(w) = \sum_{i=1}^{M} [y^{(i)} \log(hw(x^{(i)})) + (1 - y^{(i)}) \log(1 - hw(x^{(i)}))]. \tag{3.4}$$

To adjust the regression coefficients w0, and w1 of the classification function, we leverage gradient descent as an optimization technique to diminish the loss function. The gradient is denoted as ∇. The gradient of the function is then expressed as follows:

$$\nabla f(w) = \begin{pmatrix} \frac{\partial f(w)}{\partial w_0} \\ \frac{\partial f(w)}{\partial w_1} \end{pmatrix}. \tag{3.5}$$

The step size of the move is α, and the iterative formula for gradient descent is shown below:

$$w := w - \alpha \nabla f(w)(w = [w_0, w_1]^T). \tag{3.6}$$

After determining the regression coefficients, a threshold value needs to be set to determine the task type corresponding to the Sigmoid function value, which is settled on 0.5 within this paper. Tasks are classified as I/O-intensive when the value meets or exceeds the threshold, otherwise they are categorized as CPU-intensive.

3.2. Multi-objective multi-task scheduling model

In MCE, time and cost are the two most important metrics. Figure 1 illustrates the general flow of a user submitting a task in a MCE. In the MCE, each cloud is independent and does not share resources. Users submit tasks to the cloud database. Then, the data center produces the optimal scheduling scheme based on the optimization objectives and limitations specified by users, accessible resources from cloud providers, and the chosen scheduling algorithm. These scheduling schemes consist of a series of VM numbers, indicating the execution order and location of tasks on different cloud resources.

Since there are differences in the data characteristics of tasks submitted by users to the cloud, there are also differences in the resource requirements. For the characteristics of I/O-intensive tasks and CPU-intensive tasks, this paper considers the two objectives of task completion time and completion cost and considers the objectives corresponding to their data characteristics separately for different types of tasks. When handling I/O-intensive tasks, our objective is to optimize virtual machine throughput, thereby maximizing the number of tasks fulfilled over a given timeframe on each VM. For CPU-intensive tasks, we consider one of the CPU energy-cost targets to optimize the high energy consumption problem due to the large amount of computation.

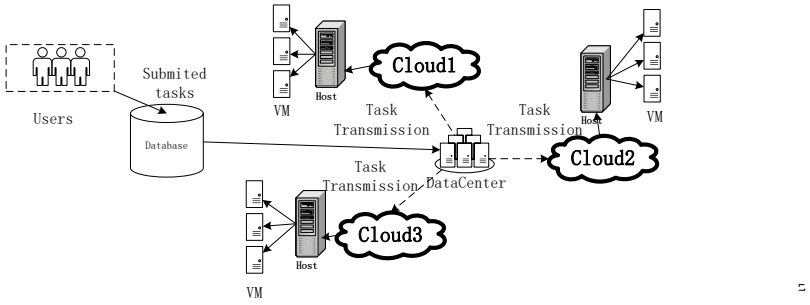

Figure 1. Task scheduling process in a multi-cloud environment

1) Task Execution Time Model

Task execution time includes task computation time and task transfer time. Task computation time is affected by CPU computation power Ck and task size TCi, whereas the time required for task transfer correlates with the data volume TLi necessary for completing the task and the transfer bandwidth Lk available.

Therefore, the expected completion time of task Ti when allocated on virtual machine VMk can be formulated as follows:

$$CTime(T_i, VM_k) = \frac{TC_i}{C_k} + \frac{TL_i}{L_k}, \tag{3.7}$$

where TCi/Ck denotes the computation time required for task Ti to execute on the kth virtual machine on the cloud, and TLi/Lk denotes the transmission time required for task Ti to execute on the kth task on the cloud, subject to the deadline, which should be less than the task's deadline TDi, that is CTime $\leq$ TDi;

Then the total task execution time is shown in Eq. (3.7), the variable i represents the task number, while N denotes the total number of tasks:

$$TotalTime = \sum_{\mathrm{i}=1}^{\mathrm{N}} CTime(T_i, VM_k). \tag{3.8}$$

2) Task Completion Cost Model

Task completion cost includes computation cost, transmission cost, and storage cost. The computational cost of task Ti on VMk is related to the task size TCi and the unit cost Pk of task computation on VMk, and the transfer cost correlates to the data size TLi requiring transmission for task completion, and the per-unit transfer cost Qk on virtual machine VMk. Storage costs are incurred as the data needs to be stored after the transfer of the data required for the task is completed. Storage costs are related to the size of the data to be transferred for the task execution TLi and the unit cost of data storage on the VMk of the virtual machine Sk. Thus, the execution cost of the task Ti on the VMk is represented as follows:

$$Task\mathrm{Cost}(T_i, \mathrm{VM_k}) = TC_i \times P_k + TL_i \times Q_k + TL_i \times S_k, \tag{3.9}$$

where TCi×Pk denotes the computational cost required for task Ti to be executed on the kth virtual machine in the cloud; TLi×Qk denotes the transfer cost required for task Ti to be executed on the kth virtual machine in the cloud; TLi×Sk denotes the storage cost required for task Ti to be executed on the kth virtual machine in the cloud, Subject to the cost budget, the task execution cost should be less than the budgeted cost TMi, that is TaskCost $\leq$ TMi.

Then the total task completion cost can be expressed as:

$$Cost = \sum_{i=1}^{\mathrm{N}} TaskCost(Ti, VM_k). \tag{3.10}$$

3) VM Throughput Model

This target is specific to I/O-intensive tasks. When performing I/O-intensive tasks, most of the time is spent waiting for I/O operations to complete, so the CPU will be in a longer waiting state, resulting in less energy consumption generation. In this case, the optimization of CPU power consumption can be ignored for the time being. However, there are memory contention problems due to more tasks performing frequent I/O operations, which can lead to longer task execution times, making a virtual machine serve one task for a long time. Therefore, this paper takes the number of tasks completed on VMk per unit of time as the throughput of VMk and reduces the possibility of the above problem by optimizing this objective.

The throughput of each VMk is expressed as the ratio of the number of Countk tasks executed on VMk to the total time required for the execution of Countk tasks, expressed as follows:

$$VMThroughput(VM_k) = \frac{Count_k}{\sum_1^{countk} CTime(T_i, VM_k)}. \tag{3.11}$$

Then the average throughput of the virtual machine can be expressed as:

$$Throughput = \frac{1}{K}\sum_{i=1}^{K} VMThroughput(VM_K). \tag{3.12}$$

4) CPU Energy Consumption Model

This target is specific to CPU-intensive tasks. CPU-intensive tasks require high CPU performance and high computation volume, resulting in long CPU computation time. During task execution, the CPU is almost always under high load, so energy consumption is relatively high. For CPU resources, most of the time is spent on performing task calculations, while CPU idle time is negligible. In this case, CPU-intensive tasks are more concerned with optimizing energy consumption compared to the throughput of virtual machines. Therefore, this paper specifically considers the energy consumption target for CPU-intensive tasks.

The energy consumption generated by the task Ti when it is executed on the resource VMk is related to the task execution time and the energy consumption Vk generated per unit time. The task execution time is represented by the ratio of the task size TCi to the CPU computing power Ck, then the energy consumption of the task Ti executing on the resource VMk is expressed as follows:

$$EC(T_i, VM_{\mathrm{K}}) = \frac{TC_i}{C_k} \times V_k. \tag{3.13}$$

Then the total energy consumption of the virtual machine can be expressed as:

$$EnergyConsumption = \sum_{\mathrm{i}=1}^{\mathrm{K}} EC(T_i, VM_K). \tag{3.14}$$

In summary, the scheduling model for multi-objective multitasking is constructed as follows:

$$\begin{cases} I/O - Intensive - task - model \begin{cases} TotalTime = \sum\limits_{i=1}^{N} CTime(T_i, VM_k), \\ Cost = \sum\limits_{i=1}^{N} Task\mathrm{Cost}(T_i, VM_k), \\ Throughput = \frac{1}{K}\sum\limits_{i=1}^{K} VMThroughput(VM_k), \end{cases} \\ CPU - Intensive - task - model \begin{cases} TotalTime = \sum\limits_{i=1}^{N} CTime(T_i, VM_k), \\ Cost = \sum\limits_{i=1}^{N} Task\mathrm{Cost}(T_i, VM_k), \\ EnergyConsumption = \sum\limits_{i=1}^{K} EC(T_i, VM_K). \end{cases} \end{cases} \tag{3.15}$$

3.3. Multi-objective multi-factor evolutionary algorithm based on the quadratic crossover (I-MOMFEA-II)

The I-MOMFEA-II algorithm builds upon the MOMFEA-II algorithm to address MTMO problems. In the I-MO-MFEA-II algorithm, each multi-objective problem is regarded as a sub-task, and multiple sub-tasks are solved uniformly by one population. Using the vertical culture and mating selection transfer mechanism of MO-MFEA-II, tasks can communicate knowledge with each other and share information in tasks to achieve better optimization results. To make the algorithm converge faster and better, this paper introduces quadratic crossover. That is, after completing the simulated binary crossover, cross over again at two points between offspring and parents to ensure that the offspring retains more excellent gene locations from the parents. The pseudocode for the I-MOMFEA-II algorithm is presented in Algorithm 1.

3.3.1. Encoding scheme

The issue of task scheduling within MCE adopts real number encoding, the specific encoding form is depicted in Figure 2, where individual encodings represent the mapping between tasks and virtual machines. In Figure 2, tasks t1, t4, and t6 are executed on VM2, tasks t2 and t3 are executed on VM1, tasks t5 and tN are executed on VM3, and task t7 is executed on VMK.

Since this paper classifies tasks before task scheduling, the quantity of the two task varieties may diverge, meaning the dimensions of the decision variables of the two types of tasks may be different. As shown in Figure 3, assuming that the dimension of the first type of task T1 is Dt, and the dimension of the second type of task T2 is Dk, if Dt, then the dimension of the decision variable when multi-task optimization processes these two types of tasks simultaneously are Dk, take the first t dimension when decoding task T1.

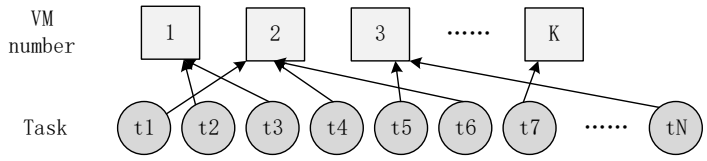

Figure 2. Individual encoding of the cloud task scheduling model in the multi-cloud environment

3.3.2. Selective mating

In the I-MOMFEA-II algorithm, mating between parents is not completely random. Individuals preferentially mate with others sharing the same skill factors, while inter-cultural mating probabilities are dictated by the random mating probability matrix (RMP). In multi-task optimization, assuming K optimization tasks are given, RMP is a K×K dimensional matrix. Matrix elements represent inter-task knowledge transfer probabilities. This probability can be determined based on factors such as the similarity between sub-tasks, the importance of tasks, and the fitness of solutions. By adjusting the elements of the RMP matrix, knowledge

Algorithm1: I-MOMFEA-II algorithm

1. Initialize P(0)
2. for each pi ∈ P(0)
3. τi = mod (i, K) + 1 //τi is skill factor, K is the total number of tasks
4. Evaluate pi on task τi
5. Φi is computed through ranking based on NFi and CDi. // Φi is scalar fitness
6. t = 1
7. While stopping conditions not met
8. Pm(t)= ∅
9. Select n·K parents from P(t) using Tournament Selection
10. Learn RMP(t)
11. While offspring≤ N+1 per task:
12. Randomly select qi, qj from P(t)
13. if τi == τj
14. Intra-task crossover and mutate (qi, qj) to get [qa, qb] with skill τi
15. else if rand ≤ rmpτi,τj
16. Inter-task crossover and mutate (qi, qj) to get [qa, qb] with random τi or τj
17. else
18. Select qi with τi and qj with τj
19. Mutate and Intra-task crossover each to get [qa, qb] with respective skills
20. Evaluate [qa, qb] on assigned tasks only
21. Pm(t) = Pm(t)∪ [qa, qb]
22. C(t) = Pm(t)∪ P(t)
23. Update Φi for each pi∈ C(t)
24. Select top N·K fittest from C(t) as P(t+1)
25. t = t + 1

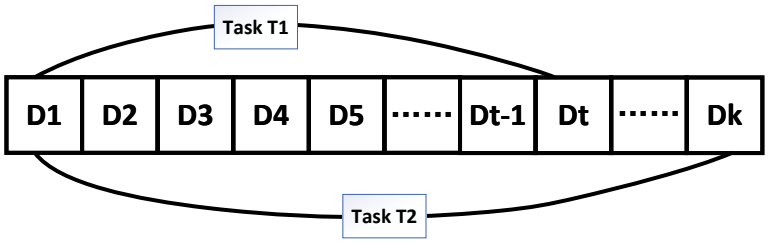

Figure 3. Multi-task individual encoding

transfer intensity between sub-tasks is controllable. Higher probability values indicate that knowledge is more easily transmitted between two sub-tasks, while lower probability values indicate less possibility of knowledge transfer. The MOMFEA-II algorithm realizes selective mating between individuals by introducing the RMP

matrix. Its specific representation is shown in formula (3.15):

$$\mathrm{RMP}\begin{bmatrix} rmp_{1,1}\ rmp_{1,2}\ . . \\ rmp_{2,1}\ rmp_{2,2}\ . . \\ .\quad .\quad . . \\ .\quad .\quad . . \end{bmatrix}. \tag{3.16}$$

Where rmpi,j represents the probability that sub-task i transfers knowledge to sub-task j, and rmpk,k=1.To make the algorithm converge quickly, this paper introduces a crossover strategy based on a quadratic crossover based on selective mating in the original MOMFEA-II algorithm. In the multi-task optimization algorithm, the population is implicitly segmented into multiple sub-populations per skill factor, each sub-population optimizes a subtask, and multiple similar tasks are optimized simultaneously. During crossover, there are intra-population crossover and inter-population crossover. Intra-population crossover refers to two-parent individuals from the same sub-population, and the offspring also belong to the same sub-population; inter-population crossover refers to two-parent individuals from different sub-populations and the offspring will be randomly assigned to any one of the sub-populations to which the two parents belong.

First, a simulated binary crossover operator is used for the first crossover. In simulated binary crossover, the size of the crossover distribution index parameter affects the result of the crossover operation. When the crossover distribution index parameter is large, the similarity between the offspring and the parents is higher. To reduce the negative effect of negative transfer, this paper adopts different crossover distribution index parameters in intra-population crossover and inter-population crossover. The crossover distribution index parameter of inter-population crossover is smaller so that it can reduce the negative impact of one type of task on the subpopulation to which the current offspring belongs when the similarity between the two task types is scant.

Secondly, the second crossover is performed using a two-point crossover. The offspring generated by the first crossover again with one of their two parent individuals. In the intra-population crossover, each offspring randomly selects a parent individual and randomly selects one to two gene bits from that parent individual to exchange with the gene bits at the corresponding positions of the offspring. In inter-population crossover, each offspring only exchanges one gene bit with a parent individual. This crossover strategy enhances population diversity and boosts the search capability of the population. When optimizing multiple similar tasks simultaneously, the quadratic crossover operator, shown in Algorithm 2, effectively reduces negative transfer impact.

4. Simulation experiment

To assess I-MOMFEA-II's performance across different task scheduling models in MCE, we first tested it against other algorithms using two types of models in an MCE. Then, the performance of I-MOMFEA-II is evaluated through comparative experiments with other algorithms on the CEC17-MTMO [31] benchmark suite, demonstrating its capabilities in multi-task multi-objective optimization.

Algorithm2: Crossover Operator Based on Quadratic Crossover

Input:

parent individuals pa and pb;

Inter-population crossover distribution index parameter:mu-between;

Intra-population crossover distribution index parameter:mu-in;

Output:

Offspring individuals ca and cb;

1. if τa==τb // Intra-task crossover
2. (ca1,cb1)=SBX(pa,pb,mu-in);
3. Generate two random numbers:s1,e1,and e1 $\leq$ s1+2;
4. if rand() $\geq$ 0.5
5. ca=Two-Point-Cross(ca1,pa,s1,e1);
6. else
7. ca=Two-Point-Cross(ca1,pb,s1,e1);
8. The cb undergoes the same steps as the ca to complete the crossover.
9. if τa$\neq$$\tau$b // Inter-task crossover
10. (ca1,cb1)=SBX(pa,pb,mu-between);
11. Generate two random numbers:s2,e2 and e2 $\leq$ s2+1;
12. if rand() $\geq$ 0.5
13. ca=Two-Point-Cross(ca1,pa,s2,e2);
14. else
15. ca=Two-Point-Cross(ca1,pb,s2,e2);
16. The cb undergoes the same steps as the ca to complete the crossover.

4.1. I-MOMFEA-II experiments to solve two types of task scheduling models in multi-cloud environments

In this paper, the I-MOMFEA-II algorithm is compared with the MO-MFEA-II [3] algorithm, the MO-MFEA algorithm [12], the EMT-PD algorithm [21], and the AMT-NSGA-II [6] algorithm. To reflect multi-cloud characteristics, three different clouds were set up in the simulations, differing primarily in execution speed and cost. Cloud 1 has lower expenses but a slower speed, ideal for small tasks. Cloud 2 has medium speed and cost. Cloud 3 is faster but costlier, suitable for larger tasks. The specific cloud parameters are shown in Table 1. We initialized 300 tasks, each with 500 million instructions (MI), a 200KB file size, and a 100KB output file. The length, file size and output size of each task were then gradually increased by 500MI, 10KB, and 10KB respectively.

In order to test the effectiveness of the proposed model in this paper, the developed algorithm and model were utilized to schedule tasks in MCE using 100, 300 and 600 cloud tasks scenarios. As we obtained a range of solutions for each of the two types of tasks, we compared the objective values of the two types of tasks across different scenarios, and evaluated the optimal, maximum and average values to determine the performance of the objective values. The optimal and average val-

ues give a good indication of the algorithm's convergence on the multi-cloud model, whereas the maximum value reflects the diversity of the algorithm.

Table 1. Multi-cloud environment simulation parameters settings.

Cloud provider	Execution Speed	Transmission	Bandwidth	Cost
Cloud1	300-500 MIPS	0.015 $/GB	1024-2048 Mbps	0.03 $/h
Cloud2	500-1000 MIPS	0.030 $/GB	2048-3072 Mbps	0.06 $/h
Cloud3	1500-2000 MIPS	0.045 $/GB	3072-4096 Mbps	0.09 $/h

As can be seen from Table 2, the I-MOMFEA-II algorithm performs best on the three objectives of CPU-intensive tasks, with the minimum completion cost, completion time, and energy consumption. However, in terms of the maximum values of the three objectives, there is a slight underperformance in the efficacy of the I-MOMFEA-II approach relative to alternative techniques.

From Table 3, it can be seen that under three different task quantities, the I-MOMFEA-II algorithm performs best on the three objectives of I/O-intensive tasks. When the number of tasks is 100 and 300, the I-MOMFEA-II algorithm achieves good results in the mean and minimum values of the three objectives: completion cost, completion time, and virtual machine throughput. When the number of tasks is 600, the I-MOMFEA-II algorithm also performs well in the mean values of the three objectives: completion cost, completion time, and virtual machine throughput, but in other cases, the performance of the I-MOMFEA-II algorithm is slightly inferior to other algorithms.

Figure 4 shows the convergence curves of the I-MOMFEA-II algorithm on three target values for CPU-intensive tasks under the condition that the number of cloud tasks is 300 and the number of iterations is 500. Among them, the first figure of Figure 4 presents the convergence of the total cost of CPU-intensive tasks, the second figure of Figure 4 demonstrates the convergence of the total energy consumption of CPU-intensive tasks, and the third figure of Figure 4 demonstrates the convergence of the total time of CPU-intensive tasks. By looking at these three graphs, it can be concluded that the I-MOMFEA-II algorithm shows good convergence on all objectives for CPU-intensive tasks.

Figure 5 displays the convergence curves of the I-MOMFEA-II algorithm on the three target values for IO-intensive tasks at a cloud task count of 300 and 500 iterations. Among them, the first figure of Figure 5 presents the convergence of the total completion cost of IO-intensive tasks; the second figure of Figure 5 presents the convergence of the throughput of the IO-intensive task virtual machine; the third figure of Figure 5 presents the convergence of the total execution time of IO-intensive tasks. The results show that while the I-MOMFEA-II algorithm performs on par with the MOMFEA-II algorithm in terms of the VM throughput objective of the IO-intensive task, it has good convergence on the other two objectives.

The above experiments demonstrate the superior optimization capabilities of the I-MOMFEA-II algorithm in concurrently scheduling both task varieties across MCE. These experimental results provide strong support for the application of the I-MOMFEA-II algorithm to solve CPU-intensive and IO-intensive task scheduling problems.

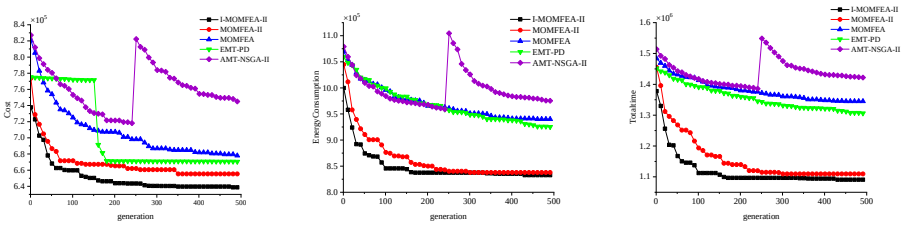

Figure 4. CPU-intensive task target convergence curve

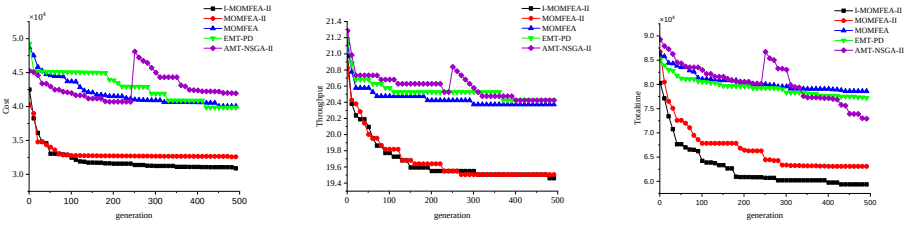

Figure 5. IO-intensive task target convergence curve

4.2. Performance experiments of the I-MOMFEA-II algorithm

In this paper, the problem set CEC17-MTMO based on multi-task multi-objective optimization is used as the test set. The CEC17-MTMO problem set contains tasks from several different domains with different objective functions and constraints. These tasks may have properties that affect or compete with each other, making it necessary for the algorithm to balance and trade off between multiple tasks, Table 4 details the features of the CEC17-MTMO test set. In the experiments, the maximum number of iterations is set to 20,000 for all algorithms. In addition, the genetic operation uses the quadratic crossover-based crossover strategy and polynomial variation proposed in this paper. In the quadratic crossover-based crossover strategy, the within-population crossover distribution index parameter mu-in=18, the between-population crossover distribution index parameter mu-between=16, and the polynomial variation probability pc=1/D.

Two commonly used performance metrics for evaluating multi-objective intelligent algorithms are IGD [38] (Inverse Generation Distance) and HV [37] (Hypervolume). A smaller IGD value indicates the algorithm's solution set is closer to the true frontier, signifying better convergence and diversity. The IGD can be calculated as:

$$IGD(\mathrm{U},\mathrm{W}) = \frac{1}{|\mathrm{W}|}\sum_{i=1}^{|\mathrm{w}|}\min_{j=1}^{|u|} d(w_i,u_j). \tag{4.1}$$

The Euclidean distance d(wi,uj)=‖wi-ui‖2 represents the distance between reference point wi and individual ui. A sufficiently large reference set $|W|$ can adequately represent the PF by evenly distributing along it in objective space. Lower IGD values require individual ui to closely approach the entire PF without gaps. Thus, IGD measures individual diversity and convergence to the PF to some degree.

Table 5 shows the comparative results of IGD values for six algorithms such as I-MOMFEA-II on the CEC17-MTMO test set. Table 5 presents algorithm results for

Table 2. Numerical analysis of the three objectives of different algorithms for CPU-intensive tasks in different task number scenarios

Number	Indicator	Algorithm	Cost	Totaltime	EnergyCost
100	Best	I-MO-MFEA-II	21179.72894	**88284.25423**	**75631.105**
		MO-MFEA-II	19940.38307	111757.8566	88735.275
		MO-MFEA	23875.55307	128728.439	94879.68
		EMT-PD	**19723.43973**	134386.4681	98417.94
		AMT-NSGA-II	28395.99979	145586.2715	104945.7
100	Avg	I-MO-MFEA-II	**31142.64415**	**168429.1646**	**111633.3242**
		MO-MFEA-II	32348.71931	171441.2934	116901.8965
		MO-MFEA	32259.67516	175290.548	117928.5396
		EMT-PD	32142.44594	173416.7845	118332.3375
		AMT-NSGA-II	32574.49725	168599.0866	116222.116
100	Worst	I-MO-MFEA-II	42078.34615	237359.4439	145846
		MO-MFEA-II	45879.91768	242683.2795	144876.48
		MO-MFEA	40146.47789	221099.7724	139278.28
		EMT-PD	39136.38272	252257.0596	150850.06
		AMT-NSGA-II	**36851.97372**	**191919.3569**	**127946.495**
300	Best	I-MO-MFEA-II	**638630.051**	**1090622.473**	**832673.975**
		MO-MFEA-II	655320.4023	1109395.444	837881.81
		MO-MFEA	677970.674	1345108.055	940069.14
		EMT-PD	670555.5406	1306208.076	925187.48
		AMT-NSGA-II	745073.983	1422042.322	975372.625
300	Avg	I-MO-MFEA-II	**805204.0897**	**1448079.981**	**876880.2925**
		MO-MFEA-II	851781.0197	1476530.245	958920.18
		MO-MFEA	839696.4128	1598269.064	1024713.888
		EMT-PD	841926.6476	1816515.319	1034011.113
		AMT-NSGA-II	8216473.4982	1569904.877	1045709.05
300	Worst	I-MO-MFEA-II	1047031.03	1805537.49	**1150850.72**
		MO-MFEA-II	1054619.114	1843665.045	1166835.1
		MO-MFEA	937010.2162	1851430.073	1192559.15
		EMT-PD	977439.7797	2326822.562	1378464.26
		AMT-NSGA-II	**899860.8315**	**1717767.433**	1127171.395
600	Best	I-MO-MFEA-II	**930623.7063**	**1490538.112**	**1097135.47**
		MO-MFEA-II	975778.2003	1508011.374	1102930.11
		MO-MFEA	1090956.648	1646311.229	1163331.655
		EMT-PD	975958.2079	1711638.441	1199973.335
		AMT-NSGA-II	1104343.322	1734535.297	1214935.98
600	Avg	I-MO-MFEA-II	**1143712.812**	**1831982.893**	**1244669.67**
		MO-MFEA-II	1193470.461	1889552.957	1265711.818
		MO-MFEA	1237128.824	1872127.675	1279238.934
		EMT-PD	1210836.164	1928431.128	1309618.757
		AMT-NSGA-II	1217746.142	1931142.884	1315836.338
600	Worst	I-MO-MFEA-II	1403211.894	2181834.984	**1402274.345**
		MO-MFEA-II	1436035.36	2243946.812	1423787.155
		MO-MFEA	1388993.843	**2141106.381**	1419295.44
		EMT-PD	1346440.367	2917590.421	1721938.02
		AMT-NSGA-II	**1340369.111**	2146659.571	1426468.975

Table 3. Numerical analysis of the three objectives of different algorithms for IO-intensive tasks in different task number scenarios

Number	Indicator	Algorithm	Cost	Totaltime	Throughput
100	Best	I-MO-MFEA-II	**2890.238864**	**12143.51318**	**5.825808331**
		MO-MFEA-II	3102.704178	13955.08161	5.85994726
		MO-MFEA	3303.555609	16684.19355	6.220839813
		EMT-PD	2893.731999	17017.18024	6.289308176
		AMT-NSGA-II	3344.3125	18998.68991	6.226650062
100	Avg	I-MO-MFEA-II	**4418.786534**	**24878.21412**	**6.953420237**
		MO-MFEA-II	4789.641631	25646.82982	7.497919156
		MO-MFEA	4601.182884	27155.37879	7.478445402
		EMT-PD	4478.31604	28457.18911	7.640473168
		AMT-NSGA-II	4582.58134	27712.28615	7.047596744
100	Worst	I-MO-MFEA-II	6787.610974	**35855.98145**	8.587376556
		MO-MFEA-II	6932.383415	36976.44453	9.474182852
		MO-MFEA	6204.38716	38248.96081	8.729812309
		EMT-PD	6180.329367	40956.99642	9.358914366
		AMT-NSGA-II	**6058.295301**	36122.05033	**8.025682183**
300	Best	I-MO-MFEA-II	**30875.4249**	**59356.0348**	**19.45979606**
		MO-MFEA-II	32585.94356	63085.05889	19.50382275
		MO-MFEA	40050.49861	78604.22959	20.37334057
		EMT-PD	39851.62454	77198.88864	20.42400402
		AMT-NSGA-II	41943.68086	72935.37292	20.42400402
300	Avg	I-MO-MFEA-II	**47270.5969**	**97420.48162**	**20.95571817**
		MO-MFEA-II	47950.06081	99261.43062	21.27204433
		MO-MFEA	50069.66806	100516.5661	21.99025778
		EMT-PD	47276.70667	104776.0142	22.97535003
		AMT-NSGA-II	52135.45089	97665.03331	22.00189323
300	Worst	I-MO-MFEA-II	67872.1387	130135.2967	**22.9273661**
		MO-MFEA-II	71010.98319	131613.4131	23.88002675
		MO-MFEA	**60945.13684**	123166.9512	23.11176851
		EMT-PD	62232.44941	141840.25	26.93965517
		AMT-NSGA-II	65473.05155	**119054.7381**	23.48961759
600	Best	I-MO-MFEA-II	3540345.12	**5312295.839**	18.86080724
		MO-MFEA-II	3555358.71	5484956.443	19.94893074
		MO-MFEA	3960619.051	6410807.644	27.47932181
		EMT-PD	**3476506.099**	6409351.036	**17.68922811**
		AMT-NSGA-II	4169851.122	6657695.974	29.42820988
600	Avg	I-MO-MFEA-II	**4162167.227**	**6497333.099**	**25.43393297**
		MO-MFEA-II	4472127.164	6669290.767	26.79803548
		MO-MFEA	4461393.272	7341211.767	39.53459907
		EMT-PD	4371861.671	7894403.262	37.68551485
		AMT-NSGA-II	4622232.663	7444146.455	40.06392349
600	Worst	I-MO-MFEA-II	5003848.539	**8077400.911**	**35.77049649**
		MO-MFEA-II	5315973.197	8254824.852	37.38178012
		MO-MFEA	**4986573.076**	8443009.884	49.3973523
		EMT-PD	5022926.84	11536508.27	52.523767
		AMT-NSGA-II	5251084.323	8681919.372	52.523767

Table 4. Multi-cloud environment simulation parameters settings.

CEC17-MTMO Test Set Features
1.Multi-tasking: The problems in the CEC17-MTMO test set contain multiple related optimization tasks, and there may be dependent or conflicting relationships between these tasks. The algorithm needs to consider the optimization objectives of multiple tasks simultaneously.
2.Multi-objective: The problems in the CEC17-MTMO test set involve multiple competing objective functions and require finding a balance point or achieving an optimal solution set between different objectives.
3.Complexity: The problems in the CEC17-MTMO test set usually have complex nonlinear characteristics, high dimensionality, and multimodal properties, making the solution space have multiple locally optimal solutions and globally optimal solutions.
4.Diversity: The problems in the CEC17-MTMO test set contain multiple different types of problems, covering multiple fields such as continuous optimization, discrete optimization, and hybrid optimization.

each test problem as IGD value and standard deviation pairs, reflecting algorithm performance and variability respectively. A smaller value of IGD indicates a better performance of the algorithm on that problem. The standard deviation is used to measure the reliability of the results, with smaller values indicating more stable results. The "+/-/=" in the table indicates that the performance of the algorithm is better/worse/equal to the algorithm proposed in this paper compared to the algorithm proposed in this paper. By analyzing these data, the proposed algorithm demonstrates reasonably favorable performance on most problems under examination, as evidenced by the results. However, the performance of the I-MOMFEA-II algorithm on CEC17-MTMO2 is comparable to the other algorithms. In addition, for the CEC17-MTMO8 problem, the I-MOMFEA-II algorithm performs relatively poorly, probably because the complexity of the CEC17-MTMO8 problem is high, its solution space can be of high dimensionality, and constraints may exist that amplify the difficulties faced by the algorithm during optimization. From the statistical point of view, the I-MOMFEA-II algorithm outperforms the other algorithms.

5. Conclusion

With the development of IoT, task scheduling problems with multi-cloud environments are becoming increasingly complex. Multi-task optimization models can better allocate resources to ensure that various scheduling tasks run at the fastest speed and best effect. In this paper, according to the data characteristics and resource requirements of CPU-intensive tasks and I/O-intensive tasks in multi-cloud environments, we construct a multi-task multi-objective model. In order to schedule these two types of tasks at the same time, we introduce the idea of multi-factor optimization. And proposed a multi-objective multi-factor evolutionary algorithm based on quadratic crossover, I-MOMFEA-II. Through experiments, we can conclude the following: First, in the experiment of solving the scheduling model of two types of tasks in the MCE, under three different task quantities, the I-MOMFEA-II

Table 5. Numerical analysis of the three objectives of different algorithms for IO-intensive tasks in different task number scenarios

	I-MO-MFEA-II	**MO-MFEA-II**	**MO-MFEA**	**EMT-PD**	**AMT-NSGA-II**
CEC17-MTMO1-CI-HS-T1	6.6762e-03 (7.83e-04)	7.0321e-03 (1.28e-03) =	2.7681e-01 (5.78e-02) -	1.3412e-01 (2.79e-02) -	2.9384e-01 (2.64e-01) -
CEC17-MTMO1-CI-HS-T2	4.1054e-02 (6.97e-03)	4.7409e-02 (9.61e-03) -	5.3060e-01 (6.73e-02) -	3.8413e-01 (4.59e-02) -	3.6205e-01 (1.75e-01) -
CEC17-MTMO2-CI-MS-T1	7.2902e-03 (2.64e-03)	9.4576e-03 (5.43e-03) =	9.7653e-03 (6.08e-03) =	6.9870e-03 (2.45e-03) =	1.0402e-02 (6.08e-03) -
CEC17-MTMO2-CI-MS-T2	1.2344e-02 (5.32e-03)	9.5225e-02 (4.49e-01) =	1.4965e-02 (6.48e-03) =	1.1486e-02 (5.04e-03) =	1.0472e-02 (4.37e-03) =
CEC17-MTMO3-CI-LS-T1	6.4362e-03 (5.38e-04)	6.8211e-03 (6.56e-04) -	1.8280e+00 (5.18e+00) -	1.3280e+00 (6.43e+00) -	7.5394e+00 (1.31e+01) -
CEC17-MTMO3-CI-LS-T2	4.9716e-03 (1.94e-04)	4.9694e-03 (2.25e-04) =	1.6043e-02 (1.82e-02) -	1.2875e-02 (3.05e-02) -	1.4008e-02 (4.03e-03) -
CEC17-MTMO4-PI-HS-T1	1.0459e-02 (3.24e-03)	1.0063e-02 (1.88e-03) =	5.6257e-01 (1.62e-01) -	7.2766e-01 (2.07e-01) -	2.5821e-01 (2.28e-01) -
CEC17-MTMO4-PI-HS-T2	3.8166e-01 (1.78e-01)	6.1772e-01 (4.37e-01) -	3.6630e+01 (8.39e+00) -	3.6982e+01 (8.35e+00) -	3.9510e+01 (9.18e+00) -
CEC17-MTMO5-PI-MS-T1	1.2592e-01 (3.05e-02)	1.3300e-01 (3.55e-02) =	9.9677e-01 (1.58e-01) -	1.1360e+00 (5.10e-01) -	6.5402e-01 (9.60e-02) -
CEC17-MTMO5-PI-MS-T2	4.5107e+02 (9.26e+01)	4.3336e+02 (1.12e+02) =	1.1522e+03 (1.48e+02) -	1.0883e+03 (2.50e+02) -	1.3177e+03 (3.02e+02) -
CEC17-MTMO6-PI-LS-T1	7.9716e-03 (4.19e-03)	8.7751e-03 (4.00e-03) =	8.0392e-02 (1.53e-02) -	1.1073e-01 (1.71e-02) -	2.1130e-02 (1.76e-02) -
CEC17-MTMO6-PI-LS-T2	2.0045e+01 (2.51e-02)	2.0042e+01 (1.49e-02) =	3.4612e+00 (2.83e-01) +	3.7265e+00 (3.83e-01) +	4.2312e-01 (3.24e-01) +
CEC17-MTMO7-NI-HS-T1	5.1470e+01 (2.06e+01)	4.7918e+01 (3.74e-01) =	9.0859e+01 (1.25e+01) -	7.2926e+01 (6.27e+00) -	1.1440e+02 (5.00e+01) -
CEC17-MTMO7-NI-HS-T2	9.2787e-03 (3.68e-03)	9.6652e-03 (1.76e-03) -	3.0700e-01 (7.32e-02) -	1.6649e-01 (4.98e-02) -	1.4998e-01 (1.45e-01) -
CEC17-MTMO8-NI-MS-T1	4.8994e+01 (3.64e+01)	5.0713e+01 (3.28e+01) =	1.3548e+01 (5.48e+00) +	1.4928e+01 (1.06e+01) +	2.7387e+01 (2.50e+01) =
CEC17-MTMO8-NI-MS-T2	7.7138e-01 (6.19e-01)	1.0067e+00 (7.04e-01) =	2.4486e-01 (3.56e-01) +	3.1436e-01 (4.25e-01) +	8.2170e-01 (1.32e+00) =
CEC17-MTMO9-NI-LS-T1	7.1922e-02 (3.89e-03)	7.1180e-02 (2.82e-03) =	2.4881e-01 (6.38e-02) -	6.3871e-01 (2.47e-01) -	8.7098e-01 (1.89e-01) -
CEC17-MTMO9-NI-LS-T2	2.0294e+01 (3.08e-04)	2.0295e+01 (2.14e-04) =	2.0331e+01 (8.70e-03) -	2.0344e+01 (8.92e-03) -	2.0370e+01 (5.07e-02) -
+/-/=	Base	0/4/14	3/13/2	2/13/3	3/14/1

algorithm achieved the best results in three objectives: completion cost, completion time, and energy consumption of CPU-intensive tasks. It also performed well in the completion cost, completion time, and virtual machine throughput of I/O-intensive tasks. In addition, the I-MOMFEA-II algorithm showed good convergence of the objectives of the two types of tasks. In summary, the I-MOMFEA-II algorithm performed well in solving the scheduling problems of two types of tasks in the MCE and can provide new solutions to task scheduling problems in the multi-cloud environment. Second, in the IGD performance experiment, the algorithm in this paper performed better on most test sets, proving that this proposed algorithm surpasses other existing approaches. Future research can further explore more complex multi-cloud environment scheduling problems and further improve the algorithm to meet more challenges and requirements.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 61806138); China University Industry-University-Research Collaborative Innovation Fund (Future Network Innovation Research and Application Project) (Grant 2021FNA04014).

References

[1] L. Abualigah and M. Alkhrabsheh, *Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing*, The Journal of Supercomputing, 2021, 78(1), 740–765.

[2] T. A. Ahanger, H. A. Abdeljaber and M. Y. Uddin, *Development of a hybrid algorithm for efficient task scheduling in cloud computing environment using artificial intelligence*, International Journal of Computers Communications and Control, 2021, 16(5).

[3] K. K. Bali, A. Gupta, Y. S. Ong, et al., *Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II*, IEEE Trans Cybern, 2021, 51(4), 1784–1796.

[4] X. Cai, S. Geng, D. Wu, et al., *A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things*, IEEE Internet of Things Journal, 2021, 8(12), 9645–9653.

[5] Z. H. Cui, X. H. Xu, F. Xue, et al., *Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios*, IEEE Transactions on Services Computing, 2020. DOI: 10.1109/TSC.2020.2964552.

[6] B. Da, A. Gupta and Y. -S. Ong, *Curbing negative influences online for seamless transfer evolutionary optimization*, IEEE Transactions on Cybernetics, 2019, 49(12), 4365–4378.

[7] R. Dai, J. Jie, H. Zheng, et al., *Framework and experimental analysis of generalised surrogate-assisted particle swarm optimization*, International Journal of Computing Science and Mathematics, 2022, 15(4), 332–346.

[8] G. Ding and F. Dong, *An improved pigeon-inspired optimisation for continuous function optimisation problems*, International Journal of Computing Science and Mathematics, 2023, 17(3), 207–219.

[9] T. T. Dong, L. Zhou, L. Chen, et al., *A Hybrid algorithm for workflow scheduling in cloud environment*, International Journal of Bio-Inspired Computation, 2023, 21(1), 48–56.

[10] M. Farid, R. Latip, M. Hussin, et al., *Weighted-adaptive inertia strategy for multi-objective scheduling in multi-clouds*, Computers, Materials and Continua, 2022, 72(1), 1529–1560.

[11] A. Gupta, Y. S. Ong and L. Feng, *Multifactorial evolution: Toward evolutionary multitasking*, IEEE Transactions on Evolutionary Computation, 2016, 20(3), 343–357.

[12] A. Gupta, Y. S. Ong, L. Feng, et al., *Multiobjective multifactorial optimization in evolutionary multitasking*, IEEE Transactions on Cybernetics, 2017, 47(7), 1652–1665.

[13] Y. S. Hao, M. D. Xia, N. Wen, et al., *Parallel task scheduling under multiclouds*, KSII Transactions on Internet and Information Systems, 2017, 11(1), 39–60.

[14] S. Hubert Shanthan and B. J. Arockiam, *Rate aware meta task scheduling algorithm for multi cloud computing (RAMTSA)*, 2nd National Conference on Computational Intelligence, 2018, 012001, Bangalore, India.

[15] S. Kang, B. Veeravalli and K. M. M. Aung, *Dynamic scheduling strategy with efficient node availability prediction for handling divisible loads in multi-cloud systems*, Journal of Parallel and Distributed Computing, 2018, 113, 1–16.

[16] H. Lan, G. Xu and Y. Yang, *An enhanced multi-objective particle swarm optimization with levy flight*, International Journal of Computing Science and Mathematics, 2023, 17(1), 79–94.

[17] X. Lin, T. Ren, J. Yang, et al., *Multi-objective cellular memetic algorithm*, International Journal of Computing Science and Mathematics, 2022, 15(3), 213–223.

[18] Y. L. Lv, J. Zhang and L. L. Zuo, *Genetic regulatory network-based optimization of master production scheduling and mixed-model sequencing in assembly lines*, International Journal of Bio-Inspired Computing, 2022, 20(3), 150–159.

[19] J. P. B. Mapetu, Z. Chen and L. Kong, *Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing*, Applied Intelligence, 2019, 49(9), 3308–3330.

[20] S. K. Mishra, *Energy-aware task allocation for multi-cloud networks*, IEEE Access, 2020, 8, 178825–178834.

[21] Z. Liang, W. Liang, Z. Wang, et al., *Multiobjective evolutionary multitasking with two-stage adaptive knowledge transfer based on population distribution*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(7), 4457–4469.

[22] Z. Liang, J. Zhang, L. Feng, et al., *A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking*, Expert Systems with Applications, 2019, 138.

[23] J. Liu, P. Li, G. Wang, et al., *A multitasking electric power dispatch approach with multi-objective multifactorial optimization algorithm*, IEEE Access, 2020, 8, 155902–155911.

[24] A. Roy, S. Midya, D. Hazra, et al., *A hybrid task scheduling algorithm for efficient task management in multi-cloud environment*, Advances in Intelligent Systems and Computing, 2018, 811, 47–57.

[25] W. H. Tan and J. Mohamad-Saleh, *Alligator optimisation algorithm for solving unconstrainted optimisation problems*, International Journal of Bio-Inspired Computation, 2023, 21(1), 11–25.

[26] Z. Tang, M. Gong and M. Zhang, *Evolutionary multi-task learning for modular extremal learning machine*, in 2017 Congress on Evolutionary Computation, 2017, 474–479.

[27] H. ThiThanh Binh, P. Dinh Thanh, T. Ba Trung, et al., *Effective multifactorial evolutionary algorithm for solving the cluster shortest path tree problem*, in 2018 Congress on Evolutionary Computation, 2018, 1–8.

[28] L. J. Wu, D. Wu, T. H. Zhao, et al., *Dynamic multi-objective evolutionary algorithm based on knowledge transfer*, Information Sciences, 2023, 636, 118886.

[29] F. Xue, Q. Hai, Y. Gong, et al., *RVEA-based multi-objective workflow scheduling in cloud environments*, International Journal of Bio-Inspired Computation, 2022, 20(1), 49–57.

[30] W. S. Yang, L. Chen, Y. Y. Li, et al., *A many-objective particle swarm optimization algorithm based on convergence assistant strategy*, International Journal of Bio-Inspired Computing, 2022, 20(2), 104–118.

[31] Y. Yuan, Y. S. Ong, L. Feng, et al., *Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics, and baseline results*, 2017. DOI: 10.48550/arXiv.1706.02766.

[32] Q. Zhang, S. Geng and X. Cai, *Survey on task scheduling optimization strategy under multi-cloud environment*, Computer Modeling in Engineering and Sciences, 2023, 135(3), 1863–1900.

[33] T. H. Zhao, L. J. Wu, D. Wu, et al., *Multi-factor evolution for large-scale multiobjective cloud task scheduling*, KSII Transactions on Internet and Information Systems, 2023, 17(4), 1100–1122.

[34] L. Zhou, L. Feng, J. Zhong, et al., *Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem*, in 2017 Symposium Series on Computational Intelligence, 2017, 1–8.

[35] M. Zhou, R. Wu and H. Sun, *An artificial bee colony algorithm with a distance factor*, International Journal of Computing Science and Mathematics, 2022, 16(4), 355–376.

[36] Q. H. Zhu, H. Tang, J. J. Huang, et al., *Task scheduling for multi-cloud computing subject to security and reliability constraints*, IEEE/CAA Journal of Automatica Sinica, 2021, 8(4), 848–865.

[37] E. Zitzler and L. Thiele, *Multiobjective optimization using evolutionary algorithms - A comparative case study*, in 5th International Conference on Parallel Problem Solving from Nature, PPSN 1998, Amsterdam, Netherlands, 1998, 292–301.

[38] E. Zitzler, L. Thiele, M. Laumanns, et al., *Performance assessment of multiobjective optimizers: An analysis and review*, IEEE Transactions on Evolutionary Computation, 2003, 7(2), 117–132.