Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Discrete Symbiotic Organisms Search Based Data-Intensive Scientific Workflow Optimization in Cloud Computing Environment

Kouidri Siham[1], Zerrouki Kadda[2], Kouidri Chaima[3]
[1]*GeCoDe Laboratory, Department of Computer Science, University of Saida Dr. Moulay Tahar, Saida 20000, Algeria (e-mail: skouidri2023@gmail.com)*
[2]*GeCoDe Laboratory, Department of Computer Science, University of Saida Dr. Moulay Tahar, Saida 20000, Algeria (e-mail: zerrouki.kadda.2023@gmail.com)*
[3]*Computer Science, University Mustapha Stambouli of Mascara, Mascara, 29000, Algeria (e-mail : chaima.kouidri@univ-mascara.dz)*

ARTICLE INFO

Received: 29 Dec 2024

Revised: 12 Feb 2025

Accepted: 27 Feb 2025

ABSTRACT

Introduction

The ability to access and use computer resources has been totally transformed by cloud computing, which provides networking, processing, and storage capabilities as needed. As cloud-based applications and services have become more popular, so too have the quantity and complexity of tasks that require efficient scheduling. Scheduling tasks and locating data in cloud systems have become crucial concerns for scientific processes that significantly depend on data placement and tasks computing. With efficient scheduling, the best use of resources is ensured while overall execution time and communication delays are decreased. As an NP-complete problem, workflow scheduling is difficult to tackle optimally with conventional techniques and is computationally expensive. Researchers are progressively tackling these problems by employing intelligent optimization methods based on natural occurrences.

Objectives

The goal of this work is to improve workflow scheduling performance in cloud systems by creating an optimization technique inspired by biology. The main goal is to reduce two important performance indicators: workflow completion time and data transfer time. The suggested method aims to enhance overall resource utilization, lower execution cost to minimize expenses, and speed up scientific computations in the cloud by effectively assigning jobs to virtual machines.

Methods

In order to manage the discrete character of cloud workflow scheduling, this study suggests using a Discrete Symbiotic Organism Search (**DSOS**) algorithm, which is a version of the symbiotic organism search technique. Iteratively evolving task assignments to generate near-optimal scheduling solutions. The approach is put into practice and tested in the popular simulation toolkit **CloudSim**, which is used to model and assess cloud computing infrastructures. To confirm the **DSOS** algorithm's efficacy and efficiency in resolving the scheduling issue in dynamic cloud settings, it is contrasted with a number of well-known scheduling techniques.

Conclusions

Workflow scheduling in cloud computing is a challenging problem that the suggested Discrete Symbiotic Organism Search algorithm successfully resolves. It improves scientific computation performance and helps create more responsive cloud services by cutting down on execution time, execution cost and placing data optimally.

Keywords Cloud Computing, Workflow Scheduling, Discrete Symbiotic Organism Search (DSOS), CloudSim, Bio-inspired Optimization, Cloudlet Allocation, Data Transfer Time, Execution Time, Execution Cost.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

INTRODUCTION

A dominating paradigm in contemporary computing, cloud computing is transforming how individuals, organizations, and research institutions access and use computational resources. Shared computing resources, including servers, storage, apps, and networking infrastructure, are made available online on a pay-per-use basis through its service-oriented design. This on-demand paradigm is very attractive for a variety of applications, from enterprise computing to scientific research, because it provides several benefits, such as elastic scalability, cost-effectiveness, geographical independence, and little infrastructure administration.

Scientific workflow execution is one of the key domains where cloud computing has demonstrated enormous promise. A scientific workflow, which is frequently employed in data-intensive research fields including genomics, climate science, astrophysics, and materials engineering, is an organized depiction of a series of data-processing activities (tasks). Complex task interdependence, extensive data transfers, and varied computational needs are characteristics of these workflows. A strong infrastructure to control resource allocation and job scheduling is necessary for the effective execution of these workflows, as are substantial computational resources. Researchers can now access virtualized, on-demand computing environments, including virtual machines (VMs), dynamic storage systems, and scalable networks, thanks to cloud computing, which eliminates the need for expensive and rigid physical infrastructure [1].

Despite the advantages offered by cloud platforms, workflow scheduling remains a core challenge in scientific computing. Scheduling refers to the process of mapping workflow tasks to available computational resources in a way that optimizes certain performance metrics, such as minimizing makespan (total execution time), reducing cost, balancing load across resources, and minimizing data transfer time between tasks. However, due to the inherent complexity of workflows—such as task precedence constraints, varying resource requirements, data locality issues, and resource availability dynamics—scheduling becomes a highly combinatorial optimization problem. This problem is formally classified as NP-hard, which implies that there is no known algorithm capable of solving all instances of the problem optimally within polynomial time. The difficulty increases further in cloud environments, where factors like virtualization overhead, dynamic pricing models, and fluctuating resource performance add additional layers of complexity [2].

For large-scale workflow scheduling, researchers have resorted to heuristic and approximate approaches due to the computational impracticability of accurate optimization techniques. Due to their capacity to produce excellent, nearly ideal results in manageable computation durations, meta-heuristic algorithms have become increasingly popular among them. High-level, problem-independent algorithmic frameworks known as metaheuristics successfully direct underlying heuristics in their exploration of the solution space. There are several nature-inspired methods among them, including Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Genetic Algorithms (GA). By simulating social or natural behaviors, these algorithms iteratively enhance potential solutions.

In this context, we explore the application of a **bio-inspired meta-heuristic algorithm**, namely the **Discrete Symbiotic Organism Search (DSOS)** algorithm, for the problem of scientific workflow scheduling in cloud environments. Originally designed to **mimic symbiotic relationships in nature**—such as mutualism, commensalism, and parasitism—**Symbiotic Organism Search (**SOS**)** has proven effective for solving complex optimization problems. Its discrete adaptation, **DSOS**, is tailored to address scheduling problems where task assignments and execution sequences must be expressed in discrete terms. Unlike many traditional algorithms, DSOS benefits from **adaptive learning mechanisms and diversity preservation strategies**, which enhance its global search capability while preventing premature convergence to local optima.

The core objective of this research is to **minimize makespan , execution cost and inter-task data transfer time**, two crucial metrics that directly impact the performance and cost-effectiveness of scientific workflows in cloud settings. The DSOS algorithm is designed to explore the scheduling solution space more intelligently by simulating different symbiotic interactions and evaluating their impact on workflow performance. By iteratively refining candidate schedules, **DSOS** aims to provide high-quality solutions that outperform traditional heuristic and meta-heuristic techniques.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

To validate the effectiveness of the proposed **DSOS**-based scheduling approach, we conduct extensive simulation experiments using **CloudSim**, a widely adopted cloud simulation toolkit. CloudSim provides a flexible framework to model data centers, virtual machines, user workloads, and scheduling policies, allowing for a realistic and controlled evaluation of algorithm performance. The simulation includes **scientific workflows**, various VM configurations, and performance metrics such as average task execution time, execution cost, and data transfer latency. The **DSOS** algorithm's results are compared against established scheduling algorithms to demonstrate its competitiveness and applicability.

The rest of this paper is structured as follows: Section II explains some related works. Section III introduces our proposed approach. Section IV evaluates the performance of simulation experiments using CloudSim. The conclusion and future works are presented in Section V.

RELATED WORKS

The increasing requirement for effective processing of large-scale computer activities has led to significant academic interest in optimizing the scheduling of data-intensive scientific workflows in cloud computing. Numerous fields, including high-energy physics, astrophysics, climate modeling, and genomics, heavily rely on scientific workflows, which are made up of interconnected operations with intricate data flows. The scalable and elastic infrastructure of the cloud offers an appropriate setting for carrying out these processes. However, effective task scheduling is still a difficult and computationally demanding operation because of the dynamic nature of cloud resources, heterogeneous virtual machines, and the requirement to maximize several objectives (e.g., execution time, data transmission delay, and cost execution). The two main categories of current research on workflow scheduling algorithms in cloud systems are bio-inspired metaheuristic techniques and conventional heuristic-based approaches. Representative works from both groups are reviewed in the sections that follow, with an emphasis on their mechanisms, advantages, and disadvantages.

A. Heuristic-Based Task Scheduling Algorithms

Heuristic-based algorithms are widely used due to their simplicity, low overhead, and fast decision-making capabilities. Although these algorithms do not guarantee global optimality, they provide efficient and practical solutions for real-time scheduling in large-scale environments.

1. HEFT (Heterogeneous Earliest Finish Time):

HEFT is one of the most prominent and effective heuristics for heterogeneous computing environments. It schedules tasks based on their upward rank (a metric considering task priority and estimated execution time) and maps them to resources to minimize overall makespan. It considers both computation time and communication delay between tasks, making it suitable for workflows with data dependencies. Despite its effectiveness, HEFT may become less efficient as workflow complexity and resource heterogeneity increase [7].

2. Min-Min and Max-Min Algorithms:

These heuristics aim to optimize task-resource mappings by selecting tasks based on their minimum execution times.

- Min-Min prioritizes tasks with the smallest completion time, often leading to fast execution of short tasks but potentially causing delays for longer ones.
- Max-Min selects tasks with the maximum minimum execution time, promoting early execution of longer tasks to avoid bottlenecks. These algorithms are computationally light but may fail to maintain load balancing under highly variable workloads [6].

3. First-Come, First-Served (FCFS):

A non-preemptive scheduling strategy, FCFS assigns tasks in the order of their arrival. Although simple and fair in terms of task order, it may lead to suboptimal performance in scientific workflows, especially when large or long-running tasks block shorter ones, resulting in high average waiting and turnaround times [4].

4. Shortest Job First (SJF):

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

This algorithm schedules tasks with the shortest execution times first. While it reduces average waiting time and improves throughput, it requires prior knowledge of task execution time, which is often unavailable or unpredictable in real-world cloud environments [6].

5. Priority-Based Scheduling:

In this approach, each task is assigned a priority level, and scheduling decisions are made based on these levels. High-priority tasks are executed first, ensuring that critical operations are not delayed. However, this method risks starvation of low-priority tasks, especially in systems with frequent high-priority job arrivals [5].

B. Bio-Inspired Metaheuristic Scheduling Algorithms

To overcome the limitations of classical heuristics, researchers have adopted bio-inspired algorithms, which mimic natural and biological processes to explore complex solution spaces. These population-based, adaptive algorithms are particularly well-suited for multi-objective, dynamic, and NP-hard problems like workflow scheduling in cloud computing.

1. Genetic Algorithms (GA):

GA uses operators for mutation, crossover, and selection to mimic the course of natural evolution. When used in workflow scheduling, GA improves goals like makespan, cost, and load balancing by iteratively evolving a population of task-resource mappings over generations. Although it is very extendable and adaptable, it can be computationally demanding and may converge slowly if the parameters are not well tuned [8].

2. Particle Swarm Optimization (PSO):

Inspired by the collective behavior of birds and fish, PSO uses a swarm of particles to search for optimal solutions by updating positions based on individual and group experiences. In scheduling, particles represent possible task assignments. PSO is effective in continuous search spaces, but when applied to discrete problems like task scheduling, it often requires custom encoding and modification to maintain solution feasibility [9].

3. Ant Colony Optimization (ACO):

Based on the pheromone-laying behavior of ants, ACO uses probabilistic models to construct and improve solutions iteratively. Ants prefer paths with higher pheromone levels, leading to the discovery of efficient task sequences. ACO is particularly suitable for solving graph-based scheduling problems, such as DAG (Directed Acyclic Graph) workflows, but suffers from slow convergence and high computational overhead in large-scale scenarios [10].

4. Bee Colony Optimization (BCO):

Modeled after the foraging behavior of honeybees, BCO assigns tasks to food sources (resources) and iteratively improves solutions based on fitness feedback shared among bees. The algorithm balances exploration and exploitation effectively and is suitable for parallel task scheduling, but it may require fine-tuning of parameters like scout bee ratios and neighborhood size [11].

5. Firefly Algorithm:

Inspired by the light-emitting communication of fireflies, this algorithm treats task schedules as fireflies, with their attractiveness determined by the quality (brightness) of the schedule. Fireflies move towards brighter ones, thus converging on better solutions. FA is effective for multi-modal optimization, but similar to PSO, it requires discrete adaptation to be applied to workflow scheduling [12].

Traditional heuristic algorithms like HEFT, Min-Min, and SJF offer fast and low-overhead scheduling, suitable for small to medium-sized workflows. However, they often lack the flexibility and adaptability required for complex, large-scale cloud environments with dynamic workloads and multiple objectives.

Bio-inspired metaheuristics, on the other hand, provide a strong foundation for resolving such challenging issues. They are ideally suited for scheduling scientific workflows because of their capacity to conduct global searches and manage multi-objective optimization. However, when used for discrete scheduling problems, these approaches could

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

be more computationally complex and necessitate careful design of encoding systems, fitness functions, and convergence criteria.

METHODS

In the context of **cloud computing**, particularly when using simulation environments such as **CloudSim**, a **scientific workflow** refers to a structured sequence of computational tasks—also called **cloudlets**—that must be executed in a specific order to achieve a desired scientific or analytical result. These workflows are typically **data-intensive** and consist of **multiple interdependent tasks** that operate on large datasets, commonly encountered in scientific disciplines such as bioinformatics, climate modeling, and astrophysics.

To model the dependencies and execution logic of such workflows, a **Directed Acyclic Graph (DAG)** is commonly used. A DAG provides a formal representation of the workflow's structure, where each **node** represents a computational task (cloudlet), and each **directed edge** indicates a **data or execution dependency** between two tasks (see Fig.1).

Let the workflow be represented as a graph G: noting G = (T, E) with:

- $T = \{ T1 \ldots . Tn \}$ is the finite set of independents **tasks (cloudlets)** that constitute the scientific workflow. Each task T_i represents a unit of computation that consumes input data, performs a specified operation, and produces output data. These tasks may vary in computational complexity and resource requirements.

- **E**: is the set of its arcs representing the data constraints between tasks

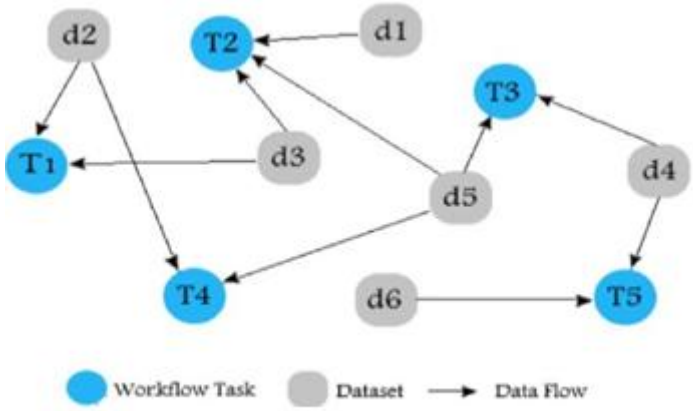

Fig.1. Scientific workflow DAG.

A. SYMBIOTIC ORGANISM SEARCH (SOS) IN NATURE

The **Symbiotic Organism Search** (SOS) algorithm is a bio-inspired metaheuristic optimization method that takes its cues from the biological idea of symbiosis. The association between various biological entities in natural environments is known as symbiosis. This relationship may be neutral to one party, detrimental to the other, or advantageous to both. The SOS algorithm imitates three main symbiotic relationship types: parasitism, commensalism, and mutualism. These exchanges are symbolic of cooperation and the development of solutions.

1. Mutualism

Mutualism describes a symbiotic interaction in which both species involved gain benefits from the association. This cooperative relationship promotes mutual survival, growth, or reproduction and is widely observed in both microscopic and macroscopic ecosystems.

Humans and some strains of the gut-dwelling Escherichia coli (E. coli) are a common example of mutualism in biology (see Figure 2). E. Coli thrives in the host's nutrient-rich intestinal environment, but it also produces critical substances like vitamin K, which is needed to make human blood clotting proteins. This reciprocal interaction and

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

a situation where both species gain from each other's touch are replicated by the SOS algorithm, which concurrently increases the fitness of two potential solutions [13].

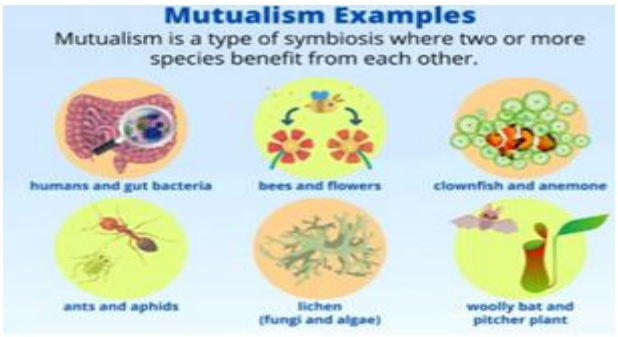

Fig.2. Mutualism example [13].

2. Commensalism

Commensalism is a symbiotic relationship in which **one species benefits**, while the other remains **unaffected—neither harmed nor helped**. This neutral interaction is common in ecosystems where one organism exploits the environment or resources altered by another organism **without interfering with its biological function**.

An ecological example of commensalism can be seen in birds that build nests in trees. The bird benefits from the shelter and support provided by the tree, while the tree is largely unaffected by the presence of the nest (see **Figure.3**.) In SOS, the commensalism phase involves one candidate solution improving its state by interacting with another, while the latter remains unchanged. This phase promotes **exploration of the solution space** by allowing certain solutions to learn from others without reciprocal influence [13].

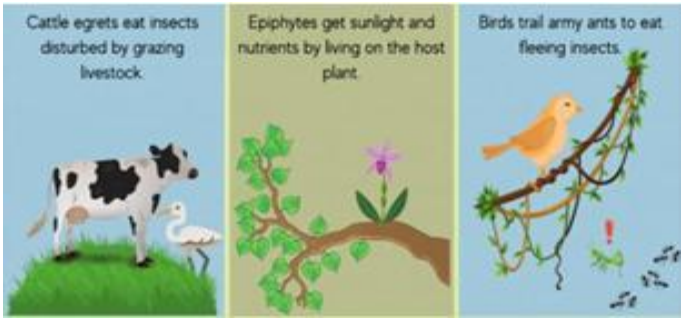

Fig.3. Commensalism example [13].

3. Parasitism

Parasitism represents a **non-mutual symbiotic relationship**, where **one organism (the parasite)** benefits at the **expense of another (the host)**. This interaction typically causes harm or resource loss to the host, while enhancing the survival or reproduction of the parasite.

Common examples include ticks feeding on mammals or parasitic fungi attacking plants. In the SOS algorithm, the parasitism phase mimics this interaction by introducing a parasitic candidate solution that attempts to replace a host solution in the population. If the parasite exhibits better fitness, it replaces the host; otherwise, it is discarded. This strategy allows SOS to maintain diversity and inject competitive pressure into the evolutionary process (see Figure4).

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Fig.4. Parasitism example

In summary, the **Symbiotic Organism Search** algorithm emulates the adaptive mechanisms found in nature, translating ecological interactions into mathematical operators that govern solution evolution, diversification, and intensification. These biologically inspired mechanisms allow SOS to efficiently explore complex, multi-dimensional search spaces in various optimization problems.

B. FORMAL DESCRIPTION OF THE DISCRETE SYMBIOTIC ORGANISM SEARCH (DSOS) ALGORITHM

To address the challenges of solving **large-scale and complex optimization problems**, such as those found in **data intensive scientific workflow scheduling**, we adopt an algorithm inspired by the **natural symbiotic relationships** observed in biological ecosystems—namely, the **Discrete Symbiotic Organism Search (DSOS)** algorithm. This algorithm extends the original SOS framework to **discrete problem domains**, such as task scheduling, where candidate solutions consist of discrete permutations or mappings rather than continuous variables.

Formally, the **DSOS** algorithm operates by maintaining a population of candidate solutions, referred to as **organisms** (see Fig. 5), within an abstract ecosystem. Each organism represents a **potential solution** to the target optimization problem and is evaluated using a **fitness function** that quantifies its quality with respect to the defined objective—such as minimizing the **makespan**, **data transfer time**, and **execution cost** in a workflow.

S.W elements	ID
Cl1	0
Cl2	1
Cl3	2
Cl4	3
Cl5	4
D1	5
D2	6
D3	7
D4	8
D5	9
D6	10

Cloudlets and Data assignments vector:

0	1	2	3	4	5	6	7	8	9	10
Cl1	Cl2	Cl3	Cl4	Cl5	D1	D2	D3	D4	D5	D6
Vm2	Vm1	Vm2	Vm3	Vm3	Vm1	Vm2	Vm3	Vm1	Vm3	Vm1

Fig.5. An example of an organism (our adaptation).

Once the initial ecosystem is constructed—consisting of a randomly generated population of candidate solutions—the search process is immediately initiated. Each candidate solution, referred to as an **organism**, undergoes iterative refinement through simulated symbiotic interactions aimed at improving its fitness within the ecosystem.

The evolution of organisms follows **biologically inspired strategies**, mimicking natural symbiotic relationships observed in ecological systems. These strategies are divided into **three distinct interaction phases**:

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

1. **Mutualism Phase** – where two organisms collaborate in a way that benefits both, allowing for shared improvement in their solution structures;

2. **Commensalism Phase** – where one organism benefits from interaction with another, while the latter remains unaffected;

3. **Parasitism Phase** – where one organism (the parasite) attempts to replace another (the host) by introducing a mutated version that competes for survival.

Each organism in the population participates in these phases, during which a **new or modified version** of the organism is generated. The resulting offspring is evaluated using the defined **fitness function**, and it **replaces the original** only if it demonstrates a measurable improvement in quality (i.e., a lower objective function value for minimization problems).

This adaptive mechanism ensures that the population continually evolves toward more optimal solutions, while avoiding unnecessary or regressive updates. The process of interaction, evaluation, and selective replacement is repeated over **multiple generations**.

The algorithm proceeds in this manner **iteratively**, applying the symbiotic phases to each organism in succession during every generation. The optimization loop continues until a **termination condition** is satisfied, such as:

- Reaching a predefined number of iterations,
- Achieving convergence (no significant improvement in fitness over time),
- Meeting a computational budget or runtime threshold.

By adopting this strategy, the **DSOS** algorithm efficiently explores the search space, balances diversification and intensification, and progressively improves the solution quality toward the global optimum.

- **Mutualism phase:** in this phase, the organism xj is randomly selected from the ecosystem to mutually interact with the organism xi with the sole aim of increasing their mutual survival advantage in the ecosystem. The resulting new solutions $\boldsymbol{xi'}$ and $\boldsymbol{xj'}$ which is as a consequence of this interaction is calculated based on equations (3) and (4).

$$s_1(p) \leftarrow x_i + r_1\left(x^{\text{best}} - \frac{x_i + x_j}{2}\right) \quad (1)$$

$$s_2(p) \leftarrow x_i + r_2\left(x^{\text{best}} - \frac{x_i + x_j}{2}\right) \quad (2)$$

$$x_i'(q) \leftarrow |s_1(p)| mod\ m \quad (3)$$

$$x_j'(q) \leftarrow |s_2(p)| b\ mod\ m \quad (4)$$

$$\forall \rho \in \{1,2,3,\dots,n\} \quad \forall q \in \{1,2,3,\dots,m\}$$

- **Commensalism phase:** Similar to the mutualism phase, an organism xj is randomly selected from the ecosystems population and made to interact with the organism $\boldsymbol{xi}$. The relationship interaction is such that only one organism benefits from the interaction. For example, the organism drives benefit from its interaction with $\boldsymbol{xj}$ **.**

$$s_3 \leftarrow r_3\left(x^{\text{best}} - x_j\right) \quad (5)$$

$$x_i'(q) \leftarrow |s_3(p)| b\ mod\ m \quad (6)$$

$$\forall \rho \in \{1,2,3,\dots,n\} \quad \forall q \in \{1,2,3,\dots,m\}$$

Where r_3 is a uniformly generated random number between 0 and 1.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

We use **eq.5** and **eq.6** to obtain the modified position of the organism ***xi*** in the commensalism phase.

- **Parasitism phase**: in this phase, an artificial parasite vector denoted by x_{pv} created in the problem search space by mutating the organism *xi* then modifying its randomly selected dimensions using a random number. The organism ***xj*** with **i/= j** is selected randomly from the ecosystems population to serve as a host to the x_{pv}. The evaluation is carried out such that, if the fitness value of the x_{pv} is better than that of the organism ***xj***, then x_{pv} will replace the position of ***xj*** in the population, otherwise, if the fitness value of ***xj*** is better, then ***xj*** will build an immunity against x_{pv} after which x_{pv} is removed from the population.

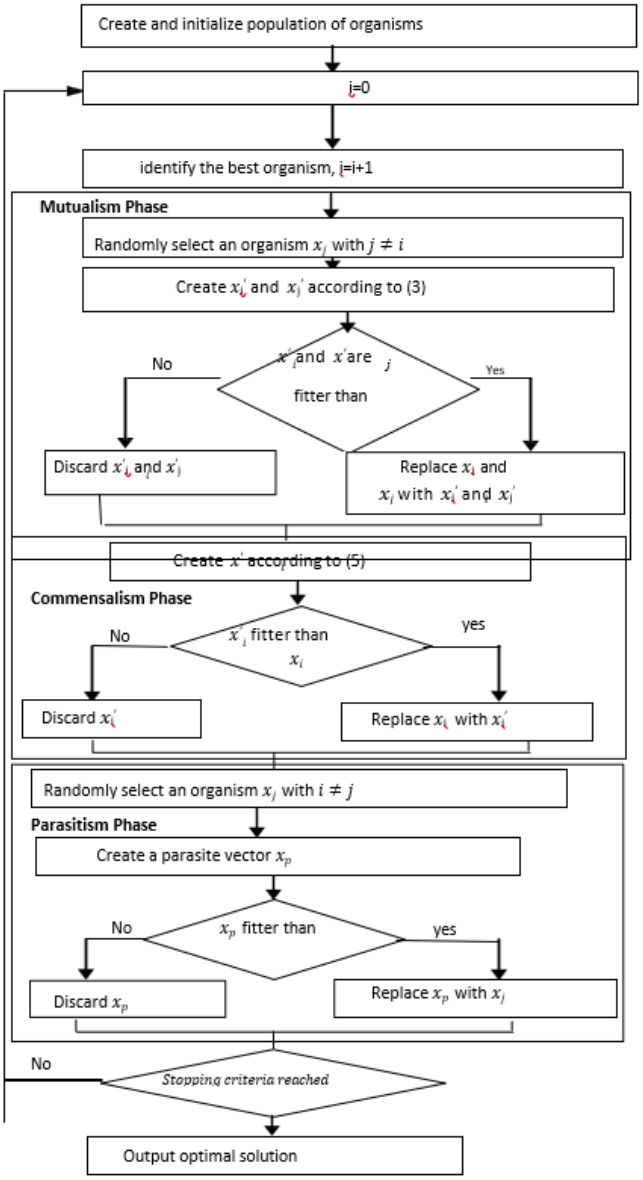

Fig. 6. Flowchart of the symbiotic organism search.

FITNESS FUNCTION

A fitness function is used to evaluate the quality of each solution in the population, and to pick the optimal one, solutions can be compared to choose which is fitter, in this case it is the minimum of the makespan time and the execution cost, that is calculated by scheduling the cloudlets to the given VMs, each cloudlet has a processing time, in addition to the Data transfer time, in case the data are not located in the same virtual machine as the cloudlet.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

$$Fitness = Min\ \beta_1 \sum_{i=1}^{n}(PT_i + DT_i) + \beta_2 \sum_{i=1}^{n} ExecutionCost_i \qquad \textbf{(7)}$$

Where

$$Cloudlet\ processing\ time\ PT_i: PT_i = \frac{Cloudlet\ lenght_i}{Processing\ speed_k}$$

$$Data\ Transfer\ Time\ DT_i: DT_i = \frac{\sum_{j=1}^{l} Data\ Transfer\ Time_j}{Bandwidth}$$

$$\beta_1 + \beta_2 = 1$$

As indicated in the equation 7 the total value of the objective's coefficients must be one and each coefficient control the impact of each objective in the fitness value.

RESULTS

To validate the proposed approach, we have implemented our algorithm in CloudSim [3]. The CloudSim simulation layer provides support for modeling and simulation of virtualized Cloud-based data center environments including dedicated management interfaces for virtual machines (VMs), memory, storage, and bandwidth. The fundamental issues such as provisioning of hosts to VMs, managing application execution, and monitoring dynamic system state are handled by this layer.

A. Impact of cloudlet number on the respond time

In this simulation, we have designed three Data Centers comprising of 2 diverse Hosts. Each Host is equipped with a single processor with varying speeds in MIPS ranging from 20,000 to 200,000. The bandwidth of each Host ranges from 10,000 to 200,000. The size of the generated data is randomly generated between 900 MB and 1,100 MB. The purpose of this simulation is to investigate the effect of varying the number of Cloudlets on the response time. We fixed 30 virtual machines, 10 data, 200 organisms, $\beta_1 = 0.7$, $\beta_2 = 0.3$, and 1000 iteration.

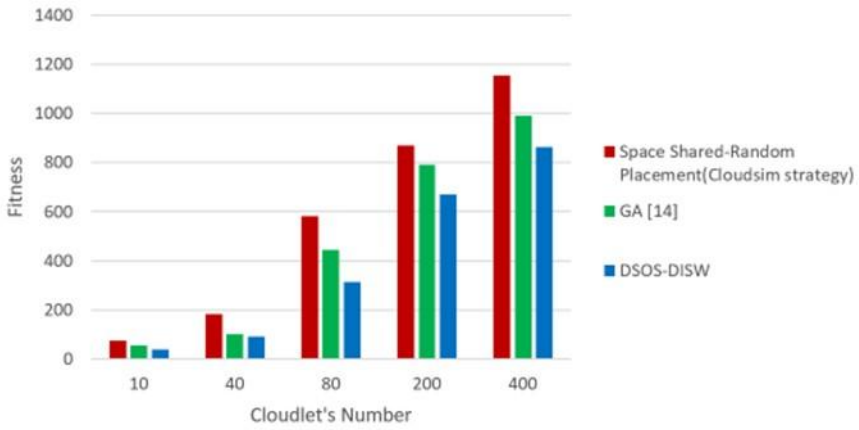

Fig.7. Fitness Vs Cloudlet's Number.

To show the **effectiveness** of **DSOS-DISW** compared to **GA [14]** and **Space Shared-Random Placement**, we can analyze the relative percentage improvement in fitness for each number of cloudlets.

Table.1 Improvement of **DSOS-DISW**

Cloudlets	DSOS-DISW vs Space Shared (%)	DSOS-DISW vs GA (%)
10	**44.4%**	**28.6%**
40	**44.4%**	**23.1%**

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

80	**45.0%**	**21.4%**
200	**23.3%**	**11.5%**
400	**26.3%**	**11.2%**

DSOS-DISW proves to be significantly more effective than both comparison strategies. Its ability to maintain lower fitness across all workload.

B. Impact of the iteration number on the response time

We conducted an experiment fixing 100 cloudlets, 30 virtual machines, 40 data and 200 organisms to observe the impact of the iteration number on the response time, the results are shown in the figure 8.

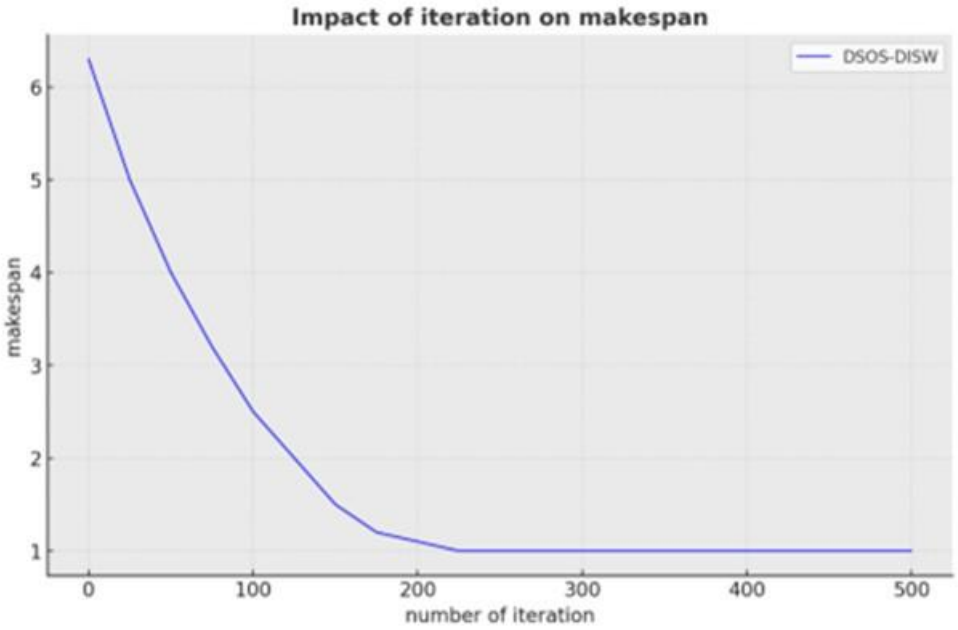

Fig.8. Impact of iteration on makespan.

We can say that the makespan decreases rapidly in the first 100–150 iterations, which indicates that the **DSOS-DISW** algorithm is effective in quickly improving the schedule or system efficiency during early search. After about 225 iterations, the curve flattens, stabilizing at around 1.0. This suggests convergence or reaching a near-optimal solution.

C. Impact of Execution Cost

Figure 9 presents the execution cost of the workflow with 100 cloudlets over 200 iterations. The results of the proposed **DSOS-DISW** based workflow scheduling approach are compared with those of the GA based workflow scheduling [14]. As shown in this plot, the **DSOS-DISW** algorithm achieves lower execution costs compared to the GA method across both configurations. Notably, the use of 80 VMs results in a significant reduction in overall cost compared to the 30 VMs scenario, due to improved parallelism and resource availability. However, it is also evident that increasing the number of VMs leads to a higher baseline cost, as more computational resources are involved. Nevertheless, the **DSOS-DISW** based approach consistently demonstrates superior cost efficiency in both cases, confirming its effectiveness in workflow scheduling.

Journal of Information Systems Engineering and Management
2025, 10(57s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

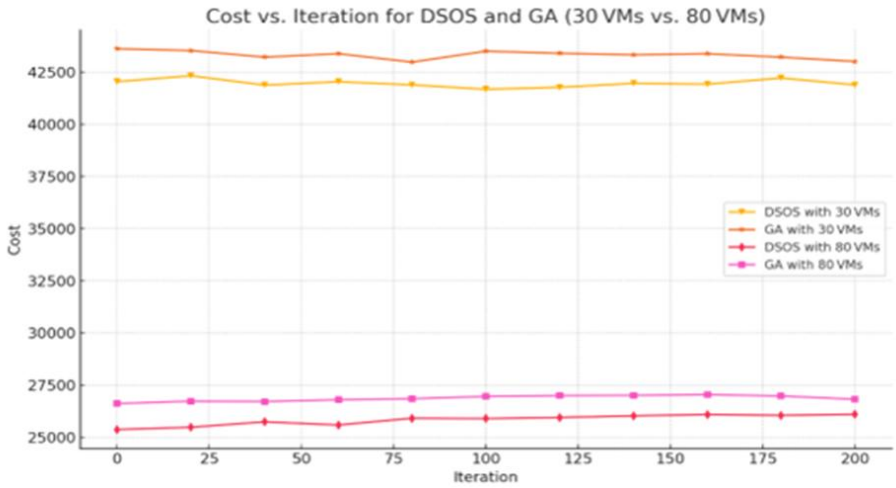

Fig.9. Impact of execution Cost.

D. Time and cost Vs Iteration

The graphs for Cloudlets execution/transferring data time, and cloudlets execution cost using our **DSOS** based algorithm are illustrated in Fig.10.

As can be seen, **execution time** fluctuates significantly from **3700 to 1500 seconds**, and **execution cost** varies between **250 and 420 USD per hour**, over the span of **2000 iterations**. A notable trade-off is observed in the interval **[100, 200]**, where the **cost increases sharply** while the **execution time decreases**. This reflects the conflicting objectives within the task scheduling strategy: improving one (e.g., execution speed) may worsen the other (e.g., cost). Further fluctuations continue in later iterations, showing unstable optimization behavior. In the obtained solution at **iteration 1400**, the execution time and cost settle at **2200 seconds** and **275 USD/hour** respectively, suggesting a suboptimal compromise between the two metrics at the end of the run.

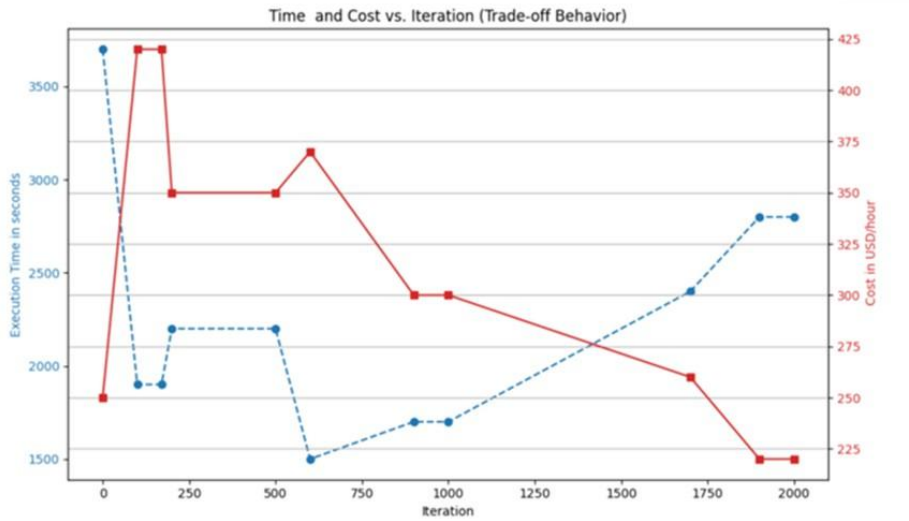

Fig.10. Time and Cost Vs. Iteration.

CONCLUSION

Cloud computing has significantly transformed the IT landscape by offering scalable, flexible, and cost-effective solutions for data storage and processing. Its adaptability has revolutionized the way individuals and organizations manage and access computational resources. A key component of cloud computing is task scheduling, which plays a crucial role in ensuring the efficient utilization of these resources. By intelligently mapping tasks to available virtual

machines, scheduling algorithms enhance system performance, reduce response times, and minimize data transfer overhead. Task scheduling is particularly vital in the execution of scientific workflows, where it ensures the efficient management of task dependencies and resource allocation.

In this study, we propose the use of the Discrete Symbiotic Organisms Search (**DSOS**) algorithm to optimize cloudlet scheduling, aiming to minimize execution time and execution cost to minimize expenses while optimizing performance by selecting the most optimal solution through its three-phase evolutionary process. To evaluate the effectiveness of this approach, we developed a CloudSim-based simulation environment integrating the **DSOS** algorithm. The experimental results demonstrate the superior performance of our proposed method, validating its effectiveness in improving scheduling efficiency and achieving significant reductions in execution time.

The incorporation of bio-inspired tactics into scheduling systems creates new opportunities for cloud management intelligence. Extending DSOS to facilitate energy-conscious scheduling, and deployment in hybrid or edge-cloud systems may be the main focus of future research.

REFRENCES

[1] Stanoevska-Slabeva, K., & Wozniak, T. (2009). Cloud basics–an introduction to cloud computing. In *Grid and cloud computing: a business perspective on technology and applications* (pp. 47-61). Berlin, Heidelberg: Springer Berlin Heidelberg.

[2] Siham, K., Chahinez, C., & Hassane, M. M. (2023, October). DSOS based Scientific Workflow Scheduling Optimization in Cloud Computing. In *2023 International Conference on Networking, Sensing and Control (ICNSC)* (Vol. 1, pp. 1-5). IEEE.

[3] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), 23-50.

[4] Wang, Y., Guo, Y., Guo, Z., Liu, W., & Yang, C. (2020). Protecting scientific workflows in clouds with an intrusion tolerant system. *IET Information Security*, *14*(2), 157-165.

[5] Durillo, J. J., Nae, V., & Prodan, R. (2014). Multi-objective energy-efficient workflow scheduling using list-based heuristics. *Future Generation Computer Systems*, *36*, 221-236.

[6] Hamayun, M., & Khurshid, H. (2015). An optimized shortest job first scheduling algorithm for CPU scheduling. *J. Appl. Environ. Biol. Sci*, *5*(12), 42-46.

[7] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, *13*(3), 260-274.

[8] Lu, H., Niu, R., Liu, J., & Zhu, Z. (2013). A chaotic non-dominated sorting genetic algorithm for the multi-objective automatic test task scheduling problem. *Applied Soft Computing*, *13*(5), 2790-2802.

[9] Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010, April). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications* (pp. 400-407). IEEE.

[10] Lin, M., Xi, J., Bai, W., & Wu, J. (2019). Ant colony algorithm for multi-objective optimization of container-based microservice scheduling in cloud. *IEEE access*, *7*, 83088-83100.

[11] Yildiz, A. R. (2013). Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. *Information Sciences*, *220*, 399-407.

[12] Navimipour, N. J., & Milani, F. S. (2015). Task scheduling in the cloud computing based on the cuckoo search algorithm. *International Journal of Modeling and Optimization*, *5*(1), 44.

[13] Abdullahi, M., Ngadi, M. A., Dishing, S. I., Abdulhamid, S. I. M., & Usman, M. J. (2020). A survey of symbiotic organisms search algorithms and applications. *Neural computing and applications*, *32*(2), 547-566.

[14] Kouidri, S., & Kouidri, C. (2022). Bi-Objective Optimizing for Data-Intensive Scientific Workflow Scheduling in Cloud Computing. *International Journal of Organizational and Collective Intelligence (IJOCI)*, *12*(1), 1-12.