Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032215
Article

Optimization Task Scheduling Using Cooperation Search Algorithm for Heterogeneous Cloud Computing Systems

Ahmed Y. Hamed[1], **M. Kh. Elnahary**[1,*], **Faisal S. Alsubaei**[2] **and Hamdy H. El-Sayed**[1]

[1]Department of Computer Science, Faculty of Computers and Artificial Intelligence, Sohag University, Sohag, 82524, Egypt
[2]Department of Cybersecurity, College of Computer Science and Engineering, University of Jeddah, Jeddah, 21959, Saudi Arabia
*Corresponding Author: M. Kh. Elnahary. Email: mk409055@gmail.com
Received: 10 May 2022; Accepted: 05 July 2022

Abstract: Cloud computing has taken over the high-performance distributed computing area, and it currently provides on-demand services and resource polling over the web. As a result of constantly changing user service demand, the task scheduling problem has emerged as a critical analytical topic in cloud computing. The primary goal of scheduling tasks is to distribute tasks to available processors to construct the shortest possible schedule without breaching precedence restrictions. Assignments and schedules of tasks substantially influence system operation in a heterogeneous multiprocessor system. The diverse processes inside the heuristic-based task scheduling method will result in varying makespan in the heterogeneous computing system. As a result, an intelligent scheduling algorithm should efficiently determine the priority of every subtask based on the resources necessary to lower the makespan. This research introduced a novel efficient scheduling task method in cloud computing systems based on the cooperation search algorithm to tackle an essential task and schedule a heterogeneous cloud computing problem. The basic idea of this method is to use the advantages of meta-heuristic algorithms to get the optimal solution. We assess our algorithm's performance by running it through three scenarios with varying numbers of tasks. The findings demonstrate that the suggested technique beats existing methods New Genetic Algorithm (NGA), Genetic Algorithm (GA), Whale Optimization Algorithm (WOA), Gravitational Search Algorithm (GSA), and Hybrid Heuristic and Genetic (HHG) by 7.9%, 2.1%, 8.8%, 7.7%, 3.4% respectively according to makespan.

Keywords: Heterogeneous processors; cooperation search algorithm; task scheduling; cloud computing

1 Introduction

The cloud computing model emerged with the growth of the Internet and the services that the Internet provides to its users. The cloud model is based on distributed computing and consists of different virtual computers that can be interconnected and dynamic to form computing resources. The vacant resources should be used globally to increase the utilization and gain of the resources by increasing the economic efficiency of these resources. The cloud model best serves this. The primary goal of the cloud computing concept is to enable users to share data and resources. It is a platform to provide applications and services to its users. There are three forms of cloud computing: platform as a service (paas), software as a service (saas), and infrastructure as a service (iaas) [1]. These services are available to users on a pay-on-demand basis and share servers, computing resources, applications, networks, and data storage. The licenced software is delivered to the user on a subscription basis for the services in a saas service. These services may be accessed from any device using a web browser. In paas, users can create their services using the available cloud services and then publish those services on their devices. In iaas, the organizational infrastructure for clients is available online. To use this, the consumer does not need to comprehend the internal nature of the infrastructure [1]. Instead of buying the entire infrastructure for business requirements, which the customer takes as basic rent when there are no more infrastructure requirements, the customer uses the amount paid for the services. In the last year, the number of cloud users has increased, so the volume of tasks you need to manage by default for this task is required for scheduling [1]. To solve the problem of scheduling tasks better, we have proposed a novel efficient approach based on the cooperation search algorithm called the Efficient Cooperation Search Algorithm (ECSA) to minimize the makespan of the user requests on the resources. In the cooperation search algorithm, the representation of a vector is a continuous value, so we use two methods to convert the continuous value to a discrete value. We assess our algorithm's performance by running it through three cases with varying numbers of tasks.

The following is the paper's structure: Section 2 contains the notations. Related work for problems of scheduling tasks is given in Section 3. The problem description is given in Section 4. Section 5 shows the cooperation search algorithm. Section 6 describes the ECSA approach. The results were obtained by applying the ECSA and compared with the other results and discussion presented in Section 7. Section 8 concludes and offers future work.

2 Notations

TG	represents the graph of tasks
Tas_i	represents the task i
$Pros_i$	represents the processor i
NMP	represents the processor's number
NMT	represents the task's number
$C(Tas_i, Tas_j)$	The communication cost between Tas_i and Tas_j
$Start\_Time(Tas_i, Pros_j)$	represents the task's start time i on a processor $Pros_j$
$Finish\_Time(Tas_i, Pros_j)$	represents the task's finish time i on a processor $Pros_j$
$Ready\_Time(Pros_i)$	represents the processor's ready time i
LT	represents a list of jobs arranged in directed acyclic graph topological order
$Data\_Arrive(Tas_i, Pros_j)$	represents the time of data arrival of task i to processor j

3 Related Work

In terms of cloud computing, scheduling tasks directly influences resource utilization and operational expenses. Many metaheuristic approaches and variants for optimizing scheduling have been created to boost the efficiency of task executions in the cloud. This study [2] introduces the Whale Optimization Algorithm (WOA) for scheduling tasks in the cloud using a multiobjective improvement model, intending to increase the cloud system performance with limited resources for computing. Based on that assumption, an improved method called Improved Whale Optimization Algorithm (IWC) for scheduling tasks in the cloud was created to boost the whale optimization algorithm-based method's best solution search capability. The algorithm aims to Improve convergence speed and accuracy and increase the resource utilization cost.

Cloud computing is a relatively new paradigm for exploiting distant resources for computing; It is also becoming a more robust and popular technology that enables on-demand resource allocation and release in near real-time. Scheduling tasks are crucial in cloud computing and are one of the most challenging things to master in this field. As a result, a strong and effective scheduling system is necessary to better resource utilization. An excellent algorithm for scheduling tasks can improve the quality of service, overall performance, and end-user experience. As the number of available tasks grows, so does the complexity of the problem, resulting in a huge search space. To offer a technique capable of finding a near-optimal solution for a multiobjective scheduling task issue in a cloud computing environment while also reducing search time. The authors provide the hybridized bat algorithm, a swarm-intelligence-based technique for multiobjective task scheduling [3]. The algorithm aims to utilize makespan, cost, and Hyervolume indicator.

The most critical need in a cloud computing environment is scheduling tasks since it is vital to the overall functioning of cloud computing facilities. In cloud computing, scheduling tasks implies allocating the possible resources for the work to be executed while considering dependability, time, scalability, cost, makespan, throughput, resource consumption, availability, etc. The suggested approach takes reliability and availability into account. Because of the difficulties of achieving these criteria, most scheduling methods do not incorporate cloud computing ecosystem stability and availability. A mathematical model for cloud computing that uses mutation of load balancing and a particle swarm optimization based allocation and schedule that considers the time of transmission, time of the round trip, reliability, time of execution, load balancing, makespan, cost of transmission, and between virtual machines and tasks is presented [4]. It can contribute to the dependability of the cloud computing environment by considering the availability of resources and rescheduling tasks that fail to allocate them.

Several advantages are provided by central cloud facilities based on virtual computers, including reduced costs of scheduling and improved service accessibility and availability. The cloud computing strategy is viable because of the combination of security measures and online services. The feature spaces of the source and destination domains differ in task transfer. This problem is exacerbated by network traffic, which causes data transmission delays and prevents some vital operations from being completed on time. This work introduces a hybrid multi-verse optimizer with a genetic algorithm as a practical optimization approach for job scheduling. The suggested algorithm is intended to improve the performance of the tasks transported over the cloud network based on the workload of resources in the cloud. It is required to give suitable decisions of transfer to reschedule the transfer tasks in the cloud based on the efficiency weight of the acquired jobs. The suggested technique considers a variety of cloud resource attributes, including throughput, capacity, number of virtual machines, number and size of tasks, and speed [5]. The algorithm aims to optimize the transfer time of large cloud tasks, which reflects its effectiveness.

Cloud computing is a type of on-demand Internet-based computing widely used by both working and non-working individuals throughout the world. One of the essential applications utilized by cloud service providers and end-users is task scheduling. The task scheduler's main challenge is to locate the best resource for the given input job. A hybrid electro search combined with a genetic algorithm is proposed [6] to optimize scheduling task behaviour by considering load balancing, makespan, multi-cloud cost, and resource usage criteria. The proposed solution leveraged the benefits of both an electrical search algorithm and a genetic algorithm. The genetic algorithm gives the best solutions within the local optimum, whereas the electro-search method delivers the best solutions within the global optimal. The algorithm aims to decrease makespan, cost, and response time.

The scheduling task is essential to improving the performance of collaborative and distributed e-science and e-business applications at scale. This typical application includes multiple communication tasks that are performed on virtual machines. The primary objective of any scheduling method is to reduce the extent of the configuration, which reflects the exit task completion time. Focusing on downsizing for allocating or scheduling multiple tasks across heterogeneous virtual machines, in this paper [7] a model based on the Crow Search Algorithm (CSA) was proposed and the main objective of this model is to reduce the makespan.

Scheduling tasks is the primary issue in cloud computing, which reduces the system's performance. It's an effective method to organize the user's requirements and achieve multiple goals. An efficient scheduling task algorithm is required to improve the system's performance. A Genetic Algorithm (GA) has been described for task assignment and execution. The algorithm aims to decrease the execution cost and makespan of tasks and increase resource utilization [8].

4 Problem Description

During this work, the task scheduling model is defined as distributed Number of Tasks (NMT) tasks to be implemented on Number of Processors (NMP) processors. The processors may be different in general. Task Graph (TG) may be mapped to describe the problem structure. TG is a Directed Acyclic Graph (DAG) made up of NMT tasks: Tas_1, Tas_2, Tas_3, ... Tas_n. Every node in the graph is termed a task. A task is assumed to be a set of instructions that must be implemented sequentially by an assigned processor. A task (node) might have pre-demanded data (inputs) before implementation. When all the inputs are received, the task can be triggered to be implemented. These inputs are expected to be delivered after some other tasks end their implementation, as these tasks evaluate them. We call this relying on task dependency. If a task is dependent on other tasks' data, then we consider that task as the parent of the task, and the task is their child. A task with no parents is called an entry task, and a task with no children is called an exit task [9]. The execution time of a task is what we call the computation cost. Whenever the computation cost of a task Tas_i is denoted by weight (Tas_i, $Pros_j$), the graph additionally has directed edges representing a partial order among the tasks. The partial order introduces a precedence-constrained DAG and implies that if ($Tas_i \rightarrow Tas_j$), then Tas_j is a child, which means it cannot start until its parent Tas_i finishes. The weight on edge represents the communication cost between the tasks and is denoted by $C(Tas_i, Tas_j)$; the communication cost is considered only if Tas_i and Tas_j are assigned to different processors; otherwise, it's calculated to be zero. In that case, Tas_i and Tas_j are assigned to the same processor. If a node Tas_i is assigned to processor $Pros_j$, the task's start time and finish time are denoted by $Start\_Time(Tas_i, Pros_j)$ and $Finish\_Time(Tas_i, Pros_j)$, respectively. After scheduling the tasks, the makespan is defined as the max $\{Finish\_Time(Tas_i, Pros_j)\}$

across all processors. The task scheduling problem is to find a schedule for the tasks in the processors such that the makespan is decreased over possible schedules, where the task dependency constraints are preserved. Task dependency constraints state that any task can't start until all parents have finished. Assume that Pros_j is the processor and that the Kpath parent task Tas_k of task Tas_i, is scheduled. The Data_Arrive of Tas_i at processor Pros_j is the time at which the per-demanded data for the task execution becomes available. This is defined in [9] by the following:

$$\text{Data\_Arrive}(\text{Tas}_i, \text{Pros}_j) = \max\left\{\text{Finish\_Time}\left(\text{Tas}_k, \text{Pros}_j\right) + \text{C}\left(\text{Tas}_i, \text{Tas}_k\right)\right\} \text{ where kpa} = 1, 2, 3 \ldots \text{Number of Parent} \tag{1}$$

$$\text{Start\_Time}(\text{Tas}_i, \text{Pros}_j) = \max\left\{\text{Ready\_Time}(\text{Pros}_j), \text{Data\_Arrive}\left(\text{Tas}_i, \text{Pros}_j\right)\right\} \tag{2}$$

$$\text{Finish\_Time}(\text{Tas}_i, \text{Pros}_j) = \text{Start\_Time}\left(\text{Tas}_i, \text{Pros}_j\right) + \text{weight}\left(\text{Tas}_i, \text{Pros}_j\right) \tag{3}$$

$$\text{Ready\_Time}(\text{Pros}_j) = \text{Finish\_Time}\left(\text{Tas}_i, \text{Pros}_j\right) \tag{4}$$

$$\text{Makespan} = \max\left\{\text{Finish\_Time}\left(\text{Tas}_i, \text{Pros}_j\right)\right\} \quad \text{where i} = 1, 2, 3 \ldots \text{NMT} \tag{5}$$

5 Cooperation Search Algorithm

The specifics of the Cooperation Search Algorithm (CSA) approach [10] are provided in this part to provide a unique additional tool for global optimization problems: The cooperative team behaviours in current organizations are explained first, followed by the search concept of the CSA approach. The following are some examples of frequent restricted minimization optimization problems:

$$\begin{array}{ll} \min g(y) & y = \left[y_1, \ldots, y_j, \ldots, y_J\right] \in R^J \\ \text{Such that } gg_{ee}(y) \le 0 & ee = 1, 2, 3, \ldots, EE \\ hh_g(y) = 0 & g = 1, 2, 3, \ldots, G \\ \underline{y_j} \le y_j \le \overline{y_j} & j = 1, 2, 3, \ldots, J \end{array} \tag{6}$$

where g(x) denotes the J-dimensional solution's objective value. y_j denotes the jth variable in solution y, $\overline{y_j}$ and $\underline{y_j}$ are the upper and lower bounds of the jth variable, respectively. The number of possible decision factors is denoted by the letter J. The inequality constraint is $gg_{ee}(y)$. The gth equality constraint is expressed as $hh_g(y)$. The variables G and EE represent the number of equality and inequality constraints, respectively.

5.1 Insights from Teamwork Behaviour in Modern Businesses

In recent years, all types of businesses have played an increasingly essential part in economic and social growth. Cooperative team conduct is critical to one company's normal functioning. The boards of supervisors and directors, the chairman, and the staff are four different positions frequently used in the corporate cooperation process. The board of directors, which comprises shareholders who have been chosen to represent them, oversees the company's business operations and administers the productive duties inside. In other words, the board of directors oversees all of the company's affairs and transactions. The board of supervisors is tasked with supervising the executive directors and promoting the interests of the shareholders [10]. In contrast, the boards of supervisors and directors can't participate in the firm's corporate decision-making process and can't represent the organization externally. As an executive elected by the board of directors, the chairman is primarily responsible for the company's scientific activities. Furthermore, as the firm's spokesman, the chairman frequently has significant, if not decisive, influence over the company to ensure smooth and orderly operations before reaching a consensus on board decisions. The staff is required to participate in specific tasks under the direction of the board of directors, which typically has the authority to select members of the boards of supervisors and directors. It is generally recognized that human people are one of the essential components in increasing productivity, implying that growing staff strength is a critical aspect of the company's scientific development. In order to achieve this aim, it is vital to assist individual staff members in gaining as much information as possible. In general, a group of leaders on the boards of supervisors and directors and a chairperson can all impact staff knowledge at the same time. Because of their role as team leaders, the chairman on duty frequently has the most power. At the same time, members of the boards of supervisors and directors supply ample information to minimize or even eliminate potential mistakes. After some time, each staff member is urged to contemplate self-improvement methods and given the option to assume the position of their superior leaders once their performance has improved. In other words, the chairman and the members of the board of directors and supervisors can be dynamically updated to boost the company's market competitiveness. It has been discovered that there are close connections among members of various social statuses; underperforming leaders or staff might be replaced by promising young people, while ordinary staff has the opportunity to improve their knowledge and advance their job positions through hard work. As a result, the firm can continue to operate to achieve sustainable development [10].

5.2 The CSA Method's Search Principle

In CSA, the optimization process of the target problem is viewed as the development of an enterprise; each solution is viewed as a staff member, while a group of staff members forms an enterprise team; the value of the fitness of the problem is equal to the performance per staff member; the board of supervisors is made up of personal best-known solutions; the board of directors is made up of the external archive set (M global best-known solutions found by far); the chairman on duty is rando. The population may then gradually develop high-quality solutions by replicating team cooperation behaviours in the modern industry by employing three evolutionary operators: The team communication operator is used to assist employees in capturing useful leaders' knowledge; The operator for reflective learning is used to increase the staff's overall strength by drawing lessons from the past. The internal competition operator improves the top solutions' work experiences and leadership vision. Following that, the technical elements of the CSA technique are as follows [10]:

The team-building phase: The equation determines all team members at random Eq. (7). After evaluating all solutions' performances, $M\epsilon\,[1, I]$ solutions for leaders will be picked from the original swarm to create the external elite set.

$$y_{i,j}^{k} = O(\underline{y_j},\ \overline{y_j}),\ \text{where } i\ \in [1, I],\ j\ \in [1, J],\ k = 1 \tag{7}$$

where I is the current swarm's number of solutions. The j^th value of the i^th solution during the k^th cycle is represented by $y_{i,j}^{k}$. O (Lower Bound (LOB), Upper Bound (UPB)) is a function that generates a random number with a uniform distribution in the range [LOB, UPB] [10].

The operator of team communication: Each staff member may learn new messages by sharing information with the chairman and the board of supervisors and directors. As shown in Eq. (8), the team communication process consists of the chairman's knowledge W, E the board of directors' collective knowledge, and the collective knowledge Z from the board of supervisors. The chairman is picked randomly from the board of directors to replicate the rotating mechanism. In contrast, all supervisors and directors' board members are assigned identical positions in calculating E and Z [10].

$$q_{i,j}^{k+1} = y_{i,j}^{k} + W_{i,j}^{k} + E_{i,j}^{k} + Z_{i,j}^{k}, \text{where } i \in [1, I],\ j \in [1, J],\ k \in [1, K] \quad (8)$$

$$W_{i,j}^{k} = \log\left(\frac{1}{O(0,1)}\right).\left(globest_{inde,j}^{k} - y_{i,j}^{k}\right) \quad (9)$$

$$E_{i,j}^{k} = \alpha\ .O(0,1).\left[\frac{1}{M}\sum_{m=1}^{M} globest_{m,j}^{k} - y_{i,j}^{k}\right] \quad (10)$$

$$Z_{i,j}^{k} = \beta.O(0,1).\left[\frac{1}{I}\sum_{i=1}^{I} perbest_{i,j}^{k} - y_{i,j}^{k}\right] \quad (11)$$

$q_{i,j}^{k+1}$ denotes the jth value of the ith group solution during the (k + 1)th cycle. The jth value of the ith personal best-known solution during the kth cycle is given by $perbest_{i,j}^{k}$. The jth value of the indeth global best-known solution from the beginning of the kth cycle is represented by $globest_{inde,j}^{k}$. The index inde is chosen at random from the set of {1, 2, . . . , M}. $W_{i,j}^{k}$ represents the information obtained from the chairman, who was randomly picked from the external elite group. $E_{i,j}^{k}$ and $Z_{i,j}^{k}$ represent the average knowledge acquired from M global best-known solutions discovered by far and I personal best-known solutions, respectively. α and β are the learning coefficients used to alter the influence degrees of $E_{i,j}^{k}$ and $Z_{i,j}^{k}$, respectively [10].

The operator of reflective learning: Aside from learning from the leader's solutions, the staff may also receive new information by summarising their own experience in the other way, as seen below [10]:

$$l_{i,j}^{k+1} = \begin{cases} h_{i,j}^{k+1} \text{ if } \left(q_{i,j}^{k+1} \geq z_j\right) \\ s_{i,j}^{k+1} \text{ if } \left(q_{i,j}^{k+1} < z_j\right) \end{cases}, \text{where } i \in [1, I],\ j \in [1, J],\ k \in [1, K] \quad (12)$$

$$h_{i,j}^{k+1} = \begin{cases} O\left(\overline{y_j} + \underline{y_j} - q_{i,j}^{k+1}, z_j\right) & \text{if} \left(\left|q_{i,j}^{k+1} - z_j\right| < O(0,1).\left|\overline{y_j} - \underline{y_j}\right|\right) \\ O\left(\underline{y_j}, \overline{y_j} + \underline{y_j} - q_{i,j}^{k+1}\right) & \text{otherwise} \end{cases} \quad (13)$$

$$s_{i,j}^{k+1} = \begin{cases} O\left(z_j, \overline{y_j} + \underline{y_j} - q_{i,j}^{k+1}\right) & \text{if} \left(\left|q_{i,j}^{k+1} - z_j\right| < O(0,1).\left|\overline{y_j} - \underline{y_j}\right|\right) \\ O\left(\overline{y_j} + \underline{y_j} - q_{i,j}^{k+1}, \overline{y_j}\right) & \text{otherwise} \end{cases} \quad (14)$$

$$z_j = \frac{1}{2}.\left(\overline{y_j} + \underline{y_j}\right) \quad (15)$$

$l_{i,j}^{k+1}$ denotes the jth value of the ith reflective solution during the (k + 1)th cycle.

The operator of internal competition: The team steadily improves its competitiveness in the market by ensuring that all employees with superior performance are always retained, as shown below [10]:

$$y_{i,j}^{k+1} = \begin{cases} q_{i,j}^{k+1} \text{ if } \left(G\left(q_i^{k+1}\right) \leq G\left(l_i^{k+1}\right)\right) \\ l_{i,j}^{k+1} \text{ if } \left(G\left(q_i^{k+1}\right) > G\left(l_i^{k+1}\right)\right) \end{cases}, \text{where } i \in [1, I],\ j \in [1, J],\ k \in [1, K] \quad (16)$$

G(y) denotes the value of fitness of the solution y. In order to implement multiple physical constraints, Eq. (17) is used to modify all the variables in y to the feasible zone. Then the penalty functions approach in Eq. (18) is used to achieve the fitness value G (y) by combining the value of constraint violation with the objective value g (y). Then, all constraints are well fulfilled for feasible solutions, resulting in the original objective value being equal to a fitness value; for infeasible alternatives, the constraint violation value becomes positive, resulting in a fitness value more significant than the objective value. As a result, the swarm may be directed to a practical search region [10].

$$y_j = \max\left\{\min\left\{\overline{y}_j,\ y_j\right\},\ \underline{y_j}\right\} \tag{17}$$

$$G(y) = g(y) + \sum_{ee=1}^{EE} z_{ee}^{1}.\max\left\{gg_{ee}\ (y)\,,0\right\} + \sum_{g=1}^{G} z_{g}^{2}.\left|hh_{g}(y)\right| \tag{18}$$

where y_j is the jth value to be evaluated from the solution y. The eeth inequality constraint penalty coefficient is denoted by z_{ee}^{1}. For the gth inequality constraint, the penalty coefficient is z_{g}^{2}.

Cooperation Search Algorithm [10]

```
Define the goal function and all physical limitations.
By using Eq. (7), generate a random swarm in the feasible space
By using Eq. (18), determine the fitness values of the initial solutions
Make the group and reflective solutions the same as the original one.
iteration = 1
While (iteration <= max of iteration)
For the current swarm, update I (personal best-known solutions)
Update M global best-known solutions found
By using Eqs. (8)–(11), obtain I group solutions for global exploitation
By using Eqs. (12)–(15), obtain I reflective solutions for local exploration
By using Eq. (18), calculate the fitness values of the group and reflective solutions
For the next cycle, use Eq. (16) to select I better solutions
iteration = iteration + 1
End while
The final solution to the problem is the global best-known individual
```

6 The Proposed Algorithm

We can see that the representation of a vector is a continuous value; therefore, we will apply the Smallest Position Value (SPV) rule [11] and the Largest Position Value (LPV) rule [12], as well as the modulus function with the number of processors and increasing the value by 1, as shown in Fig. 1.

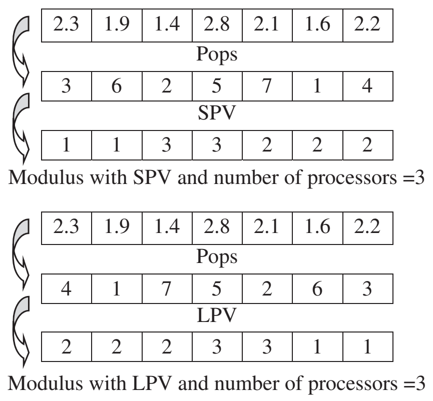

Figure 1: An example of the proposed schedule

Where the tasks Tas_1 and Tas_3 are scheduled into processor 1. The tasks Tas_2, Tas_6, and Tas_7 are scheduled into processor 2. The tasks Tas_4 and Tas_5 are scheduled into processor 3, as shown in Fig. 2.

1	2	1	3	3	2	2

Figure 2: The proposed schedule

The proposed algorithm

```
Input: DAG's computation cost and communication cost
Output: the best solution for makespan
Define all physical limitations
By using Eq. (7), generate a random swarm in the feasible space
Convert the generated swarm by using Algorithm 2
Calculate the makespan by using Algorithm 1
Make the group and reflective solutions the same as the original one.
iteration = 1
While (iteration <= max of iteration)
For the current swarm, update I (personal best-known solutions)
Update M global best-known solutions found
By using Eqs. (8)–(11), obtain I group solutions for global exploitation
By using Eqs. (12)–(15), obtain I reflective solutions for local exploration
Convert the obtained solutions by using Algorithm 2
Calculate the makespan by using Algorithm 1
For the next cycle, use Eq. (16) to select I better solutions
iteration = iteration + 1
End while
The final solution to the problem is the global best-known individual
```

Algorithm 1: Calculate the makespan of the task schedule using the Standard Genetic Algorithm (SGA) [9]

```
Input the schedule of tasks as shown in Fig. 2
Ready_Time[$Pros_j$] = 0          where j = 1, 2, 3 . . . . . . NMP.
For i = 1 to NMT
{
    Remove the first task $Tas_i$ from the list LT.
    For j = 1 to NMT
        {
      If $Tas_i$ is scheduled to processor $Pros_j$
        By using Eq. (2), compute start_time
        By using Eq. (3), compute finish_time
        By using Eq. (4), compute ready_time
      End If
        }
}
By using Eq. (5), compute the makespan
```

Algorithm 2: The function that converts a continuous value to a discrete value

```
Function Convert (s)
R = random number between [1, 2]
If   (R == 1) then
        Use the SPV rule as shown in Fig. 1
Else
        Use the LPV rule as shown in Fig. 1
End if
End function
```

We can see as shown above that Algorithm 1 is used to compute the makespan by taking the schedule after converting it by Algorithm 2, which detects the method that is used to convert the continuous value to a discrete value by generating a random number, the number maybe 1 or 2. The proposed algorithm uses the operation of a cooperation search algorithm to find the best solution for the makespan. We can see that the value of speedup, efficiency, and throughput depends on the makespan. The smaller the makespan, the higher the speedup, efficiency, and throughput.

7 Evaluation of the ECSA

In this part, we demonstrate the ECSA's efficacy by applying it to three cases. The first case is of three heterogeneous processors and eleven tasks. The second case consists of three heterogeneous processors and ten tasks. The third one consists of a different number of tasks and processors. ECSA was implemented as a system by MATLAB 2016. We set the initial values of the parameters initial population = 100, $\alpha = 0.10$, $\beta = 0.15$, max of iteration = 100.

Speedup is the ratio between the results obtained by assigning all tasks to a processor that gives the minimum makespan and the results obtained by executing tasks in parallel. It is calculated as shown in Eq. (19) [7].

$$\text{Speedup} = \min_{\text{pros}_j} \left(\sum\nolimits_{\text{Tas}_i} \frac{\text{weight}_{i,j}}{\text{makespan}} \right) \tag{19}$$

Efficiency is the ratio between the obtained speedup results and the total number of processors used. It is calculated as shown in Eq. (20) [7].

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{NMP}} \tag{20}$$

Throughput: The value of the throughput metric can be defined as the number of tasks executed per unit of time. It is calculated as shown in Eq. (21) [13].

$$\text{Throughput} = \frac{\text{NMT}}{\text{makespan}} \tag{21}$$

7.1 *Case 1*

In this case, the tasks $\{\text{Tas}_0, \text{Tas}_1, \text{Tas}_2, \text{Tas}_3, \text{Tas}_4, \text{Tas}_5, \text{Tas}_6, \text{Tas}_7, \text{Tas}_8, \text{Tas}_9, \text{Tas}_{10}\}$ will be executed on three heterogeneous processors $\{\text{Pros}_1, \text{Pros}_2, \text{Pros}_3\}$. The cost of executing each task on different processors is shown in Tab. 1 [14]. The schedule obtained by ECSA and other algorithms is shown in Tab. 2. The results obtained by the proposed ECSA are compared with those obtained by the Genetic

Algorithm (GA) [8], and New Genetic Algorithm (NGA) [14]. The task priority of ECSA {Tas_0, Tas_2, Tas_3, Tas_4, Tas_1, Tas_6, Tas_8, Tas_7, Tas_5, Tas_9, Tas_{10}}, task priority of NGA {Tas_0, Tas_4, Tas_3, Tas_2, Tas_8, Tas_1, Tas_6, Tas_5, Tas_7, Tas_9, Tas_{10}}, GA {Tas_0, Tas_2, Tas_6, Tas_4, Tas_1, Tas_3, Tas_8, Tas_7, Tas_5, Tas_9, Tas_{10}}. Figs. 3–6 represent the results obtained by the ECSA, NGA, and GA in terms of makespan, speedup, efficiency, and throughput.

Table 1: Computation cost for case 1

Tas/Pros	$Pros_1$	$Pros_2$	$Pros_3$
Tas_0	10	11	12
Tas_1	11	12	13
Tas_2	12	8	13
Tas_3	14	10	18
Tas_4	27	20	19
Tas_5	15	12	18
Tas_6	9	14	19
Tas_7	19	12	14
Tas_8	14	10	15
Tas_9	15	12	15
Tas_{10}	18	10	17

Table 2: Schedule obtained by ECSA and other algorithms for case 1

	NGA			GA			ECSA		
	$Pros_1$	$Pros_2$	$Pros_3$	$Pros_1$	$Pros_2$	$Pros_3$	$Pros_1$	$Pros_2$	$Pros_3$
Tas_0	0–10	-	-	-	0–11	-	-	0–11	-
Tas_1	-	-	38–51	38–49	-	-	36–47	-	-
Tas_2	-	29–37	-	-	11–19	-	-	11–19	-
Tas_3	-	-	20–38	-	39–49	-	-	19–29	-
Tas_4	10–37	-	-	-	19–39	-	-	29–49	-
Tas_5	-	-	51–69	49–64	-	-	47–62	-	-
Tas_6	47–56	-	-	29–38	-	-	-	-	29–48
Tas_7	-	67–79	-	-	49–61	-	-	59–71	-
Tas_8	-	37–47	-	-	-	28–43	-	49–59	-
Tas_9	-	79–91	-	-	73–85	-	-	71–83	-
Tas_{10}	-	91–101	-	-	85–95	-	-	83–93	-

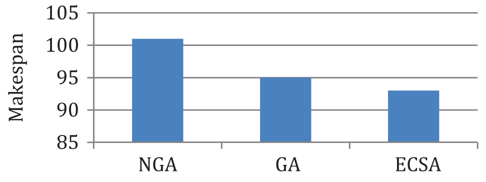

Figure 3: Comparison of makespan for case 1

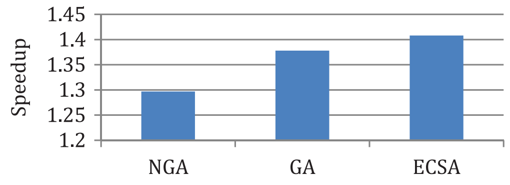

Figure 4: Comparison of speedup for case 1

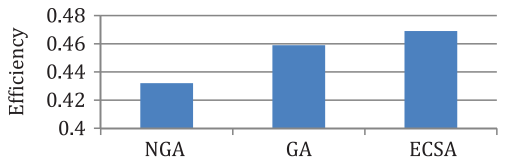

Figure 5: Comparison of efficiency for case 1

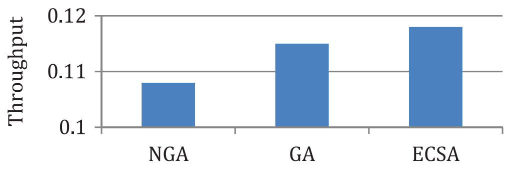

Figure 6: Comparison of throughput for case 1

7.2 Case 2

In this case, the tasks {Tas_1, Tas_2, Tas_3, Tas_4, Tas_5, Tas_6, Tas_7, Tas_8, Tas_9, Tas_{10}} are executed on three heterogeneous processors {$Pros_1$, $Pros_2$, $Pros_3$}. The cost of executing each task on different processors is shown in Tab. 3, [9]. The schedule obtained by ECSA and other algorithms is shown in Tab. 4. The results obtained by the ECSA are compared with those obtained by the Whale Optimization Algorithm (WOA) [15]. Gravitational Search Algorithm (GSA) [16], and Hybrid Heuristic and Genetic (HHG) [17]. The task priority of ECSA {Tas_1, Tas_6, Tas_4, Tas_5, Tas_2, Tas_3, Tas_8, Tas_9, Tas_7, Tas_{10}}, task priority of WOA {Tas_1, Tas_3, Tas_5, Tas_2, Tas_4, Tas_6, Tas_7, Tas_8, Tas_9, Tas_{10}}, task priority of GSA {Tas_1, Tas_3, Tas_2, Tas_6, Tas_4, Tas_5, Tas_7, Tas_8, Tas_9, Tas_{10}}, task priority of HHG {Tas_1, Tas_2, Tas_6, Tas_3, Tas_4, Tas_5, Tas_8, Tas_7, Tas_9, Tas_{10}}. Figs. 7–10 represent the results obtained by the ECSA, WOA, GSA, and HHG in terms of makespan, speedup, efficiency, and throughput.

Table 3: Computation cost for case 2

Tas/Pros	$Pros_1$	$Pros_2$	$Pros_3$
Tas_1	22	21	36
Tas_2	22	18	18
Tas_3	32	27	43
Tas_4	7	10	4
Tas_5	29	27	35
Tas_6	26	17	24
Tas_7	14	25	30
Tas_8	29	23	36

(Continued)

Table 3: Continued

Tas/Pros	$Pros_1$	$Pros_2$	$Pros_3$
Tas_9	15	21	8
Tas_{10}	13	16	33

Table 4: Schedule obtained by ECSA and other algorithms for case 2

	WOA			GSA			HHG			ECSA		
	$Pros_1$	$Pros_2$	$Pros_3$	$Pros_1$	$Pros_2$	$Pros_3$	$Pros_1$	$Pros_2$	$Pros_3$	$Pros_1$	$Pros_2$	$Pros_3$
Tas_1	-	0–21	-	-	0–21	-	-	0–21	-	-	0–21	-
Tas_2	-	-	38–56	-	-	38–58	-	21–39	-	-	21–39	-
Tas_3	-	21–48	-	-	21–48	-	-	39–66	-	-	39–66	-
Tas_4	-	48–58	-	50–57	-	-	-	-	50–54	64–71	-	-
Tas_5	34–63	-	-	-	-	56–91	-	-	54–89	-	-	34–69
Tas_6	-	58–75	-	-	48–65	-	38–64	-	-	38–4	-	-
Tas_7	64–78	-	-	64–78	-	-	-	66–91	-	-	66–91	-
Tas_8	-	75–98	-	-	68–91	-	65–94	-	-	71–100	-	-
Tas_9	86–101	-	-	-	-	91–99	-	-	89–97	-	-	78–86
Tas_{10}	-	108–124		-	106–122	-	104–117	-	-	100–113	-	-

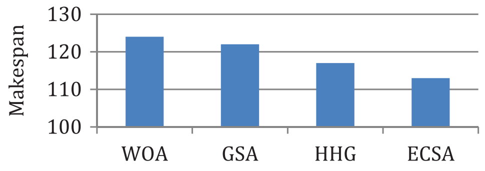

Figure 7: Comparison of makespan for case 2

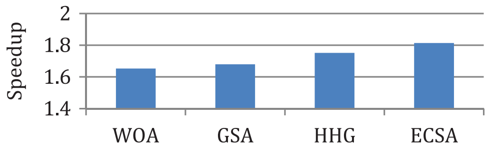

Figure 8: Comparison of speedup for case 2

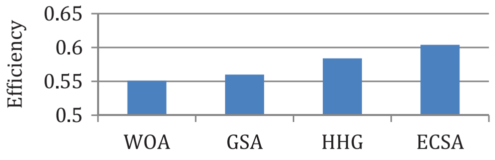

Figure 9: Comparison of efficiency for case 2

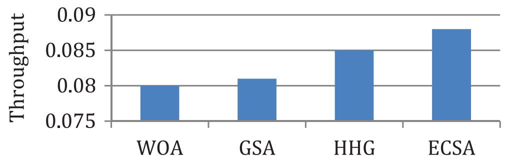

Figure 10: Comparison of throughput for case 2

7.3 Case 3

To analyze the behavior of the ECSA, we consider four cases with M = 2, 3, 4, and 8 processors. The number of tasks is 20, 30, 50, and 70 tasks in the first, second, third, and fourth cases (N = 20, 30, 50, and 70). The communication cost between the tasks and the computation cost of each task on different processors are randomly generated from 1 to 20, and 1 to 5, respectively. We run our proposed algorithm one more time. The results obtained by the proposed algorithm are shown in Tab. 5.

Table 5: The results obtained by ECSA for case 3

Number of tasks	Makespan	Speedup	Efficiency	Throughput	Number of processors
20	28	1.392	0.696	0.714	2
	26	1.5	0.5	0.769	3
	28	1.392	0.348	0.714	4
	33	1.181	0.147	0.606	8
30	33	1.575	0.787	0.909	2
	32	1.625	0.541	0.937	3
	34	1.529	0.382	0.882	4
	35	1.485	0.185	0.857	8
50	60	1.5	0.75	0.833	2
	62	1.451	0.483	0.806	3
	67	1.343	0.335	0.746	4
	74	1.216	0.152	0.675	8
70	87	1.436	0.718	0.804	2
	86	1.453	0.484	0.795	3
	94	1.329	0.332	0.744	4
	101	1.237	0.154	0.693	8

7.4 Discussion

According to the results in Figs. 3–6, it is found that the makespan of the ECSA is reduced by (7.9%) and (2.1%) about NGA and GA respectively. The speedup of the ECSA is improved by (8.5%) and (2.1%) for NGA and GA respectively. The efficiency of the ECSA is improved by (8.5%) and (2.1%) for NGA and GA respectively. The throughput of the ECSA is improved by (9.2%) and (2.6%) about NGA and GA respectively. According to the results in Figs. 7–10, it is found that the makespan of the ECSA is reduced by (8.8%), (7.7%), (3.4%) about WOA, GSA, HHG, respectively. The speedup of the ECSA is improved by (9.7%), (7.9%), and (3.5%) about WOA, GSA, and HHG, respectively.

The efficiency of the ECSA is improved by (9.6%), (7.8%), and (3.4%) about WOA, GSA, and HHG, respectively. The throughput of the ECSA is improved by (10%), (8.6%), and (3.5%) about WOA, GSA, and HHG, respectively.

8 Conclusion and Future Work

To get near-optimal results for the problem of scheduling tasks in a cloud computing environment, efficient strategies for the optimal mapping of the tasks are required. In this research, we propose a novel efficient approach based on the cooperation search algorithm called the Efficient Cooperation Search Algorithm (ECSA) to solve the problem of scheduling tasks in a cloud computing environment. The system is made up of a limited number of fully connected heterogeneous processors. The comparison of the algorithm has been made against the algorithms in terms of makespan, speedup, efficiency, and throughput. The comparative analysis explains that the proposed algorithm performance is significantly better in all cases, it reduces the makespan by (7.9%), (2.1%), (8.8%), (7.7%), (3.4%) about New Genetic Algorithm (NGA), Genetic Algorithm (GA), Whale Optimization Algorithm (WOA), Gravitational Search Algorithm (GSA), and Hybrid Heuristic and Genetic (HHG) respectively according to makespan. In our future work, we will develop an efficient coronavirus herd immunity optimizer algorithm for optimizing scheduling tasks in cloud computing.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Dubey, M. Kumar and S. C. Sharma, “Modified HEFT algorithm for task scheduling in cloud environment,” *Procedia Computer Science*, vol. 125, pp. 725–732, 2018.
[2] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu *et al.,* “A woa-based optimization approach for task scheduling in cloud computing systems,” *IEEE Systems Journal*, vol. 14, no. 3, pp. 3117–3128, 2020.
[3] T. Bezdan, M. Zivkovic, E. Tuba, I. Strumberger, N. Bacanin *et al.,* “Multiobjective task scheduling in cloud computing environment by hybridized bat algorithm,” *Journal of Intelligent & Fuzzy Systems*, vol. 42, no. 1, pp. 411–423, 2022.
[4] A. I. Awad, N. A. El-Hefnawy and H. M. Abdel_kader, “Enhanced particle swarm optimization for task scheduling in cloud computing environments,” *Procedia Computer Science*, vol. 65, pp. 920–929, 2015.
[5] L. Abualigah and M. Alkhrabsheh, “Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing,” *Journal of Supercomputing*, vol. 78, no. 1, pp. 740–765, 2022.
[6] S. Velliangiri, P. Karthikeyan, V. M. Arul Xavier and D. Baswaraj, “Hybrid electro search with genetic algorithm for task scheduling in cloud computing,” *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 631–639, 2021.
[7] A. Mishra, M. N. Sahoo and A. Satpathy, “H3CSA: A makespan aware task scheduling technique for cloud environments,” *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 10, pp. 1–20, 2021.
[8] A. Y. Hamed and M. H. Alkinani, “Task scheduling optimization in cloud computing based on genetic algorithms,” *Computers, Materials & Continua*, vol. 69, no. 3, pp. 3289–3301, 2021.
[9] A. Younes, A. BenSalah, T. Farag, F. A. Alghamdi and U. A. Badawi, “Task scheduling algorithm for heterogeneous multi processing computing systems,” *Journal of Theoretical and Applied Information Technology*, vol. 97, no. 12, pp. 3477–3487, 2019.

[10] Z. -K. Feng, W. -J. Niu and S. Liu, “Cooperation search algorithm: A novel metaheuristic evolutionary intelligence algorithm for numerical optimization and engineering optimization problems,” *Applied Soft Computing Journal*, vol. 98, pp. 1–27, 2021.

[11] I. Dubey and M. Gupta, “Uniform mutation and SPV rule based optimized PSO algorithm for TSP problem,” in *Proc. of the 4th Int. Conf. on Electronics and Communication Systems*, Coimbatore, India, pp. 168–172, 2017.

[12] L. Wang, Q. Pan and F. M. Tasgetiren, “A hybrid harmony search algorithm for the blocking permutation flow shop scheduling problem,” *Computers & Industrial Engineering*, vol. 61, no. 1, pp. 76–83, 2011.

[13] S. Nabi, M. Ibrahim and J. M. Jimenez, “DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing,” *IEEE Access*, vol. 9, pp. 61283–61297, 2020.

[14] B. Keshanchi, A. Souri and N. Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing,” *Journal of Systems and Software*, vol. 124, pp. 1–21, 2017.

[15] S. R. Thennarasu, M. Selvam and K. Srihari, “A new whale optimizer for workflow scheduling in cloud computing environment,” *Journal of Ambient Intelligence Humanized Computing*, vol. 12, no. 3, pp. 3807–3814, 2020.

[16] T. Biswas, P. Kuila, A. K. Ray and M. Sarkar, “Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems,” *Simulation Modelling Practice and Theory*, vol. 96, pp. 1–21, 2019.

[17] M. Sulaiman, Z. Halim, M. Lebbah, M. Waqas and S. Tu, “An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment,” *Journal of Grid Computing*, vol. 19, no. 1, pp. 1–31, 2021.