Journal of Theoretical and Applied Information Technology
31st December 2025. Vol.103. No.24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMISED CUCKOO-INSPIRED DISCRETE SYMBIOTIC ORGANISMS SEARCH STRATEGY FOR TASK SCHEDULING IN A CLOUD COMPUTING ENVIRONMENT

SULEIMAN SA'AD[1&2*], ABDULLAH MUHAMMED[2*], MOHAMMED ABDULLAHI[3], AZIZOL ABDULLAH[4], FAHRUL HAKIM AYOB[5]

[1,2,4,5] Department of Communication Technology and Networking, Universiti Putra Malaysia, 43400 UPM, Serdang Selangor, Malaysia
[1] Department of Information Technology, Modibbo Adama University of Technology Yola 640231, Yola Nigeria
[3]Department of Computer Science, Ahmadu Bello University Zaria 810107, Zaria Nigeria

E-mail: [1]suleimanu@mau.edu.ng, [2]abdullah@upm.edu.my, [3]abdullahilwafu@abu.edu.ng
[4]zizol@upm.edu.my, [5]fahrul@upm.edu.my,

ABSTRACT

The cloud computing paradigm is rapidly expanding as there is a notable shift away from other distributed computing approaches and traditional IT infrastructures. As a result, optimised task scheduling techniques have become essential for managing the growing complexity of cloud environments. In cloud computing, numerous tasks must be allocated to a limited number of diverse virtual machines, aiming to reduce the imbalance between local and global search spaces while optimising system utilisation. Task scheduling is an NP-complete problem, meaning there is no exact solution, and we can only achieve near-optimal results, especially when dealing with large-scale tasks in cloud computing. This paper introduces an optimised strategy, the Cuckoo-based Discrete Symbiotic Organisms Search (C-DSOS), for optimal task scheduling in cloud environments. The method is based on the Standard Symbiotic Organism Search (SOS), a nature-inspired metaheuristic optimisation algorithm used for numerical optimisation problems. SOS mimics the symbiotic relationships seen in ecosystems, such as mutualism, commensalism, and parasitism. The proposed technique was evaluated using the CloudSim toolkit simulator, and experimental results showed that C-DSOS outperforms the benchmarked Simulated Annealing Symbiotic Organism Search (SASOS) algorithm, commonly used in task scheduling. C-DSOS demonstrated a better convergence rate, especially in larger search spaces, making it well-suited for cloud task scheduling. Additionally, a t-test analysis revealed that C-DSOS is statistically significant compared to SASOS, particularly in scenarios involving larger search spaces.

Keywords: *Cloud computing, Scheduling, Metaheuristic, C-DSOS, Optimisation*

1. INTRODUCTION

Cloud computing is a modern advancement in computing, offering unrestricted access to IT resources across diverse sectors such as education, healthcare, business, mobile systems, smart technologies, and environmental computing [1], [2]. This has led to the rapid adoption of cloud computing, serving as a powerful paradigm for acquiring IT services and infrastructure through a pay-per-use model [3]. This model eliminates the need for organisations to make significant investments in IT infrastructure or software licenses.

Cloud services are categorized into three primary models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [4], [5]. These services are delivered using virtualization and containerization, making cloud resources flexible and scalable, which creates the perception of virtually unlimited resources. Users pay based on their actual usage through a pay-per-use model, a flexible pricing structure that has encouraged the migration of IT services to the cloud [5]. This paper focuses on IaaS, where computing resources are provided as services. Users subscribe to virtual machines (VMs) for task execution, and efficient task scheduling on these VMs is crucial for optimal resource utilisation.

Task scheduling in cloud computing has been widely studied but remains an NP-hard challenge [6]. The IaaS layer provides users with virtual resources through a network of servers, offering both hardware and software to deliver

flexible computational capabilities. Within this layer, the resource management subsystem is responsible for scheduling tasks for execution on VMs. Given the heterogeneous and dynamic nature of VMs, efficient task mapping is essential. Since there is no exact algorithm for solving NP-complete problems, heuristic methods are typically used to find near-optimal scheduling solutions [7], [8], [9], [10]. The main objective of task scheduling algorithms is to minimize parameters like imbalance, makespan, execution time, and cost. The algorithms determine which VM should execute a task, taking into account load balancing to reduce makespan and enhance resource efficiency [11], [12], [13].

Approaches to task scheduling typically fall into two categories: heuristic and metaheuristic methods. Heuristic methods use predefined rules to identify solutions, but their effectiveness depends on the problem size and the rules applied, often resulting in high operational costs and suboptimal solutions [14], [15].

Metaheuristic algorithms, on the other hand, optimise solutions by exploring a pool of candidate solutions rather than a single solution, offering superior performance. Popular metaheuristic methods for task scheduling include Genetic Algorithms [16] and Particle Swarm Optimisation [17]. Other techniques, such as Ant Colony Optimisation [18], [19], League Championship Algorithm [11], BAT Algorithm [20], [21], and Symbiotic Organisms Search (SOS) [12], [13], have also gained prominence. The SOS algorithm, inspired by the symbiotic relationships in ecosystems, has shown competitive performance compared to other metaheuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimisation (PSO) [22]. Since its introduction, SOS has been applied to practical optimisation problems across various domains [12], [23], [24], [25], [26], [27].

Hybridizing metaheuristic algorithms with other strategies can further improve solution quality and convergence speed. This can be achieved by combining metaheuristics with heuristic techniques or modifying operators [28], [29]. To date, no research has explored enhancing the SOS algorithm in terms of convergence speed and solution quality.

This paper introduces a fitness function model designed to calculate the degree of imbalance by optimising VM utilisation. The proposed task scheduling strategy, Cuckoo-based Discrete Symbiotic Organisms Search (C-DSOS), integrates the Cuckoo Search (CS) method with the SOS variant by [30]. While SOS performs global search efficiently, CS is employed to enhance local search capabilities, resulting in both exploratory and exploitative power. The goal is to achieve optimal task schedules by minimizing the degree of imbalance between the global and local search spaces.

Key contributions of this paper include:

i. The introduction of an objective function for task scheduling on VMs, optimising VM utilisation to reduce imbalance in the search space.

ii. The hybridization of SOS with Cuckoo Search, using Levy flight for local search to find optimal solutions.

iii. Implementation of the proposed C-DSOS strategy in CloudSim.

iv. A performance comparison between SASOS and the proposed C-DSOS method, focusing on the degree of imbalance.

The remainder of the paper is structured as follows: Section 2 reviews related work on metaheuristic algorithms for task scheduling and SOS. Section 3 explains the proposed algorithm design. Section 4 presents problem formulation. Section 5 discusses simulation results. Finally, Section 6 concludes the paper.

2. RELATED WORKS

Several metaheuristic algorithms, such as those proposed by [11], [13], [27], [31], [32], [33], [34], [35], [36], [37], [38], [39], have been used to solve task assignment challenges, focusing on reducing makespan and response time. These approaches have proven effective in optimally mapping workloads to resources, leading to lower computation costs, improved quality of service, and enhanced resource utilisation. Among these, Ant Colony Optimisation (ACO), Genetic Algorithm (GA), and their variants are the most widely used nature-inspired, population-based methods in cloud computing. However, Particle Swarm Optimisation (PSO) consistently outperforms GA and ACO in various scenarios [40] due to its faster execution and simpler implementation.

PSO has been extensively studied for workflow scheduling, with a focus on reducing communication costs and minimizing makespan [17], [41], [42], [43]. Its effectiveness in scheduling independent tasks within the cloud has also been explored [44], [45], [46], where it has achieved minimal makespan. Additionally, enhanced and

hybrid versions of PSO, as proposed by [44], [45], [47], have shown better results for cloud task scheduling compared to ACO and GA. More recently, discrete versions of the Symbiotic Organisms Search (SOS) algorithm have been applied to task scheduling in cloud environments [12], [13], [27], [34], demonstrating that SOS outperforms other nature-inspired metaheuristics and their popular variants in terms of performance.

2.1 Symbiotic organisms search

The Symbiotic Organism Search (SOS) algorithm is inspired by the symbiotic relationships observed among organisms in an ecosystem. In this framework, each organism represents a potential solution to an optimisation problem, with its position reflecting its state within the solution space. The organisms adjust their positions based on ecological interaction models, including mutualism, commensalism, and parasitism, which mirror natural dynamics.

In the algorithm's first phase, mutualism is applied, where both interacting organisms mutually benefit, enhancing their positions in the solution space. In the second phase, commensalism occurs, where one organism benefits while the other remains unaffected, allowing for further refinement of the solution space.

The third phase introduces parasitism, where one organism gains at the expense of the other, helping to further optimise the solution by promoting positions that are more advantageous. Through these interactions, the fittest organisms—those with better solutions—thrive, while weaker solutions are progressively eliminated. The best-performing organisms are identified based on their cumulative advantages gained from all three phases.

These phases are applied iteratively to the population of organisms (candidate solutions) until predefined stopping criteria are met. The fitness of each organism is determined by how well it adapts to the problem's constraints and objectives. For a detailed description of the SOS algorithm, see Algorithm 1.

Algorithm 1: Standard – Symbiotic Organisms Search

```
INPUT:
  - Ecosize (Size of the ecosystem)
  - Initial population (Initial candidate solutions)
  - Stopping criteria (Condition to stop the algorithm)
OUTPUT:
  - Optimal solution
1: Initialize the ecosystem with the given Initial population
2: Initialize a variable to keep track of the number of iterations (iteration_count)
3: Repeat the following steps until the stopping condition is met:
4:    For each organism in the ecosystem:
5:       Determine the best organism in the current ecosystem
6:       Apply the Mutualism Phase to the organism
7:       Apply the Commensalism Phase to the organism
8:       Apply the Parasitism Phase to the organism
9:    End For
10:  Update the iteration_count by 1
11: Until stopping condition is exceeded (e.g., a maximum number of iterations is reached)
12: Return the best organism found as the optimal solution
```

The SOS algorithm shares several similarities with other nature-inspired optimisation methods. It uses a population of organisms to represent possible solutions and applies operators to guide the search process. A selection mechanism is employed to retain the best solutions, requiring the population size and stopping criteria to be predefined before the search begins. Unlike algorithms such as Particle Swarm Optimisation (PSO), which rely on parameters like inertia weight, social, and cognitive factors, or Genetic Algorithm (GA), which uses crossover and mutation, SOS does not require algorithm-specific parameters. Incorrect tuning of these parameters in other algorithms can lead to suboptimal results.

The SOS algorithm starts by generating a population of organisms, forming an ecosystem. These organisms' positions are updated through the algorithm's three phases. In a D-dimensional solution search space, a population of n organisms is represented as $X = \{X_1, X_2, X_3,...,X_n\}$, where the position of each organism i is defined as $X_i = \{X_{i1}, X_{i2}, X_{i3}, ..., X_{id}\}$. A fitness function is also established to evaluate the quality of each solution. Each organism in the population represents a task schedule, encoded as a vector of dimensions $1xn$, where n is the number of tasks. The elements of this vector are natural numbers within the range $[0, m - 1]$, where m represents the number of available resources for task execution.

The best-searched position across all organisms up to the current point is identified as X^{best}. Since task scheduling is a discrete optimisation problem and SOS was originally designed for continuous optimisation, a mapping function was introduced by [12], [13], [34] to convert continuous positions into discrete ones. This mapping function is defined in Equation (1). The fitness value of each

organism is iteratively evaluated using their positions in the three phases of the SOS algorithm, as described in the following subsections.

$$X_i = round(X_i mod m) \quad [1]$$

Where m is the number of resources for executing the submitted tasks.

Mutualism Phase: In this phase, the individual X_i represents the *ith* organism within the ecosystem. A random organism X_j is selected from the population to interact with X_i, where $i \neq j$. The objective of this interaction is to provide mutual benefits to both organisms, enhancing their chances of survival in the ecosystem. The updated candidate solutions for X_i and X_j are calculated based on the procedures outlined in Equations (2) and (3) [30].

$$X_i^* = X_i + U(0,1) * (X^{best} + MV * \alpha) \quad [2]$$

$$X_j^* = X_j + U(0,1) * (X^{best} + MV * \beta) \quad [3]$$

$$MV = \frac{1}{2}(X_i + X_j) \quad [4]$$

For values of i ranging from *1* to the *ecosize* (excluding cases where *(j = i)), (U(0, 1))* represents a vector of uniformly distributed random numbers between *0* and *1*. The mutual relationship vector between X_i and X_j, denoted as *MV* and defined in Equation (4), is involved. In this context, X^{best} refers to the organism with the highest fitness value. The factors α and β control the benefits from the interaction between organisms X_i and X_j. Since organisms can gain varying degrees of advantage from mutual interactions, α and β are determined probabilistically, each taking values of either *1* or *2*. A value of *1* indicates a lesser benefit, while a value of *2* signifies a greater benefit.

If the new candidate solutions have better fitness values than the old ones, they replace the original solutions. In such cases, X^*_i and X^*_j replace X_i and X_j, respectively, in the next generation of the ecosystem. If the fitness values of X^*_i and X^*_j do not improve, the original solutions are retained, and X_i and X_j proceed to the next generation. This scenario is captured by Equations (5) and (6) [30].

$$X_i = \{X_i^* \; if \; f(X_i^*) > f(X_i) \; or \; (X_i \; if \; f(X_i^*) \leq f(X_i)\} \quad [5]$$

$$X_j = \{X_j^* \; if \; f(X_j^*) > f(X_j) \; or \; (X_j \; if \; f(X_j^*) \leq f(X_j)\} \quad [6]$$

In this context, *f(.)* denotes the fitness evaluation function.

Commensalism Phase: During this phase, the *ith* organism in the ecosystem, represented by X_i, randomly selects another organism, X_j, where i does not equal j, to interact with. In this interaction, X_i benefits from the relationship, while X_j may either remain unaffected or experience a slight gain or loss. The mathematical representation of this interaction is provided in Equation (7) [30].

$$X_i^* = X_i + U(-1,1) * (X^{best} + X_j) \quad [7]$$

In this scenario, *U(-1, 1)* represents a vector consisting of uniformly distributed random numbers between *-1* and *1*. Similar to the mutualism phase, X^{best} refers to the organism with the best fitness value. If the fitness value of $f(X_i^*)$, denoted as $f(X_i^*)$, exceeds the current fitness value of X_i, represented as $f(X_i)$, then X_i is updated to X_i^*, as determined by Equation 7. The process for updating X_i is outlined in Equation 8 [30].

$$X_i = (X_i^* \; if \; f(X_i^*) > f(X_i) \; or \; (X_i \; if \; f(X_i^*) \leq f(X_i) \quad [8]$$

Parasitism Phase: In this phase, a "parasite vector" is created by cloning the *ith* organism, X_i, and modifying it using a randomly generated number. Afterward, a random organism X_j is selected from the ecosystem, and the fitness values of both the parasite vector and X_j are evaluated. If the parasite vector has a better fitness value than X_j, it replaces X_j. However, if X_j has a superior fitness value, it continues to the next generation of the ecosystem, and the parasite vector is discarded. The update of X_j follows the relationship described in Equation 9 [30].

$$X_j = (PV \; if \; f(PV) > f(X_j) \; or \; (X_j \; if \; f(PV) \leq f(X_j) \quad [9]$$

Here, PV represents the parasite vector.

2.2 Cuckoo search algorithm

The Cuckoo Search (CS) algorithm is a nature-inspired metaheuristic that mimics the aggressive breeding behavior of cuckoo birds, known for their unique reproductive strategies. A key feature of CS is the Levy flight mechanism, which initiates with an exploratory phase in the search space and transitions into an exploitation phase. This allows CS to strike an effective balance between local and global exploration. The algorithm maintains a strong equilibrium between searching for a variety of solutions and exploiting the most promising ones. The Levy flight mechanism is mathematically represented in Equation 10, as detailed by [48].

$$X_i^t = X_i^{t+1} + \alpha\, \Theta Levy(\lambda) \quad [10]$$

In this context, X^t_i represents the new solution, while X_i^{t+1} denotes the current solution. The term $\alpha\ \Theta\ Levy(\lambda)$ refers to the transition probability involved in the search process.

2.3 Cuckoo-based discrete symbiotic organism search strategy

The basic Symbiotic Organism Search (SOS) algorithm includes several important features, such as being parameterless and having a well-organised sequence of three phases: mutualism, commensalism, and parasitism. Since it is parameterless, there is no need for complex fine-tuning of control parameters, making its implementation easier. The mutualism and commensalism phases primarily focus on local searches, while the parasitism phase handles the global search aspect. In the parasitism phase, a random organism (X_k) from the ecosystem replaces the global best candidate (X^{best}) to introduce diversity into the solution space. Furthermore, incorporating Levy Flight in the mutualism phase improves the algorithm's performance by speeding up convergence, avoiding local optima traps, and addressing load balancing issues, as illustrated in Equations 11 to 14.

$$S_1(p) = X_i + (eq.10) * r' X_k - \left(X_i + X_j/2\right) f_1 [11]$$

$$S_2(p) = X_i + (eq.10) * r'' X_k - \left(X_i + X_j/2\right) f_2 \quad [12]$$

$$X_I^*(q) = \lceil S_1(p) \rceil \, mod\ m + 1 \quad [13]$$

$$X_j^*(q) = \lceil S_2(p) \rceil \, mod\ m + 1 \quad [14]$$

In this context, $X_I^*(q)\ and\ X_j^*(q)$ represent the updated positions of the *ith* and *jith* members of the ecosystem, with $i \neq j$, X_j is the randomly chosen organism during the *ith* iteration, also ensuring $\backslash\ i \neq j$. The parameters $f_1\ and\ f_2$, randomly selected from the set {1, 2}, represent the benefit factor derived from mutual interaction. Additionally, r' and r'' are uniformly generated random numbers within the interval (0,1), and ⌈⌉ denotes the upper bound function.

Furthermore, Equations 15 to 19 demonstrate how the commensalism phase, working alongside the mutualism phase, performs local search operations by restricting the random value range from *[-1, 1]* to *[0.4, 0.9]* (Do et al., 2018) This modification enhances the algorithm's convergence speed and reduces load imbalance.

$$r' \leftarrow rand[0.4,0.9] \quad [15]$$

$$S_3(p) = X_i + r'\left(X^{best} - X_j\right) \quad [16]$$

$$X_i^*(q) = \lceil S_3(p) \rceil mod m + 1 \quad [17]$$

$$S_4(p) = r_4 X_i \quad [18]$$

$$X^p(q) = \lceil S_4(p) \rceil mod m + 1 \quad [19]$$

In Equation 15, as referenced by [49], r' is a random number uniformly generated within the range of 0.4 to 0.9. Although local search offers advantages, it is important to note that it may not always produce optimal results due to the risk of premature convergence, potentially trapping the algorithm in local optima. This challenge can hinder task scheduling efforts that aim to reduce imbalance.

Given the effectiveness of SOS in solving NP-hard problems, as highlighted in previous studies, modifications have been introduced in both the mutualism and commensalism phases. Specifically, Levy Flight is incorporated into the local search component of the discrete Symbiotic Organisms Search (CDSOS) algorithm. This modification enables the CDSOS strategy to achieve a balanced exploration of both local and global solution spaces, resulting in improved outcomes. This adaptation leads to reduced computational time and, consequently, minimizes task scheduling

Journal of Theoretical and Applied Information Technology
31st December 2025. Vol.103. No.24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

imbalances. Therefore, this study employs the CDSOS task scheduling strategy to optimise task allocation to virtual machines.

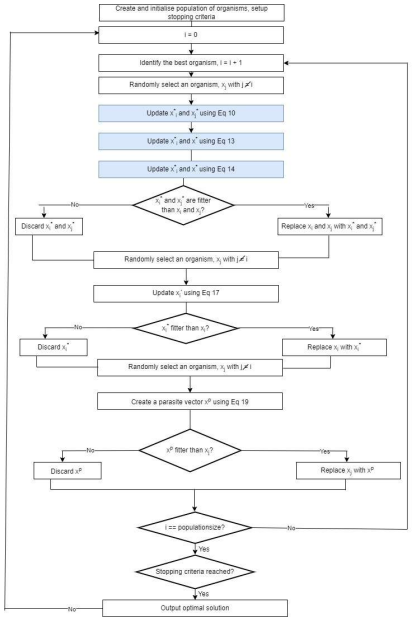

Figure 1: The flowchart of the proposed CDSOS strategy

Therefore, the hybrid strategy is designed to reduce the level of imbalance within the search space, thereby enhancing the convergence rate and improving load balancing. As a result, the search process becomes more efficient, leading to faster retrieval of search results due to the increased effectiveness of the proposed approach. Figure 1 presents the flowchart of the proposed CDSOS task scheduling strategy, outlining the scheduling process that generates a task scheduling sequence with minimal imbalance at the final stage. Algorithm 2 provides the pseudo-code for the CDSOS task scheduling strategy.

Algorithm 2 Cuckoo-based Discrete Symbiotic Organism Search Algorithm

Create and Initialise the Population of Organisms in the Ecosystem $X = \{x_1, x_2, x_3 \ldots y_N\}$
Setup stopping criteria
$iteration\_number \leftarrow 0$
$x^{best} \leftarrow 0$
Do
  $iteration\_number \leftarrow iteration\_number + 1$
  $i \leftarrow 0$
  Do
  $i \leftarrow i + 1$
  for $j = 1$ *to* N **do**
    if $f(x_j) > f(x^{best})$ **then**
      $x^{best} \leftarrow y_j$
    end if
  end for
  //**Mutualism phase**
  Randomly select x_j with $i \neq j$
  Update x_i^* and x_j^* using Equations 13 and 14
  if $f(x_i^*) > f(x_i)$ **then**
    $x_i \leftarrow x_i^*$
  end if
  if $f(x_j^*) > f(x_j)$ **then**
    $x_j \leftarrow x_j^*$
  end if
  //**Commensalism phase**
  Randomly select x_j with $i \neq j$
  $r_3 \leftarrow$ using Equation 15
  Update x_i^* according Equation 17
  if $f(x_i^*) > f(x_i)$ **then**
    $x_i \leftarrow x_i^*$
  end if
  //**Parasitism phase**
  Randomly select x_j with $i \neq j$
  Create a parasite vector x^p from x_i using random number according Equation 19
  if $f(x^p) > f(x_j)$ **then**
    $x_j \leftarrow x^p$
  end if
  While $i <= N$ 2
  While stopping condition is not true

3. PROBLEM FORMULATION

The task is structured to align with the objective function, ensuring that each task design solution is evaluated fairly based on the overall requirements. This evaluation is carried out through the use of the objective function, which plays a vital role in achieving a polynomial time approximation within the framework of the proposed CDSOS strategy. The objective function is aimed at producing results that closely approach the optimal value, thereby effectively indicating how well the proposed strategy minimizes imbalance. Essentially, scheduling involves the efficient allocation of tasks to selected virtual machines with the primary goal of reducing the degree of imbalance.

Consider a set of independent tasks that need to be scheduled for execution on a set of heterogeneous virtual machines (VMs). The VMs are denoted as $V = \{v_k \vee m \geq k \geq 1\}$, where m represents the total number of VMs. Similarly, the set of tasks is represented as $T = \{t_i \vee n \geq i \geq 1\}$, where n is the total number of tasks. The goal is to efficiently assign each task $t_i \forall i = \{1,2, \ldots, n\}$ to an appropriate cloud virtual machine $v_k \forall k = \{1,2, \ldots, m\}$ in order to minimize the imbalance. The

execution time of task t_i processed on virtual machine v_k is calculated using Equation 20.

$$exe_k = \frac{\sum x_{ik} * t_{ik}}{npe_k} \times v_{mipsk} \quad [20]$$

$$i \in Task, k \in virtualmachine$$

In this context, exe_k represents the execution time of a task assigned to virtual machine v_k. The variable x_{ik} is set to *1* if task i is allocated to virtual machine k, and *0* otherwise. t_{ik} refers to the amount of the task allocated to v_k, while v_{mipsk} denotes the length of a task in Million Instructions (MIs), and v_{mipsk} also represents the speed of v_k in Millions of Instructions per Second (MIPS). Additionally, npe signifies the number of processing elements. If multiple virtual machines $v_k \forall k = 1,2,\dots,m$ process task $t_i \forall k = 1,2,\dots,n$, the total execution time for the task processed across all virtual machines v_k is calculated using Equation 21.

$$T_{exek} = \sum exe_k \quad [21]$$

$$\forall i = \{1,2,\dots,n\} k = \{1,2,\dots,m\}$$

As a result, Equation 22 defines the objective function for the task scheduling problem, with the goal of minimising the degree of imbalance (DI).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad [22]$$

$$\forall i = \{1,2,\dots,n\} k = \{1,2,\dots,m\}$$

The Degree of Imbalance (DI) equation, as proposed by (Babu & Krishna [50], is a useful metric for evaluating the performance of any task scheduling solution. It enables the assessment of how effectively the solution tackles real-world NP-hard scheduling problems. Moreover, this model can be easily integrated into nature-inspired metaheuristic algorithms to aid in the search for optimal solutions.

4. SIMULATION AND RESULTS

The proposed task scheduling strategy is implemented using the CloudSim simulator, as introduced by [51]. To support the simulation process, the datacentre broker policy in CloudSim is extended. The selection of properties for virtual machines, hosts, and tasks adheres to the standards commonly used in numerous previous studies. These properties are detailed in Table 1.

Table 1: Datacentre parameter settings

Entity Type	Parameter	Value
Datacentre	No. of datacentre	2
	No. of host	2
	Host RAM	20 GB
	Storage	1 TB
	Bandwidths	10 GB/s
	Accumulated host processing power	1000000 MIPS
	Lengths	100 - 1000 MIs
Cloudlets	No. of Cloudlets (tasks)	100 - 1000, NASA Ames iPSC/860 and HPC2N workloads
Virtual Machine (VM)	No. of VMs	25
	VMs Monitor	Xen Linux
	Operation System	0.5 GB 10 GB 1 GB/s
	RAM	500 - 5000 MIPS
	Storage Bandwidth	1
	VMs processing power	Time-shared
	Processing element	
	VM Policy	

In this context, particular emphasis is placed on the selection of virtual machine (VM) sizes, with an investigation into whether VM dimensions could impact the proposed task scheduling strategy. Consistent with standard practices in cloud computing task scheduling, each task is characterized by its length, which varies across tasks, making each one distinct. Hosts are equipped with a storage capacity of one terabyte (1TB) to support VMs in each datacentre. Several prior studies on task scheduling in cloud computing, such as those by [12], [13], [30], [34], [37], [38], [52], have adopted these property settings to assess

their task scheduling and resource provisioning methods.

The inertia weight values and coefficient factors, as shown in Table 2, are chosen based on their usage in previous research, including works by [29], [30], [53], [54], [55].

Table 2: Parameter settings for the scheduling approaches [56]

Algorithm	Parameter	Value
SOS	Number of organisms	100
	Number of iterations	1000
SA	Initial temperature, *Finit*	10
	Final temperature, *Ffinal*	0
	Cooling rate, δ	0.9

The task instances are generated from synthetic workloads, and for comparison, the Simulated Annealing Symbiotic Organism Search (SASOS) algorithm, as introduced by [30], is utilised. SASOS serves as a benchmark metaheuristic algorithm to evaluate the improvements introduced in the mutualism and commensalism phases of the DSOS algorithm, particularly regarding local search efficiency. SASOS is applied within a cloud computing environment to tackle the heterogeneous task scheduling problem and optimise the degree of imbalance. Thus, this paper adapts the algorithm as an enhancement to the standard SOS algorithm, aimed at reducing premature convergence and minimizing the degree of imbalance. A comparative analysis between this algorithm and the proposed CDSOS strategy provides valuable insights into the latter's efficiency.

Moreover, results from a simulation involving the execution of the proposed CDSOS approach alongside a comparative algorithm over 10,000 iterations demonstrate the effectiveness of the proposed strategy, especially in environments with heterogeneous virtual machines (VMs). This approach highlights the importance of achieving varying degrees of imbalance. Consequently, the simulation results are presented by examining maximum, minimum, and average values. The comparison with SASOS focuses primarily on evaluating the degree of imbalance. For a more detailed analysis, refer to Figures 2 to 5, while specific simulation results are provided in Tables 3 to 6.

Table 3: Comparison of degree of imbalance obtained by SASOS and CDSOS for normal distribution dataset

Task Size	SASOS			CDSOS				
	max	min	avg	max	min	avg	t-value	p-value
100	19.91	16.99	17.35	19.91	16.50	17.98	6.3475	3.62E-08
200	19.91	18.26	18.77	19.91	17.49	18.66	-1.2621	0.21200
300	19.91	19.03	19.15	19.91	16.99	18.75	-3.0073	0.00389
400	19.91	19.26	19.38	19.91	16.66	18.75	-4.5462	2.83E-05
500	19.91	19.47	19.52	19.91	17.10	19.15	-3.3504	0.00142
600	19.91	19.55	19.61	19.91	18.74	19.35	-4.9272	7.31E-06
700	19.91	19.62	19.66	19.91	18.00	19.41	-3.7358	0.00043
800	19.91	19.66	19.71	19.91	18.43	19.36	-5.9218	1.83E-07
900	19.91	19.70	19.73	19.91	18.43	19.41	-4.4924	3.41E-05
1000	19.91	19.74	19.77	19.91	18.34	19.53	-4.2476	7.92E-05

Table 4: Comparison of degree of imbalance obtained by SASOS and CDSOS for left-half distribution dataset

Task Size	SASOS			CDSOS				
	max	min	avg	max	min	avg	t-value	p-value
100	18.59	16.55	17.41	18.59	15.77	17.63	1.6660	0.10110
200	19.44	18.53	18.75	19.44	17.32	18.54	-1.8174	0.07433
300	19.44	19.05	19.19	19.44	16.24	18.49	-4.6371	2.06E-05
400	19.65	19.31	19.38	19.65	17.02	18.80	-4.8204	1.07E-05
500	19.67	19.42	19.52	19.67	18.04	19.13	-4.6988	1.65E-05
600	19.75	19.54	19.59	19.75	17.86	19.23	-4.5002	3.32E-05
700	19.80	19.61	19.66	19.80	18.80	19.37	-5.5718	6.84E-07
800	19.91	19.62	19.7	19.91	18.77	19.46	-4.7656	1.30E-05
900	19.91	19.70	19.74	19.91	18.40	19.45	-4.8315	1.03E-05
1000	19.91	19.73	19.76	19.91	18.33	19.53	-3.9145	0.00024

Journal of Theoretical and Applied Information Technology
31st December 2025. Vol.103. No.24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 5: Comparison of degree of imbalance obtained by SASOS and CDSOS for right-half distribution dataset

Task Size	SASOS			CDSOS				
	max	min	avg	max	min	avg	t-value	p-value
100	19.91	16.01	17.07	19.91	15.75	17.75	4.6624	1.88E-05
200	19.91	18.40	18.66	19.91	15.62	18.44	-1.2319	0.22300
300	19.91	18.96	19.1	19.91	17.15	18.69	-3.5595	0.00075
400	19.91	19.21	19.34	19.91	16.51	19.04	-2.2903	0.02566
500	19.91	19.36	19.48	19.91	16.22	19.19	-2.3254	0.02357
600	19.91	19.49	19.56	19.91	17.88	19.14	-4.1277	0.00012
700	19.91	19.56	19.63	19.91	18.37	19.37	-3.8935	0.00026
800	19.91	19.63	19.67	19.91	18.52	19.36	-4.0421	0.00016
900	19.91	19.66	19.72	19.91	18.50	19.38	-4.6156	2.2E-05
1000	19.91	19.71	19.76	19.91	18.67	19.57	-3.3509	0.00142

Table 6: Comparison of degree of imbalance obtained by SASOS and CDSOS for uniform distribution dataset

Task Size	SASOS			CDSOS				
	max	min	avg	max	min	avg	t-value	p-value
100	19.91	17.04	17.37	19.91	16.57	17.83	5.04960	4.69E-06
200	19.91	18.40	18.70	19.91	17.33	18.62	-0.72619	0.47060
300	19.91	19.00	19.15	19.91	17.62	18.71	-4.23180	8.36E-05
400	19.91	19.22	19.35	19.91	17.85	18.91	-4.12710	0.00012
500	19.91	19.41	19.49	19.91	18.28	19.20	-3.68910	0.00050
600	19.91	19.50	19.58	19.91	17.74	19.19	-3.88510	0.00027
700	19.91	19.58	19.65	19.91	16.80	19.22	-3.72020	0.00045
800	19.91	19.59	19.68	19.91	18.21	19.32	-4.30080	6.6E-05
900	19.91	19.70	19.73	19.91	18.31	19.29	-5.66680	4.8E-07
1000	19.91	19.71	19.75	19.91	18.46	19.46	-4.26180	7.5E-05

The Degree of Imbalance (DI) evaluates whether the proposed CDSOS task scheduling strategy can effectively balance tasks across heterogeneous virtual machines, thereby minimizing the DI. A lower DI value signifies that the CDSOS approach outperforms the SASOS algorithm. Figures 2 to 5 depict the performance of the CDSOS method in terms of achieving a minimum DI for all scheduled tasks compared to the SASOS algorithm. Furthermore, the figures highlight that the CDSOS approach consistently achieves a lower DI for each task distribution. This improvement is largely attributed to the incorporation of the cuckoo search method, which enhances the local search process of SASOS, enabling more effective task distribution decisions and optimising resource usage through efficient load balancing.

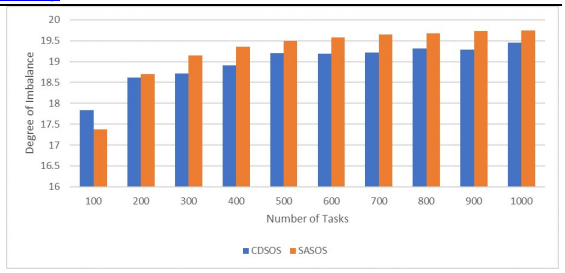

Figure 2: Degree of Imbalance for Normal Distribution Dataset

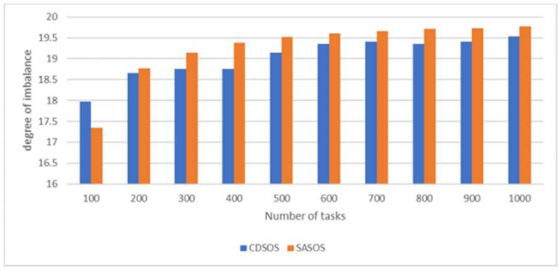

Figure 3: Degree of Imbalance for Left Half Distribution Dataset

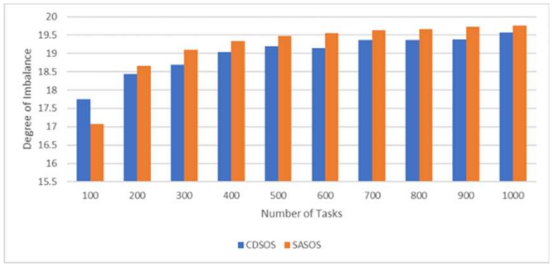

Figure 4: Degree of Imbalance for Right Half Distribution Dataset

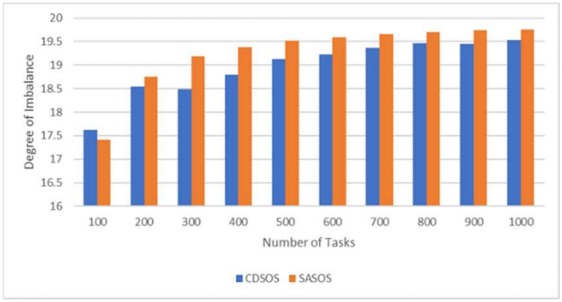

Figure 5: Degree of Imbalance for Uniform Distribution Dataset

Furthermore, the Figures demonstrate that, apart from the values for the average degree of imbalance, the number of virtual machines considered has no significant impact on the performance of the CDSOS task scheduling strategy. Largely, the results indicate that the CDSOS strategy

outperforms the SASOS algorithm in terms of performance.

For a better understanding of the strategy's performance with 100 task instances, the results show lower performance, likely because the strategy is more effective with larger tasks. However, for task instances ranging from 200 to 1000, the Figures reveal that the performance correlates with the degree of imbalance. The CDSOS strategy effectively minimizes this imbalance compared to the SASOS algorithm, demonstrating that it efficiently distributes tasks among the most appropriate virtual machines while considering their computational needs. This leads to a significant increase in the convergence rate and ensures an efficient distribution of tasks, reducing imbalance.

Additionally, a two-sample t-test was conducted to analyse the data, with Tables 3-6 displaying the t-values and p-values for all distributions. In each case, the p-values are lower than the alpha value of 0.05, confirming a significant improvement of the CDSOS strategy over the SASOS benchmark.

5. CONCLUSION

Task scheduling in a cloud computing environment is an NP-hard problem, where cloud performance is significantly influenced by the efficiency of task scheduling strategies in managing virtual machines. Inefficient strategies often lead to higher levels of imbalance. Although the SASOS algorithm has had some success in addressing this NP-hard problem, it still struggles with a higher degree of imbalance, primarily due to disparities between the local and global search spaces. Like other metaheuristics, SASOS cannot guarantee an optimal solution for every NP-hard problem, making it challenging to consistently find an optimal solution, especially as the search space expands, as is typical in cloud computing environments. This paper introduces the CDSOS strategy as a novel approach to task scheduling. It incorporates the Levy-Flight component of the Cuckoo Search algorithm into the local search phase of DSOS (specifically, the mutualism phase), thereby optimising the degree of imbalance during task scheduling. Additionally, the hybridization of DSOS enhances the strategy's ability to find improved solutions in the later stages of the search process, boosting both exploration and exploitation capabilities. Simulation results using the CloudSim simulator demonstrate that the proposed CDSOS strategy offers significant advantages in optimising the degree of imbalance compared to the benchmark SASOS algorithm.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude for the resources and services provided by Universiti Putra Malaysia (UPM), the faculty members of Computer Science and Information Technology at UPM, and the Department of Information Technology at Modibbo Adama University, Yola, Nigeria. They also extend their appreciation to the Tertiary Education Trust Fund (TETFund) in Nigeria. Additionally, this work was supported by the Malaysian Ministry of Higher Education through Grant FRGS/1/2021/ICT08/UPM/02/2.

REFRENCES:

[1] F. Pop and M. Potop-Butucaru, "ARMCO: Advanced topics in resource management for ubiquitous cloud computing: An adaptive approach," *Futur. Gener. Comput. Syst.*, vol. 54, pp. 79–81, 2016, doi: https://doi.org/10.1016/j.future.2015.07.016 .

[2] M. Mezmaz *et al.*, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, 2011, doi: 10.1016/j.jpdc.2011.04.007.

[3] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in *2012 international conference on computing, electronics and electrical technologies (ICCEET)*, 2012, pp. 877–880.

[4] P. Mell, T. Grance, and others, "The NIST definition of cloud computing," 2011.

[5] N. B. Ruparelia, *Cloud computing*. Mit Press, 2023.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. in Mathematical Sciences Series. Freeman, 1979. [Online]. Available: https://books.google.com.ng/books?id=fjxGAQAAIAAJ

[7] K. Kaur, A. Chhabra, and G. Singh, "Heuristics based genetic algorithm for scheduling static tasks in homogeneous parallel system," *Int. J. Comput. Sci. Secur.*, vol. 4, no. 2, pp. 183–198, 2010.

[8] G. Ming and H. Li, "An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing," *Int. J. Innov. Res. Comput. Commun. Eng. (An ISO Certif. Organ.*, vol. 32972, no. 2, pp. 217–223, 2012, [Online]. Available: www.ijircce.com

[9] U. Bhoi and P. N. Ramanuj, "Enhanced max-min task scheduling algorithm in cloud computing," *Int. J. Appl. or Innov. Eng. Manag.*, vol. 2, no. 4, pp. 259–264, 2013.

[10] E. U. Munir, J. Li, and S. Shi, "QoS sufferage heuristic for independent task scheduling in grid," *Inf. Technol. J.*, vol. 6, no. 8, pp. 1166–1170, 2007.

[11] S. M. Abdulhamid, M. S. A. Latiff, and I. Idris, "Tasks scheduling technique using league championship algorithm for makespan minimization in IAAS cloud," *arXiv Prepr. arXiv1510.03173*, 2015.

[12] M. Abdullahi, M. A. Ngadi, and S. M. Abdulhamid, "Symbiotic Organism Search optimization based task scheduling in cloud computing environment," *Futur. Gener. Comput. Syst.*, vol. 56, pp. 640–650, 2016, doi: 10.1016/j.future.2015.08.006.

[13] S. Sa'ad, A. Muhammed, M. Abdullahi, A. Abdullah, and F. H. Ayob, "An Enhanced Discrete Symbiotic Organism Search Algorithm for Optimal Task Scheduling in the Cloud," *Algorithms*, vol. 14, no. 7, pp. 1–24, 2021, doi: https://doi.org/10.3390/a14070200.

[14] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," *Metaheuristics Sched. Distrib. Comput. Environ.*, pp. 173–214, 2008.

[15] C. Gogos, C. Valouxis, P. Alefragis, G. Goulas, N. Voros, and E. Housos, "Scheduling independent tasks on heterogeneous processors using heuristics and Column Pricing," *Futur. Gener. Comput. Syst.*, vol. 60, pp. 48–66, 2016, doi: 10.1016/j.future.2016.01.016.

[16] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing," *2009 5th Int. Conf. Wirel. Commun. Netw. Mob. Comput.*, pp. 1–4, 2009, doi: 10.1109/WICOM.2009.5301850.

[17] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 400–407, 2010, doi: 10.1109/AINA.2010.31.

[18] S. Xue, M. Li, X. Xu, and J. Chen, "An ACO-LB algorithm for task scheduling in the cloud environment," *J. Softw.*, vol. 9, no. 2, pp. 466–473, 2014, doi: 10.4304/jsw.9.2.466-473.

[19] X. Shengjun, Z. Jie, and X. Xiaolong, "An improved algorithm based on ACO for cloud service PDTs scheduling," *Adv. Inf. Sci. Serv. Sci.*, vol. 4, no. 18, 2012.

[20] Z. Xiaolei, M. Congan, and S. Chen, "Task scheduling based on improved bat algorithm under logistics cloud service.," *Appl. Res. Comput. Yingyong Yanjiu*, vol. 32, no. 6, 2015.

[21] S. Raghavan, P. Sarwesh, C. Marimuthu, and K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud," in *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, 2015, pp. 139–144.

[22] M.-Y. Cheng, D. Prayogo, and D.-H. Tran, "Optimizing multiple-resources leveling in multiple projects using discrete symbiotic organisms search," *J. Comput. Civ. Eng.*, vol. 30, no. 3, p. 4015036, 2016.

[23] M.-Y. Cheng, D. Prayogo, and D.-H. Tran, "Optimizing Multiple-Resources Leveling in Multiple Projects Using Discrete Symbiotic Organisms Search," *J. Comput. Civ. Eng. 30(3), 04015036.*, vol. 30, no. 3, pp. 287–290, 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000512.

[24] R. Eki, F. Y. Vincent, S. Budi, and A. P. Redi, "Symbiotic Organism Search (SOS) for Solving the Capacitated Vehicle Routing Problem," vol. 9, no. 5, pp. 873–877, 2015.

[25] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Abraham, "Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 30, no. 6, pp. 1845–1863, 2018, doi: 10.1007/s00521-016-2816-4.

[26] D.-H. Tran, M.-Y. Cheng, and D. Prayogo, "A novel Multiple Objective Symbiotic Organisms Search (MOSOS) for time--cost--labor utilization tradeoff problem," *Knowledge-Based Syst.*, vol. 94, pp. 132–145, 2016.

[27] A. A. Zubair *et al.*, "A cloud computing-based modified symbiotic organisms search algorithm (ai) for optimal task scheduling," *Sensors*, vol. 22, no. 4, p. 1674, 2022.

[28] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 275–295, 2015, doi: 10.1016/j.eij.2015.07.001.

[29] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Al-Khasawneh, "Hybrid Cat Swarm Optimization and Simulated Annealing for Dynamic Task Scheduling on Cloud Computing Environment," *J. Inf. Commun. Technol.*, vol. 17, no. 3, pp. 435–467, 2018.

[30] M. Abdullahi and M. A. Ngadi, "Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment," *PLoS One*, vol. 11, no. 6, pp. 1–29, 2016, doi: 10.1371/journal.pone.0158229.

[31] S. M. Abdulhamid, M. S. Abd Latiff, S. H. H. Madni, and M. Abdullahi, "Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 279–293, 2018, doi: 10.1007/s00521-016-2448-8.

[32] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Abraham, "Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 30, pp. 1845–1863, 2018.

[33] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Al-Khasawneh, "Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment," *J. Inf. Commun. Technol.*, vol. 17, no. 3, pp. 435–467, 2018.

[34] M. Abdullahi, M. A. Ngadi, S. I. Dishing, and S. M. Abdulhamid, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," *J. Ambient Intell. Humaniz. Comput.*, vol. 14, no. 7, pp. 8839–8850, 2023.

[35] R. Eki, F. Y. Vincent, S. Budi, and A. A. N. P. Redi, "Symbiotic organism search (SOS) for solving the capacitated vehicle routing problem," *Int. J. Ind. Manuf. Eng.*, vol. 9, no. 5, pp. 873–877, 2015.

[36] D. Prasad and V. Mukherjee, "A novel symbiotic organisms search algorithm for optimal power flow of power system with FACTS devices," *Eng. Sci. Technol. an Int. J.*, vol. 19, no. 1, pp. 79–89, 2016, doi: 10.1016/j.jestch.2015.06.005.

[37] S. H. H. Madni, M. S. Abd Latiff, S. M. Abdulhamid, and J. Ali, "Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment," *Cluster Comput.*, vol. 22, pp. 301–334, 2019.

[38] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, A. Abraham, and N. M. Dankolo, "Cloud customers service selection scheme based on improved conventional cat swarm optimization," *Neural Comput. Appl.*, vol. 32, pp. 14817–14838, 2020.

[39] M. H. Nadimi-Shahraki, E. Moeini, S. Taghian, and S. Mirjalili, "Discrete Improved Grey Wolf Optimizer for Community Detection," *J. Bionic Eng.*, pp. 1–28, 2023.

[40] M. Wang and W. Zeng, "A comparison of four popular heuristics for task scheduling problem in computational grid," *2010 6th Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2010*, pp. 1–4, 2010, doi: 10.1109/WICOM.2010.5600872.

[41] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *2010 international conference on computational intelligence and security*, 2010, pp. 184–188.

[42] Q. Tao, H. Chang, Y. Yi, C. Gu, and Y. Yu, "QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm," in *2009 Eighth International Conference on Grid and Cooperative Computing*, 2009, pp. 153–159.

[43] S. N. Sivanandam and P. Visalakshi, "Scheduling workflow in cloud computing based on hybrid particle swarm algorithm," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 10, no. 7, pp. 1560–1566, 2012.

[44] P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang, "Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization," *Appl. Math. Comput.*, vol. 184, no. 2, pp. 407–420, 2007.

[45] S. N. Sivanandam and P. Visalakshi, "Dynamic task scheduling with load balancing using parallel orthogonal particle swarm optimisation," *Int. J. Bio-Inspired Comput.*, vol. 1, no. 4, pp. 276–286, 2009.

[46] C. Jing and P. Quanke, "Discrete particle swarm optimization algorithm for solving independent task scheduling," *Comput. Eng.*, vol. 34, pp. 214–218, 2008.

[47] P.-Y. Yin, S.-S. Yu, P.-P. Wang, and Y.-T. Wang, "Task allocation for maximizing reliability of a distributed system using

hybrid particle swarm optimization," *J. Syst. Softw.*, vol. 80, no. 5, pp. 724–735, 2007.

[48] A. S. Joshi, O. Kulkarni, G. M. Kakandikar, and V. M. Nandedkar, "Cuckoo Search Optimization- A Review," *Mater. Today Proc.*, vol. 4, no. 8, pp. 7262–7269, 2017, doi: 10.1016/j.matpr.2017.07.055.

[49] D. T. T. Do, D. Lee, and J. Lee, "Material optimization of functionally graded plates using deep neural network and modified symbiotic organisms search for eigenvalue problems," *Compos. Part B Eng.*, vol. 159, no. June 2018, pp. 300–326, 2018, doi: 10.1016/j.compositesb.2018.09.087.

[50] D. L. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput. J.*, vol. 13, no. 5, pp. 2292–2303, 2013, doi: 10.1016/j.asoc.2013.01.025.

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

[52] S. M. S. M. Abdulhamid *et al.*, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," *arXiv Prepr. arXiv1510.03173*, vol. 22, no. 12, pp. 8839–8850, 2022.

[53] M. Abdullahi, M. A. Ngadi, S. I. Dishing, S. M. Abdulhamid, and B. I. eel Ahmad, "An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment," *J. Netw. Comput. Appl.*, vol. 133, pp. 60–74, 2019, doi: 10.1016/j.jnca.2019.02.005.

[54] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Cluster Comput.*, vol. 20, no. 3, pp. 2489–2533, 2017, doi: 10.1007/s10586-016-0684-4.

[55] S. M. Abdulhamid, "Task Scheduling Techniques With Fault Tolerance Awareness for Cloud Computing Environment," 2016.

[56] M. Abdullahi, M. A. Ngadi, and S. I. Dishing, "Chaotic Symbiotic Organisms Search for Task Scheduling Optimization on Cloud Computing Environment," pp. 1–4, 2017.