

Article

# An Enhanced Discrete Symbiotic Organism Search Algorithm for Optimal Task Scheduling in the Cloud

Suleiman Sa'ad <sup>1,2,\*</sup>, Abdullah Muhammed <sup>1,†</sup>, Mohammed Abdullahi <sup>3,†</sup>, Azizol Abdullah <sup>1,†</sup> and Fahrul Hakim Ayob <sup>1,†</sup>

<sup>1</sup> Department of Communication and Networking, Universiti Putra Malaysia, Seri Kembangan 43400, Malaysia; abdullah@upm.edu.my (A.M.); azizol@upm.edu.my (A.A.); fahrulhakim@gmail.com (F.H.A.)

<sup>2</sup> Department of Information Technology, Modibbo Adama University of Technology Yola, Yola 640231, Nigeria

<sup>3</sup> Department of Computer Science, Ahmadu Bello University Zaria, Zaria 810107, Nigeria; abdullahilwafu@abu.edu.ng

\* Correspondence: suleimanu@mautech.edu.ng; Tel.: +234-803-351-6997

† These authors contributed equally to this work.

**Abstract:** Recently, cloud computing has begun to experience tremendous growth because government agencies and private organisations are migrating to the cloud environment. Hence, having a task scheduling strategy that is efficient is paramount for effectively improving the prospects of cloud computing. Typically, a certain number of tasks are scheduled to use diverse resources (virtual machines) to minimise the makespan and achieve the optimum utilisation of the system by reducing the response time within the cloud environment. The task scheduling problem is NP-complete; as such, obtaining a precise solution is difficult, particularly for large-scale tasks. Therefore, in this paper, we propose a metaheuristic enhanced discrete symbiotic organism search (eDSOS) algorithm for optimal task scheduling in the cloud computing setting. Our proposed algorithm is an extension of the standard symbiotic organism search (SOS), a nature-inspired algorithm that has been implemented to solve various numerical optimisation problems. This algorithm imitates the symbiotic associations (mutualism, commensalism, and parasitism stages) displayed by organisms in an ecosystem. Despite the improvements made with the discrete symbiotic organism search (DSOS) algorithm, it still becomes trapped in local optima due to the large size of the values of the makespan and response time. The local search space of the DSOS is diversified by substituting the best value with any candidate in the population at the mutualism phase of the DSOS algorithm, which makes it worthy for use in task scheduling problems in the cloud. Thus, the eDSOS strategy converges faster when the search space is larger or more prominent due to diversification. The CloudSim simulator was used to conduct the experiment, and the simulation results show that the proposed eDSOS was able to produce a solution with a good quality when compared with that of the DSOS. Lastly, we analysed the proposed strategy by using a two-sample *t*-test, which revealed that the performance of eDSOS was of significance compared to the benchmark strategy (DSOS), particularly for large search spaces. The percentage improvements were 26.23% for the makespan and 63.34% for the response time.

**Keywords:** cloud computing; scheduling; metaheuristic; eDSOS; optimisation



**Citation:** Sa'ad, S.; Muhammed, A.; Abdullahi, M.; Abdullah, A.; Hakim Ayob, F. An Enhanced Discrete Symbiotic Organism Search Algorithm for Optimal Task Scheduling in the Cloud. *Algorithms* **2021**, *14*, 200. <https://doi.org/10.3390/a14070200>

Academic Editors: Gopal Pandurangan and Michele Scquizzato

Received: 5 May 2021

Accepted: 18 May 2021

Published: 30 June 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Cloud computing uses many resources, including storage, processors, networks, memory, and applications that have been provisioned as services, as well as continually changing information technology (IT) resources [1]. In this regard, the paradigm of cloud computing has drastically brought down the cost of acquiring both software and hardware, as well as that of the deployment of applications and the cost of maintenance. Due to the high scalability in the cloud, customers are not worried about inaccurate estimation of the

service scale, which can lead to wastage of resources when over-provisioned or a loss of revenue when under-provisioned [2,3].

Cloud computing shares resources between users of the cloud via the concept known as virtualisation. Virtualisation allows several environments that are running remotely to be safely integrated into physical machines for the optimal usage of the physical resources and energy [2]. Cloud data are situated crosswise over servers that are interconnected through resources that are networked and accessed through virtual machines (VMs). In this way, VMs in cloud server farms are a critical element of software stacks.

An example of a cloud platform that renders infrastructure services in the form of VMs is the Amazon Elastic Computing Cloud (EC2) [4]. One of the central aims of cloud computing is to maximise the revenue for both sides of the cloud provider, as well as for the consumer. Task scheduling is an NP-complete problem, and it has developed as one of the focuses in cloud computing [5], as inefficient task scheduling results in a loss of revenue, degradation of execution, and violation of service-level agreements (SLAs).

Hence, strategies that are efficient for task scheduling are necessary in order to minimise metrics that are computationally based, such as the makespan, response time, system throughput, and system utilisation, as well as those that are network-based, such as the traffic volume, network communication cost, data communication cost, and round-trip time [6]. These measurement metrics are cardinal in observing the cloud procedures for handling difficulties—for example, quality of service (QoS) and SLA guarantees, load balancing, energy efficiency, and fault tolerance [6].

There is existing research on task scheduling with encouraging solutions in the cloud computing domain, yet problems related to task scheduling still remain NP-complete [7–10]. In cloud computing, the vast majority of task scheduling strategies applied are rule-based [11–13] because they are easy to implement. According to [5], with regard to the complex problems of task scheduling, rule-based strategies are characterised by poor performance. Recently, the commonly used metaheuristic strategies for task scheduling problems in grids, as well as in cloud computing environments, have included the symbiotic organism search (SOS) [7,9,14], genetic algorithms (GAs) [15–19], particle swarm optimisation (PSO) [20–30], ant colony optimisation (ACO) [31–33], hybrid firefly–genetic [34], the whale optimisation algorithm (WOA) [35], and hybrid electro search with a genetic algorithm (HESGA) [36].

SOS is faster in terms of convergence compared with PSO, GAs, and ACO due to its exploratory and exploitative features for finding optimal solutions [9,14,24,37]. Because of the better execution of SOS compared to that of PSO, ACO, and GAs, variants of SOS were utilised to benchmark our proposed task scheduling strategy. It is well known in the literature that metaheuristics are superior to heuristics or local search scheduling strategies, which is our reason for using the metaheuristic technique.

In this paper, an enhanced discrete symbiotic organism search (eDSOS) strategy is proposed and used to schedule set-of-tasks applications in a cloud computing setting; therefore, the costs of network communication and of data transfer are not key optimisation setbacks when addressing independent tasks. In the setup of the experiment, similarly to the experiments of [9], four dataset types were used—normal, left-half, right-half, and uniform distributions—in order to appropriately test the proposed strategy. Accordingly, different kinds of distributions were adopted to gain insight into the performance trends of the proposed strategy. Most applications, such as web services, run for a long duration, which results in variations in their CPU demands.

High-performance computing (HPC) applications have a short life expectancy; however, they place an extreme pressure on the CPU. Thus, the statistically chosen models for task sizes speak to the divergent settings of simultaneously scheduling web applications and HPC. The normal distribution indicates a circumstance where web applications and HPC are balanced. Left-half circulation delineates a situation wherein web applications are less likely than HPC applications to be scheduled, and right-half appropriation indicates

the opposite. A uniform distribution indicates a situation where a single application type is scheduled [9].

The proposed strategy can also be utilised in order to find solutions that are near-optimal to well-outlined problems along discrete space, such as scheduling, production planning, design problems, optimisation of network synthesis, and inventory control. These problems come up in in many domains, including engineering, management, telecommunications, etc.

The major contributions of this paper are as follows:

- i. Proposal of a strategy for enhancing the diversification of the SOS algorithm for optimal task schedules.
- ii. Assessment of the proposed strategy by measuring the performance metrics of the makespan and response time between VMs.
- iii. Statistical validation of the results obtained with eDSOS against those of DSOS by utilising paired *t*-tests and a one-tailed distribution.

The rest of this paper is organised as follows: the related work, which includes metaheuristic algorithms used for problems of scheduling tasks in the cloud, the standard SOS, and the processes of discrete symbiotic organism search, is presented in Section 1. In Section 2 the description and formulation of the problem are presented. In Section 3, the proposed system's design and its depiction are exhibited. The simulation settings are described in Section 4. Section 5 presents the results and discussion, and Section 6 presents the conclusion.

## 2. Related Work

This section focuses on the existing literature related to metaheuristic strategies that is applied for task scheduling in a cloud computing environment; these include the symbiotic organism search (SOS), cuckoo search (CS), cat search optimisation (CSO), ant colony optimisation (ACO), particle swarm optimisation (PSO), and genetic algorithms (GAs).

### 2.1. Metaheuristic Scheduling Strategies in the Cloud

Several works [7–9,14,15,17,22,29,31,32,38–45] on metaheuristic scheduling strategies have been carried out to solve problems of task assignment with regard to minimising the makespan, degree of imbalance, convergence speed, and response time. Furthermore, the strategies have been shown to be good for finding an optimal task–resource assignment scheme that tries to minimise the computational cost, maintains a good quality of service, and increases the utilisation of computing resources. SOS, CS, CSO, PSO, ACO, GA, and their different versions are among the commonly utilised nature-inspired strategies and are based on populations found in the cloud.

In most situations, SOS outperforms PSO, GAs, and ACO [9,24,37] and has a faster execution time. SOS is simple to implement as a parameter-free strategy in comparison with other population-based strategies. Furthermore, problems to do with workflow scheduling have been researched to a great extent by using PSO [29,30,46,47], with the sole aim of minimising the cost of communication and the makespan. Likewise, independent task scheduling has additionally been carried out in the cloud by utilising SOS [7,9,14,41] and PSO [43,44,48], and both were shown to reduce the makespan. Enhanced and hybrid variants of PSO [5,26,29,47,48] were also proposed for task scheduling in the cloud environment and were found to be a solution that was superior to GAs and ACO.

Consequently, the SOS algorithm has recently gained popularity in terms of its robustness and the efficiency of the metaheuristic algorithm. As such, the SOS algorithm has attracted researchers' attention for its application in solving optimisation problems that arise from numerous disciplines, such as science and engineering. Moreover, Table 1 shows a summary of the applications of the SOS algorithm. Many attempts have been successful in adapting and modifying SOS algorithms to address multi-objective and constrained optimisation, which are important aspects in facilitating the design and optimisation of several problems in engineering, as well as in computer science.

While considerable success has recently been found in these domains, the optimisation of problems in these domains remains an active research issue. As such, SOS algorithms have been applied in order to solve optimisation problems in different areas, such as machine learning, construction project scheduling, power optimisation, design optimisation of engineering structures, energy optimisation, transportation, wireless communication, and economic dispatching.

Based on this trend of the use of the SOS for optimisation problems, it has been proven to provide all-purpose principles that can be easily adapted in order to solve an extensive range of optimisation problems in many areas, which prompted the authors of this study to utilise SOS to solve a large-scale task scheduling problem in a cloud computing setting. Consequently, this paper presents SOS-based variant strategies for scheduling large-scale tasks in an IaaS cloud computing setting.

**Table 1.** Applications of SOS algorithms in different fields.

| Area                            | Application   | Reference |
|---------------------------------|---|-----------|
| Machine learning                | This method maps the SOS algorithm from a continuous space to a discrete space using an adaptive S-shaped transfer function, and it can be used to search for the optimal feature subset in a feature selection space.                                    | [49]      |
|                                 | Input parameter optimisation in data classification.  | [50]      |
| Construction project scheduling | This work developed a new ensemble model called the evolutionary neural machine inference model (ENMIM) in order to estimate energy consumption in residential buildings based on actual data.  | [51]      |
|                                 | An optimisation tool is necessary for a construction management system in order to develop the desired construction schedule and to save time and costs.  | [52]      |
|                                 | Time, cost, and labour utilisation trade-off.   | [53]      |
|                                 | Adjustment of the resource profile through a resource-levelling process.  | [54]      |
|                                 | Optimising levelling of multiple resources  | [55]      |
|                                 | Non-linear, non-convex, and implicit with respect to the design variables.  | [56]      |
| Power optimisation              | Power optimisation of three-dimensional turbo code using solutions for more complex, nonlinear, multi-modal optimisation problems, such as ORPD.  | [57]      |
|                                 | Optimal power flow based on the valve-point effect and prohibited zones.  | [58]      |
|                                 | Optimal power flow of power system with FACTS.  | [59]      |
|                                 | Minimisation of network power loss while satisfying the power demand.   | [60]      |
|                                 | Load frequency control.   | [61]      |
|                                 | Short-term hydrothermal scheduling.   | [62]      |
|                                 | Optimal coordination of directional over-current relays.  | [63]      |
|                                 | Congestion management in a deregulated environment.   | [64]      |
|                                 | Real power loss minimisation.   | [65]      |
| Engineering structures          | Design of a reconfigurable concentric circular array with phase-only controls differentiating the beams.  | [66]      |
|                                 | The primary principles, the basic concepts, and mathematical relations of the SOS algorithm are presented, followed by the engineering applications of the SOS algorithm.   | [67]      |
|                                 | Structural design optimisation.   | [68]      |
|                                 | Optimum design of frame and grillage systems.   | [69]      |
| Energy optimisation             | Minimising the energy of point charges on a sphere.   | [70]      |
| Transportation                  | Capacitated vehicle routing.  | [71]      |
|                                 | The excellence coefficients let ECSDSOS choose shorter edges (routes) for generating better local paths.  | [72,73]   |
| Wireless communications         | Synthesis of antenna arrays.  | [74]      |
|                                 | Design of linear antenna arrays with a low side-lobe level.   | [75]      |
|                                 | Solved economic/emission dispatch.  | [76]      |
|                                 | Large-scale economic dispatch with valve-point effects.   | [77]      |
| Economic dispatch               | Economic load dispatch.   | [78]      |
|                                 | Dynamic economic dispatch with valve-point effects.   | [79]      |
|                                 | Bid-based economic load dispatch  | [80]      |
|                                 | Security-constrained economic dispatch.   | [81]      |
| Material optimisation           | The material distribution is described by control points; coordinates of these points are located along the thickness of a plate using B-spline basis functions. In addition, DNN is used as an analysis tool to supersede finite element analysis (FEA). | [82]      |

## 2.2. Standard Symbiotic Organism Search algorithm

The symbiotic organism search (SOS) algorithm is a new population-based metaheuristic algorithm that was proposed by [84]. The algorithm solves numerical optimisation problems for a continuous real space. It imitates the symbiotic associations that exist amongst organisms (i.e., the mutualism stage, commensalism stage, and parasitism stage) in an ecosystem. The mutualism stage essentially implies the concurrent co-existence of different organisms whereby both benefit from the association.

The commensalism stage is an interaction of two unique organisms in which one gains from the association and the other is not hurt. A situation in which one organism gains and the other is harmed by the relation is known as the parasitism stage. Here, each organism, as a member of the ecosystem, is made up of a vector in the solution plane. Then, each individual organism in the search space is allotted a value that hints at the degree of its adaptation to the desired objective. The algorithm iteratively utilises a population of the conceivable solutions so as to optimally converge to a position where the best global solution lies. Furthermore, it applies the three stages of symbiotic associations to update the positions of the solution vector in the search space.

According to [85], SOS is an optimisation algorithm that performs an iterative process, as described by Definition 1. The process maintains the population of organisms that demonstrate candidate solutions for the problem being addressed. The germane information regarding the decision factors and a fitness value are capsulised into the organism, and together comprise a list for its execution. Basically, the directions of the organisms are amended by applying the three stages of symbiotic relationships. The basic pseudocode of standard SOS is depicted in Algorithm 1.

**Definition 1.** We begin with the function  $f : D \rightarrow \mathbb{R}$  find  $X' \in D : \forall X \in D f(X') \leq$  or  $\geq f(X)$ , where  $\leq$  ( $\geq$ ) is a minimisation (maximisation), the objective function to be optimised is  $f$ , and  $D$  symbolises the search space, while the feasible solutions are components of  $D$ . A vector of optimisation variables is represented by  $X$ ;  $X = \{x_1, x_2, x_3, \dots, x_n\}$ . An enhanced solution is an achievable solution  $X'$  that enhances  $f$ .

---

### Algorithm 1 Standard Symbiotic Organism Search Algorithm

---

```

Create and Initialise the Population of Organisms in the Ecosystem  $X = \{X_1, X_2, X_3 \dots X_N\}$ 
Set up stopping criteria
iteration_number  $\leftarrow 0$ 
 $X^{best} \leftarrow 0$ 
Do
  iteration_number  $\leftarrow$  iteration_number + 1
   $i \leftarrow 0$ 
  Do
     $i \leftarrow i + 1$   $j = 1$  to  $N$   $f(X_j) > f(X^{best})$ 
     $X^{best} \leftarrow X_j$ 
  Mutualism phase
  Commensalism phase
  Parasitism phase
While  $i \leq N$ 
While stopping condition is not true

```

---

## 2.3. Processes of the Discrete Symbiotic Organism Search

The standard SOS was intended and used for solving problems that are continuous. Problems that are discrete in nature, such as the scheduling of tasks in a cloud computing environment, are solved by utilising the discrete symbiotic organism search (DSOS), which was proposed by [9]; here, the scheduling of the task optimisation problem was formulated as a discrete problem. The continuous variant of the standard SOS was utilised to address

optimisation problems for which the optimisation variables  $X_i, i = 1, 2, 3, \dots, n$  are continuous,  $X_i \in \mathbb{R} \ i = 1, 2, 3, \dots, n$ , but not optimisation problems for which the the variables  $X_i, i = 1, 2, 3, \dots, n$  are a set of elements that are countable.

Moreover, the standard SOS works on optimisation variables that are completely continuous, while DSOS is used for optimisation variables that are discrete. In DSOS, the transverse and location of the organisms in the continuous space are mapped out into formulated discrete functions, as displayed in Equations (3)–(8).

As proposed by [9], the DSOS algorithm comprises the following three phases.

**Initialisation phase:** In this phase, the initial population of the organisms is generated. Each organism includes  $D$  components denoting candidate solutions and a fitness function to decide the level of optimality of the solutions. Thus, each organism corresponds to a decision for encoded task scheduling in a vector with dimensions of  $1 \times n$ , as depicted in Table 2, where the number of tasks is  $n$ . The range of natural numbers  $[1, m]$  represents the vector’s elements, where the number of virtual machines (VMs) is  $m$ . Moreover, the position of the  $k$ th organism in the solution space is  $X_k$ , and the VM for which task  $j$  is allotted by the scheduler in the organism is  $X_k(j)$ .

**Table 2.** An illustration of task scheduling.

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|-------|-------|-------|-------|
| 1     | 1     | 3     | 2     |

**Iterative phase:** In this phase, the organisms’ positions are updated by imitating the three stages of symbiotic relationships (mutualism, commensalism, and parasitism). In the mutualism and commensalism stages,  $X^{best}$  updates some portions of the variables, which proceed to function as the memory of the procedure;  $X^{best}$  is the best point visited up until this point by an organism and its neighbors. Consequently, Equations (1) through (4) are utilised to adjust the positions of the organisms chosen in the mutualism stage.

$$S_1(p) = X_i + r_1(X^{best} - ((X_i + X_j)/2)f_1) \tag{1}$$

$$S_2 = X_j + r_2(X^{best} - ((X_i + X_j)/2)f_2) \tag{2}$$

$$X_i^*(q) = \lceil s_1(p) \rceil \text{mod } m + 1 \tag{3}$$

$$X_j^*(q) = \lceil s_2(p) \rceil \text{mod } m + 1 \tag{4}$$

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

where  $X_i^*$  and  $X_j^*$  are the changed positions of the  $i$ th and  $j$ th members of the ecosystem;  $i \neq j$ , and  $X_j$  is the organism that was selected randomly in the  $i$ th iteration for  $i \neq j$ .  $f_1, f_2 \in \{1, 2\}$  are decided randomly, and they make up the benefit factor from the mutual association;  $r_1, r_2 \in \text{rand}(0, 1)$  are random numbers that are uniformly generated in the specified interval, and  $\lceil \cdot \rceil$  indicates a ceiling function. In the commensalism stage, an organism’s changed position  $X_i^*$  is found by applying Equations (5) and (6).

$$S_3(p) = r_3(X^{best} - X_j) \tag{5}$$

$$X_i^*(q) = \lceil s_3(p) \rceil \text{mod } m + 1 \tag{6}$$

$\forall p \in \{1, 2, 3, \dots, n\} \forall q \in \{1, 2, 3, \dots, m\}$ , and  $j \neq i$ , where  $r_3$  is also a random number that is uniformly generated between  $-1$  and  $1$ .

Furthermore, in the parasitism stage, a vector  $X^p$ , which is known as a parasite, is produced by applying Equations (7) and (8).

$$S_4(p) = r_4 X_i \tag{7}$$

$$X^p(q) = \lceil s_4(p) \rceil \text{mod } m + 1 \tag{8}$$

$\forall p \in \{1, 2, 3, \dots, n\} \forall q \in \{1, 2, 3, \dots, m\}$  and  $j \neq i$ , where  $r_4$  is also a random number that is uniformly generated between 0 and 1.

**Termination phase:** In this phase, all of the phases are repeated until a stopping criterion is reached.

The pseudocode for the discrete symbiotic organism search is presented in Algorithm 2.

---

#### Algorithm 2 Discrete Symbiotic Organism Search Algorithm

---

```

1: Create and initialise the population of organisms in the ecosystem  $X = \{X_1, X_2, X_3 \dots X_N\}$ 
2: Set up stopping criteria
3:  $iteration\_number \leftarrow 0$ 
4:  $X^{best} \leftarrow 0$ 
5: Do
6:  $iteration\_number \leftarrow iteration\_number + 1$ 
7:  $i \leftarrow 0$ 
8: Do
9:  $i \leftarrow i + 1$   $j = 1$  to  $N$   $f(X_j) > f(X^{best})$  //  $f(X)$  is the fitness function
10:  $X^{best} \leftarrow X_j$ 
11: Mutualism phase
12: Randomly select  $X_j$  with  $i \neq j$ 
13: Update  $X_i^*$  and  $X_j^*$  using Equations (3) and (4)  $f(X_i^*) > f(X_i)$ 
14:  $X_i \leftarrow X_i^*$   $f(X_i^*) > f(X_i)$ 
15:  $X_j \leftarrow X_j^*$ 
16: Commensalism phase
17: Randomly select  $X_j$  with  $i \neq j$ 
18:  $r_3 \leftarrow rand(-1, 1)$ 
19: Update  $X_i^*$  according to Equation (6)  $f(X_i^*) > f(X_i)$ 
20:  $X_i \leftarrow X_i^*$ 
21: Parasitism phase
22: Randomly select  $X_j$  with  $i \neq j$ 
23: Create a parasite vector  $X^p$  from  $X_i$  using a random number according Equation (8)
     $f(X^p) > f(X_j)$ 
24:  $X_j \leftarrow X^p$ 
25: While  $i \leq N$ 
26: While stopping condition is not true

```

---

### 3. Problem Description

Whenever a cloud broker (CB) receives tasks from a user that must be scheduled, a cloud information service (CIS) issues a query to identify the services required by the user in order to execute the tasks; then, the tasks are scheduled for the services that were discovered. For example, tasks  $\{T_1, T_2, T_3, \dots, T_n\}$  might be submitted to the CB in an established interval of time. The fact that processing elements (VMs) are heterogeneous implies that there are uneven processing speeds and memory, which suggests that a task executed on different VMs results in variable costs of execution. Assume that the available VMs are  $\{V_1, V_2, V_3, \dots, V_m\}$  when the CB receives the tasks. As such, the tasks are scheduled on the available VMs, and the task execution is performed on a “first come, first serve” basis.

Our main goal is to accomplish greater utilisation of the VMs with a minimum makespan through the scheduling of the tasks on the VMs. Accordingly, the expected time to compute (ETC) of the tasks to be scheduled on individual VMs is used in the proposed strategy in order to settle on a scheduling choice. Moreover, the values of the ETC are obtained by using the ratio of the value of million instructions per second (MIPS) in a VM to the length of the task [86,87], as exemplified in Table 3.

The values of the ETC are normally presented in matrix form [86,87], where the number of tasks to be scheduled is presented row-wise in the matrix, while the number of available VMs is presented column-wise in the matrix. Hence, each row of the ETC matrix indicates the execution times of a given task for each VM, while each column indicates the execution times of each task on a given VM.

**Table 3.** Example of an ETC matrix.

|       | $T_1$     | $T_2$     | $T_3$     | $T_4$     |
|-------|-----------|-----------|-----------|-----------|
| $V_1$ | $T_1/V_1$ | $T_2/V_1$ | $T_3/V_1$ | $T_4/V_1$ |
| $V_2$ | $T_1/V_2$ | $T_2/V_2$ | $T_3/V_2$ | $T_4/V_2$ |
| $V_3$ | $T_1/V_3$ | $T_2/V_3$ | $T_3/V_3$ | $T_4/V_3$ |

Furthermore, the aim of this paper is to minimise the makespan and response time by obtaining the best combination of VMs with which tasks are to be executed. Let  $C_{ij}(i \in \{1, 2, 3, \dots, m\}, j \in \{1, 2, 3, \dots, n\})$  be the time needed to execute the  $j$ th task on the  $i$ th VM, where  $m$  and  $n$  are the numbers of VMs and tasks, respectively. Therefore, the fitness value of each organism is decided by applying Equation (9), which decides the extent of adaptation of the organism to the ecosystem.

$$fitness = \min\{C_{ij}, \forall \text{ tasks } j \text{ mapped to VM } i\} \quad i \in 1, 2, 3, \dots, m \tag{9}$$

#### 4. Enhanced Discrete Symbiotic Organism Search Strategy

The DSOS was proposed and used for solving task scheduling problems in the cloud, where task scheduling was formulated as a discrete optimisation problem. The enhanced discrete symbiotic organisms search (eDSOS) strategy proposed here is also used to solve the task scheduling problem in a cloud computing setting, as well as for the formation of task scheduling for discrete optimisation.

However, in the mutualism stage of eDSOS,  $X^{best}$ , which is the organism with the best fitting value, is substituted with  $X_k$ , as presented in Equations (10) through (13), in which  $X^{best}$  and  $X_k$  are the global and local optima, respectively. This is because this strategy extends the search space and increases the diversity of the ecosystem [83]. In the commensalism stage, modifications were made to improve the local exploitation ability. Here, the  $rand(-1, 1)$  coefficient is replaced with  $rand(0.4, 0.9)$ , as shown in Equation (12), as well as Algorithm 3.

It can be seen that the former coefficient considerably affects the convergence speed. With the wide range of  $[-1, 1]$ , the search space extends further, and this leads to a slow convergence speed. Therefore, a sufficient range of  $[0.4, 0.9]$ , as expressed in Equation (15), is proposed in order to reduce the computation time, but still ensure the accuracy [83]. The pseudocode and a flowchart of the proposed strategy are depicted in Algorithm 3 and Figure 1, respectively.

$$S_1(p) = X_i + r_1(X_k - ((X_i + X_j)/2)f_1) \tag{10}$$

$$S_2(p) = X_j + r_2(X_k - ((X_i + X_j)/2)f_2) \tag{11}$$

$$X_i^*(q) = \lceil s_1(p) \rceil \text{ mod } m + 1 \tag{12}$$

$$X_j^*(q) = \lceil s_2(p) \rceil \text{ mod } m + 1 \tag{13}$$

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

$$r_3 = rand(-1, 1) \tag{14}$$

$$r_3 = rand(0.4, 0.9) \tag{15}$$

The proposed system model of the enhanced discrete symbiotic organism search (eDSOS) strategy is shown in Figure 2; the model performs a scheduling process that minimises the makespan and the response time. The model consists of eight (8) steps such that each step consists of sub-steps that combine the processes of planning and performing. Firstly, the customer request step provides an interactive mechanism for users to submit their requests (tasks) with some requirements, such as data/resource requirements. Furthermore, the task manager/estimator manages the tasks, and later pushes the tasks to the scheduler.

The local resource manager constitutes the main aspect of the scheduling strategy. The local resource manager monitors and manages local virtual resources and obtains information regarding processing elements, memory, and bandwidth; this information is further submitted to the global manager, which subsequently forwards it to the scheduler for the task scheduling decision.

The scheduler component handles the preprocessing workload, task assignment, schedule refinement, queue of tasks, and eDSOS task dispatch. Tasks are preprocessed and assigned to the queue based on their expected time to compute, and the eDSOS task dispatch forms the scheduling algorithm, which moves tasks from the queue into the VMs. It first collects resource information (e.g., virtual machine capacity, computation of the execution cost of a virtual machine) from the global resource monitor, which also receives information about the availability of resources from the local resource manager. The eDSOS then assesses the capacity of available resources in order to schedule tasks with the minimum expected time to compute. The main contribution lies in that the eDSOS strategy improves the performance of the entire cloud system through the minimisation of the makespan and response time.

---

### Algorithm 3 Enhanced Discrete Symbiotic Organism Search Algorithm

---

- 1: Create and initialise the population of organisms in the ecosystem  $X = \{X_1, X_2, X_3 \dots X_N\}$
  - 2: Set up stopping criteria
  - 3:  $iteration\_number \leftarrow 0$
  - 4:  $X^{best} \leftarrow 0$
  - 5: **Do**
  - 6:  $iteration\_number \leftarrow iteration\_number + 1$
  - 7:  $i \leftarrow 0$
  - 8: **Do**
  - 9:  $i \leftarrow i + 1$   $j = 1$  to  $N$   $f(X_j) > f(X^{best})$  //  $f(X)$  is the fitness function
  - 10:  $X^{best} \leftarrow X_j$
  - 11: **Mutualism phase**
  - 12: Randomly select  $X_j$  with  $i \neq j$
  - 13: Update  $X_i^*$  and  $X_j^*$  using Equations (12) and (13)  $f(X_i^*) > f(X_i)$
  - 14:  $X_i \leftarrow X_i^*$   $f(X_j^*) > f(X_j)$
  - 15:  $X_j \leftarrow X_j^*$
  - 16: **Commensalism phase**
  - 17: Randomly select  $X_j$  with  $i \neq j$
  - 18:  $r_3 \leftarrow$  using Equation (15)
  - 19: Update  $X_i^*$  according to Equation (6)  $f(X_i^*) > f(X_i)$
  - 20:  $X_i \leftarrow X_i^*$
  - 21: **Parasitism phase**
  - 22: Randomly select  $X_j$  with  $i \neq j$
  - 23: Create a parasite vector  $X^p$  from  $X_i$  using a random number according to Equation (8)  $f(X^p) > f(X_j)$
  - 24:  $X_j \leftarrow X^p$
  - 25: **While**  $i \leq N$
  - 26: **While** stopping condition is not true
-

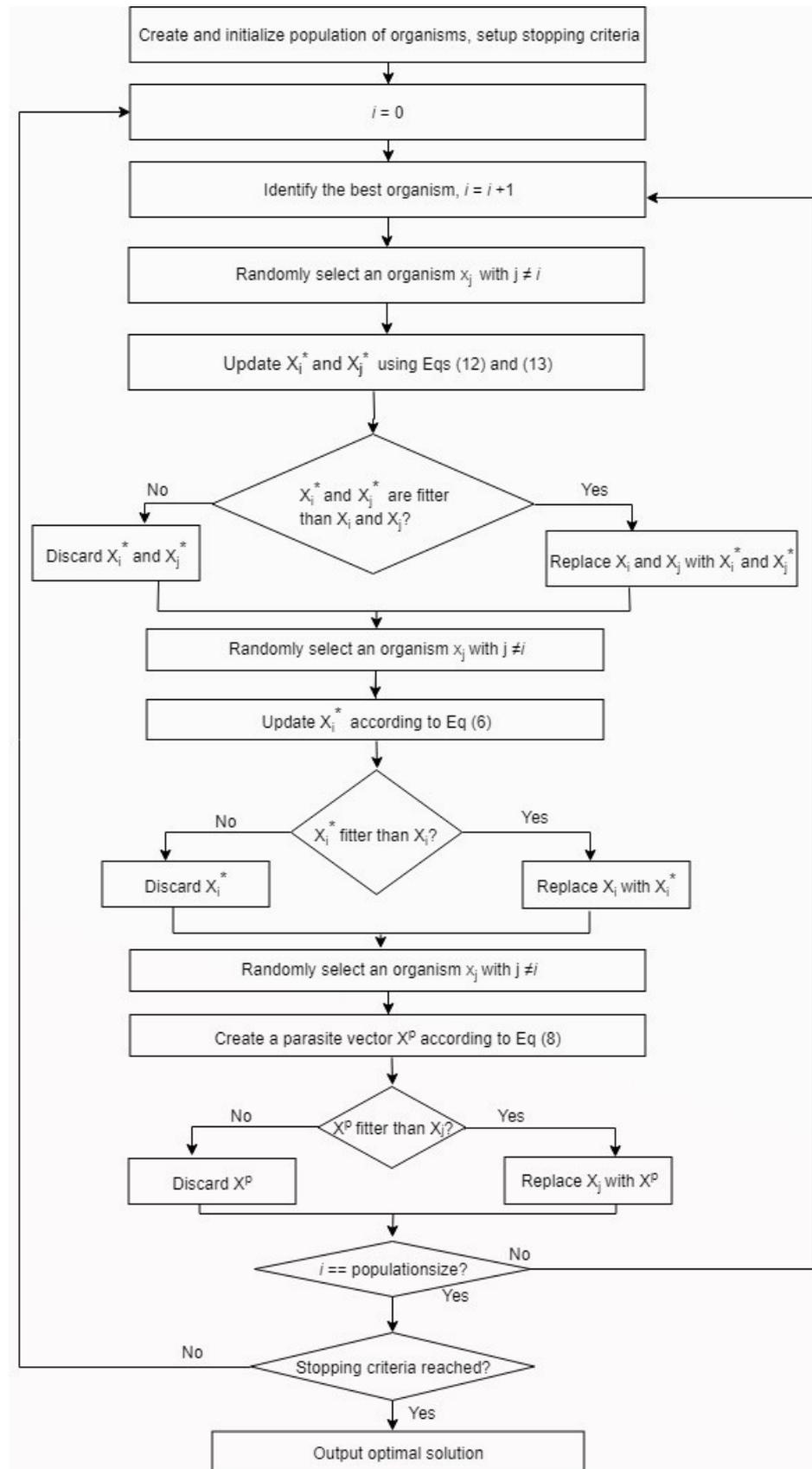


Figure 1. Flowchart of the eDSOS strategy.

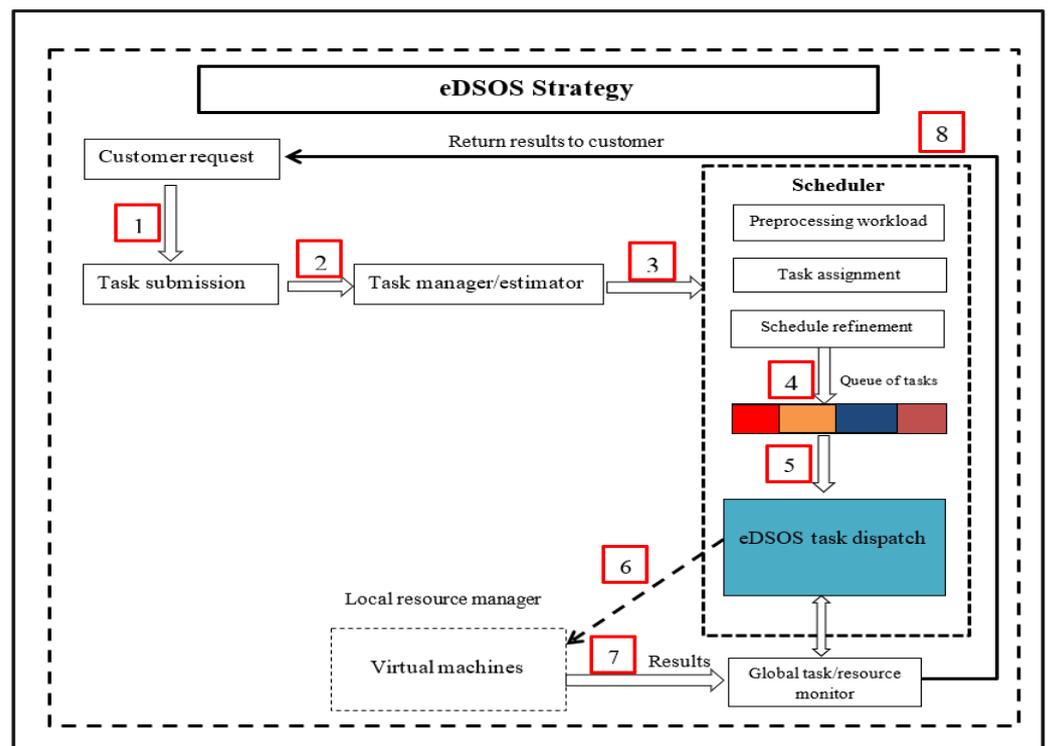


Figure 2. The proposed model of the eDSOS task scheduling strategy.

## 5. Simulation Settings

In order to validate our proposed algorithm, we tested it alongside the benchmark algorithm with the CloudSim simulator [88], a tool kit for mimicking distributed processing circumstances. We created two datacenters; each contained two hosts, and each individual host had storage (1 TB), RAM (20 GB), and bandwidth (10 GB/s). A time-shared scheduling policy for the VMs was used. The two hosts used dual-core and quad-core machines; both had an X86 architecture, a Xen virtual machine monitor (VMM), Linux OS, and 1,000,000 MIPS as the cumulative processing power. Furthermore, the number of VMs created was 20; all had 10 GB as the image size, a memory of 0.5 GB, a bandwidth of 1 GB/s, and a single processing element.

The VMs' processing power was in the range of 1000–10,000 MIPS. In addition, all of the VMs used a time-shared cloudlet scheduler policy and a Xen VMM. The task sizes were considered as cloudlet lengths and were generated from different distributions, which were normal, left-half, right-half, and uniform distributions. The normal distribution introduces more medium-sized tasks and fewer large- and small-sized tasks. The left-half distribution portrays a few small-sized tasks and more large-sized tasks, while the right-half distribution is the inverse.

The uniform distribution refers to an equivalent number of small-, medium-, and large-sized tasks. There were 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 instances created for each distribution. The larger numbers of instances empower us to increase our understanding of the scalability of the execution of these strategies for large problems. Table 4 presents the parameter settings for DSOS and eDSOS.

Table 4. Parameter settings.

| Algorithm      | Parameter            | Value |
|----------------|----------------------|-------|
| DSOS and eDSOS | Number of organisms  | 100   |
|                | Number of iterations | 1000  |

## 6. Results and Discussion

The average makespans for executing instances of tasks 10 times by utilising DSOS and eDSOS can be seen in Figures 3–6. All of the figures show a minimisation of the makespan by using eDSOS in most situations, especially in some task instances, such as with 100 or more. In addition, the percentages by which eDSOS improved compared to DSOS for different dataset instances are summarized in Tables 5–12, showing that the improvement of the performance of eDSOS compared to DSOS increases as the search space expands. Figures 7–10 indicate the response times of the VMs with DSOS and eDSOS; the figures demonstrate that eDSOS's response time is minimal for large-scale problem instances.

Moreover, in terms of convergence, for all of the dataset distributions, when applying data instances of 100–1000, our strategy demonstrates better-quality solution improvement in terms of the makespans obtained with both eDSOS and DSOS. It was demonstrated that the strategies exhibited an improvement in the quality of the solutions in the early part of the search. However, eDSOS was shown to have the ability to improve the quality of its solutions at certain points in the search process. The solution quality of eDSOS was found to be better than that of DSOS, particularly with larger problem sizes.

Tables 5–12 present the results of a two-sample *t*-test that was carried out to evaluate whether there were significant differences between the makespans and response times obtained with the eDSOS and DSOS when using similar stopping criteria for all task instances. The acceptable *p*-value must be less than alpha  $\alpha < 0.5$ ; as shown in the tables, almost all of the datasets had *p*-values that were less than this alpha value, which indicates a significant improvement in the proposed eDSOS strategy compared to the DSOS strategy. Therefore, it can be deduced that eDSOS outperforms DSOS as the search space expands.

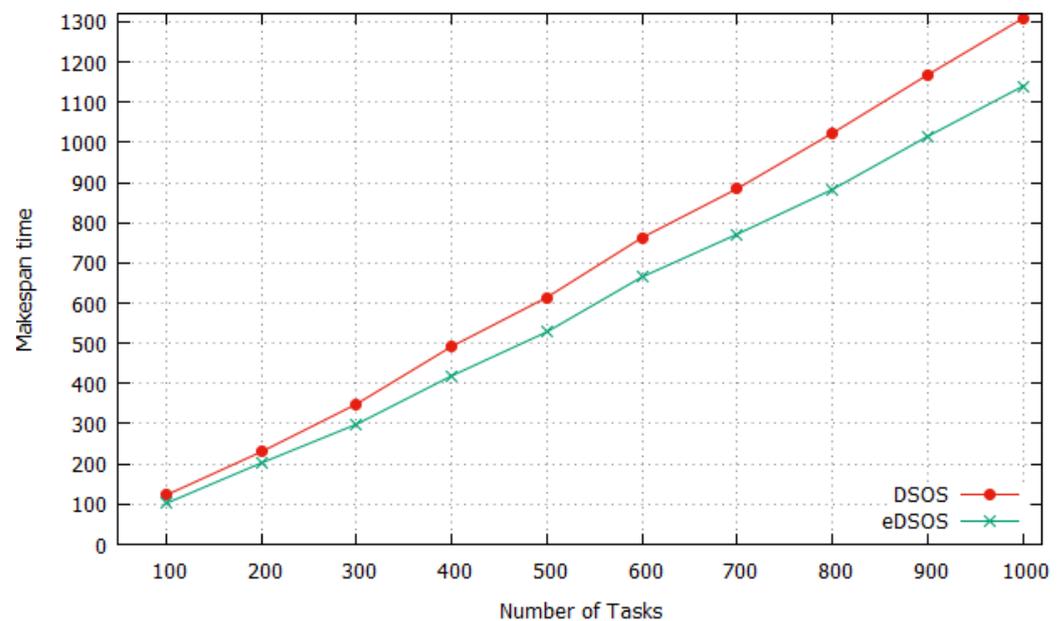


Figure 3. Makespan for dataset with the normal distribution.

The performance improvement with eDSOS is thought to be related to the diversification of the mutualism stage, as well as the modification of the commensalism stage to the reduced range of  $r_3$  for a better convergence rate. The diversification of the mutualism stage provides the search procedure the exploitative ability to traverse through different regions to find the best solution. In the parasitism stage, through the use of the parasite vector, premature convergence is avoided; thereby, the strategy does away with solutions that are not active and uses a more active solution that forces the search processes away from being entrapped in local optima.

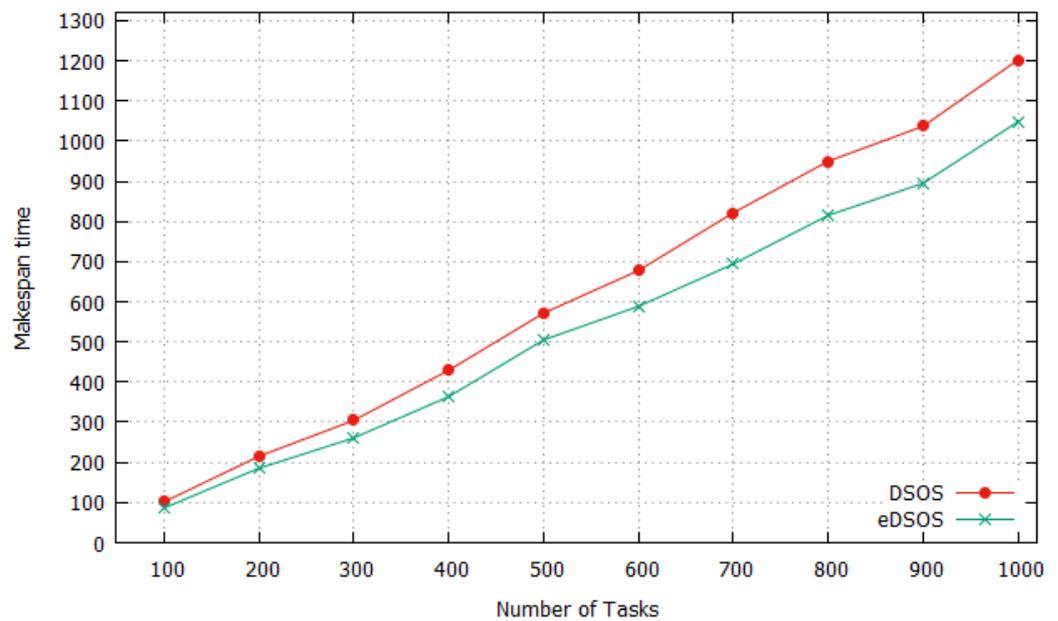


Figure 4. Makespan for the dataset with the left-half distribution.

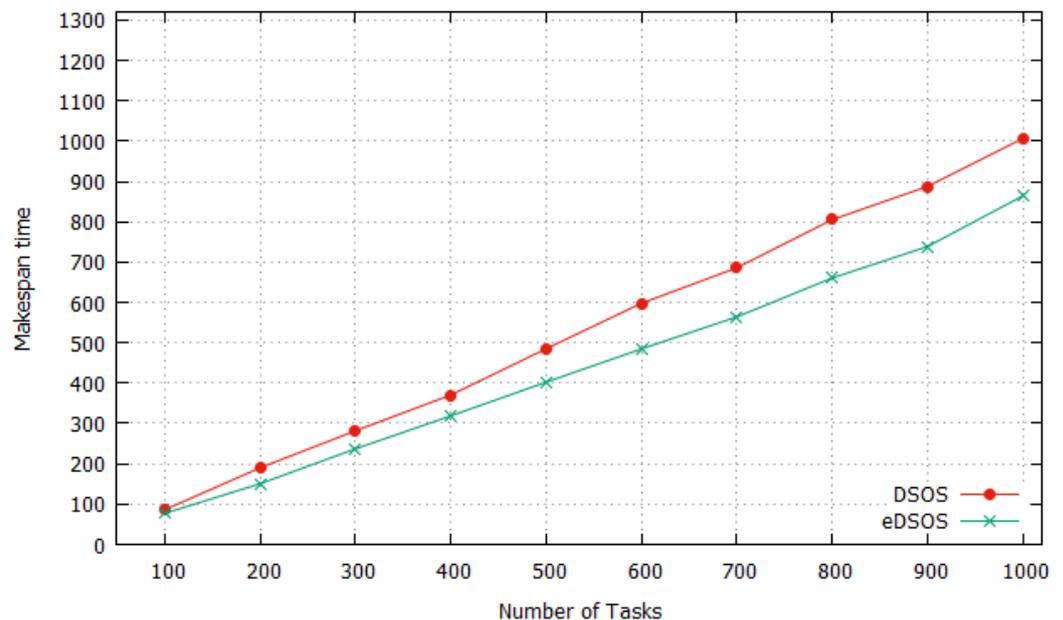


Figure 5. Makespan for dataset with the right-half distribution.

In the parasitism stage, the search procedure is empowered with further exploratory ability because it does not focus on only the best solution regions, which would likely cause the search to be entrapped in a specific region. The quality of this strategy can even be improved in later phases of the search process, which implies that eDSOS has a higher likelihood than DSOS of obtaining a solution that is near-optimal.

The convergence curves of DSOS and the proposed eDSOS algorithm with 100, 500, and 1000 instances of normally distributed tasks are shown in Figures 11–13. It can be observed from the convergence curves that the proposed eDSOS strategy has a better convergence accuracy than that of DSOS. The improved ecosystem diversity allows the eDSOS to escape entrapment in local optima, which improves the exploratory ability of the eDSOS procedure. The local exploitation strategy introduced in the commensalism phase allows each organism to perform a better local search, which also contributes to the better convergence accuracy.

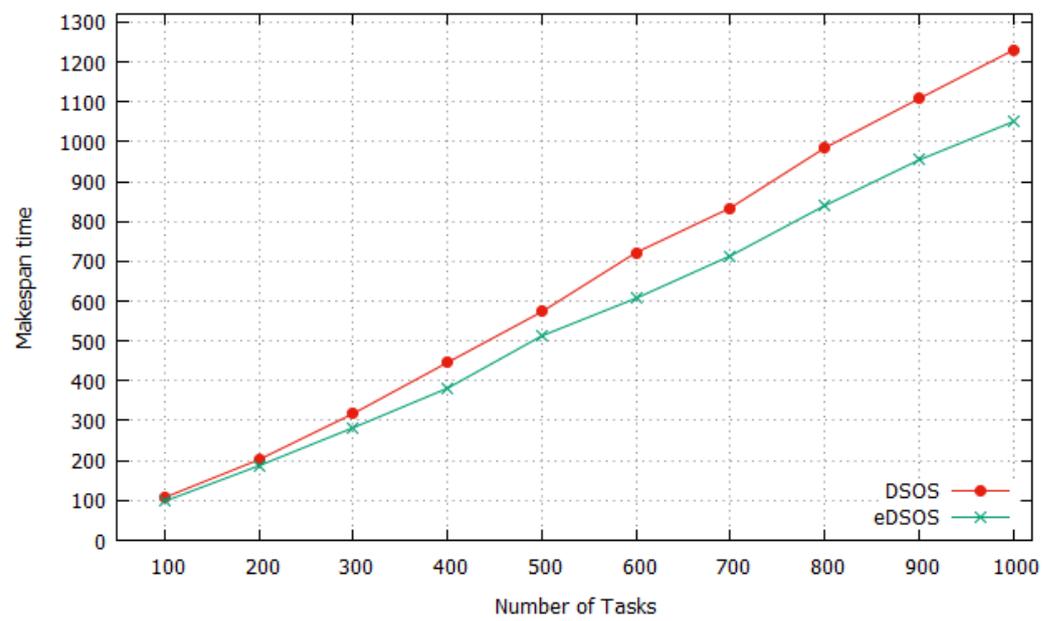


Figure 6. Makespan for the dataset with the uniform distribution.

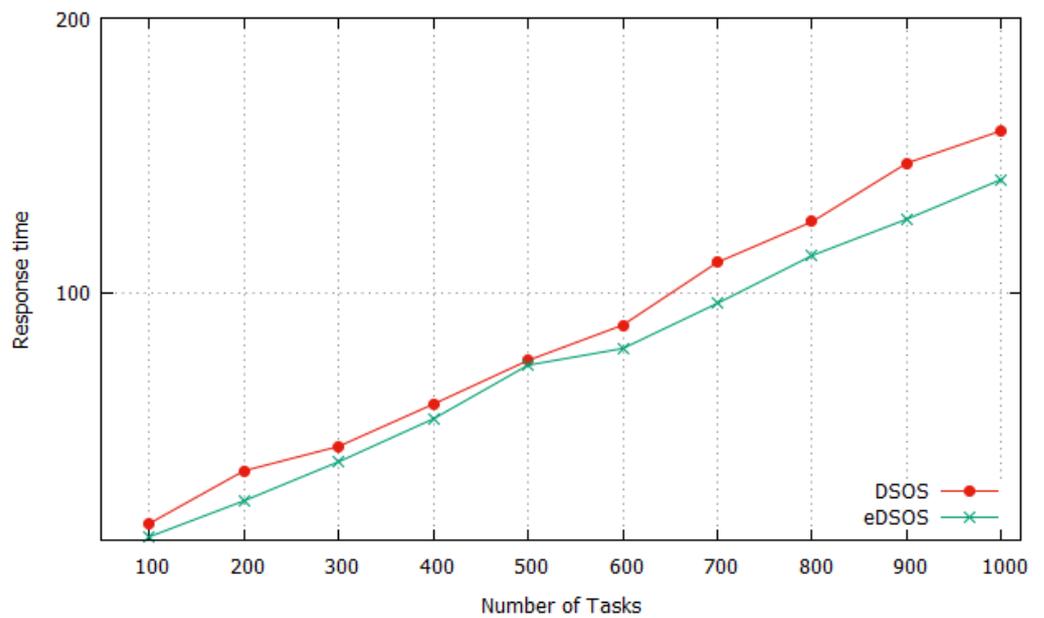


Figure 7. Response time for the dataset with the normal distribution.

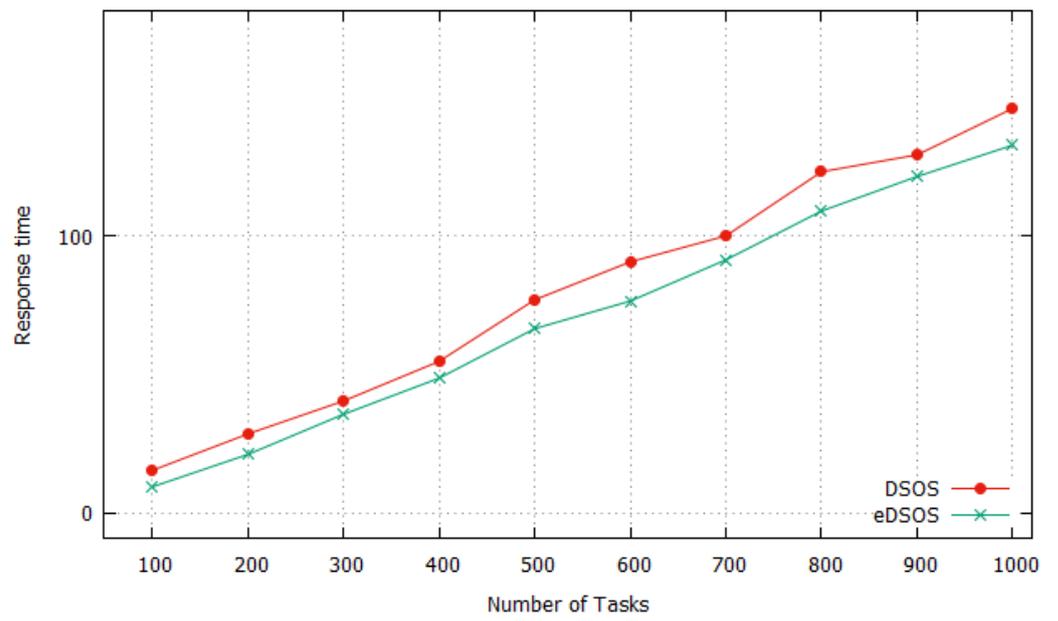


Figure 8. Response time for the dataset with the left-half distribution.

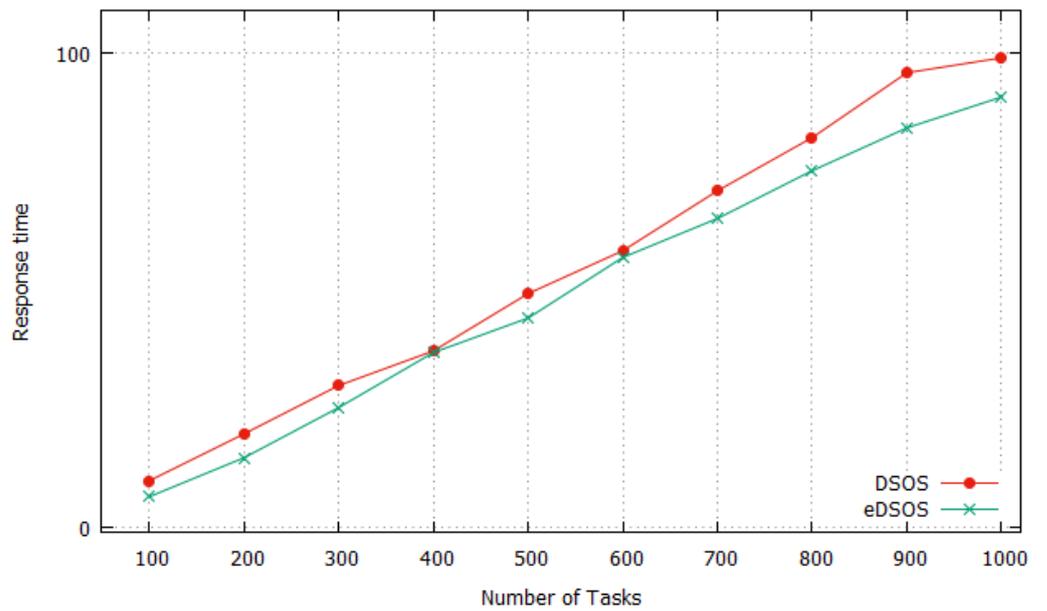


Figure 9. Response time for dataset with the right-half distribution.

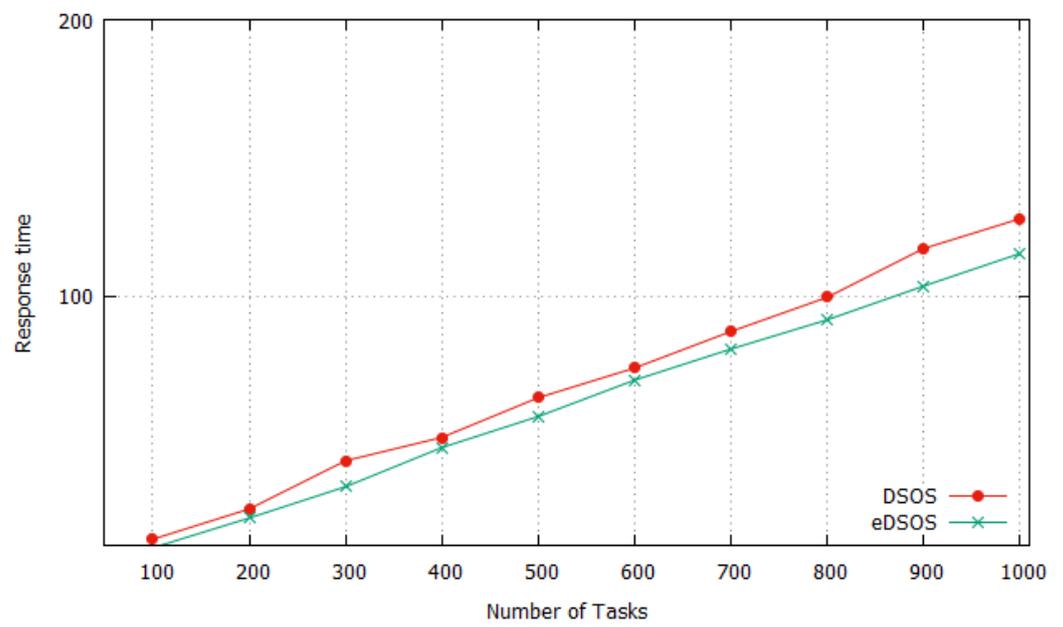


Figure 10. Response time for the dataset with the uniform distribution.

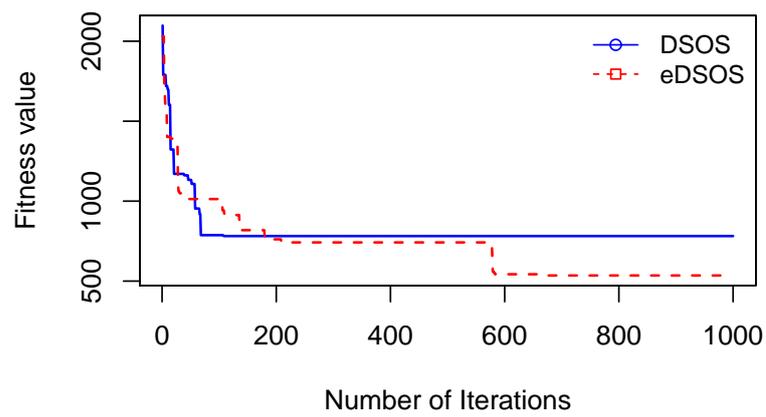


Figure 11. The convergence curves with 100 instances of tasks.

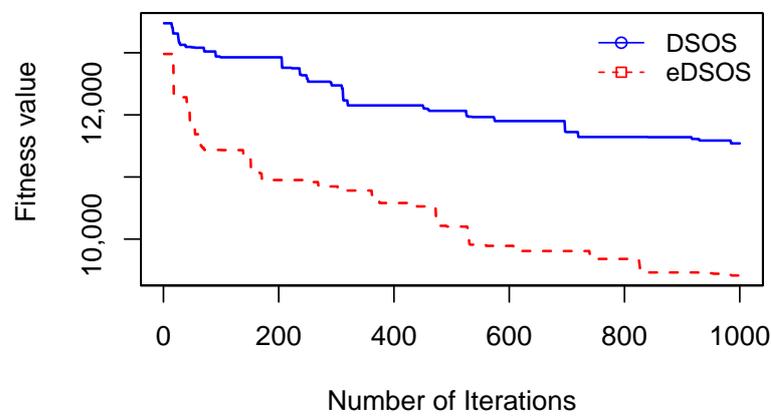
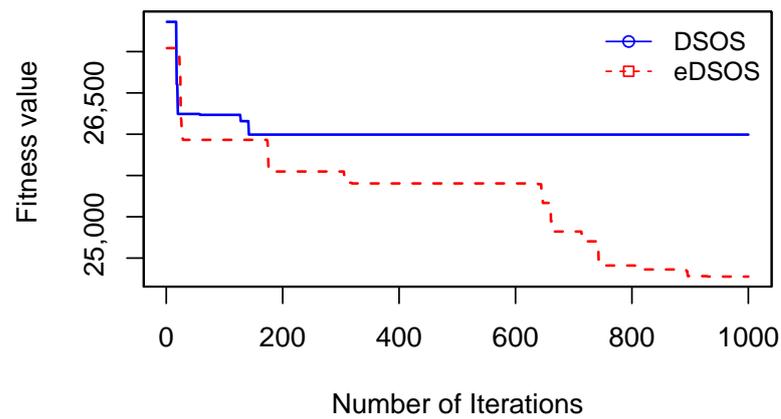


Figure 12. The convergence curves with 500 instances of tasks.



**Figure 13.** The convergence curves with 1000 instances of tasks.

**Table 5.** Comparison of the makespans of DSOS and eDSOS for the normally distributed dataset.

| Task Size | DSOS    | eDSOS   | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value        |
|-----------|---------|---------|-----------------|----------------------|------------------------|
| 100       | 122.48  | 102.02  | 20.05           | 5.4040               | $1.28 \times 10^{-6}$  |
| 200       | 230.25  | 202.55  | 13.68           | 4.4966               | $3.36 \times 10^{-5}$  |
| 300       | 349.04  | 298.90  | 16.77           | 5.1478               | $3.28 \times 10^{-6}$  |
| 400       | 492.34  | 419.30  | 17.42           | 6.837                | $5.50 \times 10^{-9}$  |
| 500       | 614.39  | 528.49  | 16.25           | 6.6650               | $1.07 \times 10^{-8}$  |
| 600       | 762.94  | 665.33  | 14.67           | 9.2745               | $4.69 \times 10^{-13}$ |
| 700       | 884.87  | 771.69  | 14.67           | 7.5949               | $2.93 \times 10^{-10}$ |
| 800       | 1022.90 | 882.76  | 15.88           | 9.3003               | $4.25 \times 10^{-13}$ |
| 900       | 1167.82 | 1014.86 | 15.07           | 9.0975               | $9.16 \times 10^{-13}$ |
| 1000      | 1308.14 | 1139.44 | 14.81           | 8.0474               | $5.10 \times 10^{-11}$ |

**Table 6.** Comparison of the makespans of DSOS and eDSOS for the dataset with the left-half distribution.

| Task Size | DSOS    | eDSOS   | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value        |
|-----------|---------|---------|-----------------|----------------------|------------------------|
| 100       | 101.34  | 86.31   | 17.41           | 4.9518               | $6.69 \times 10^{-6}$  |
| 200       | 215.08  | 185.27  | 16.09           | 5.0252               | $5.13 \times 10^{-6}$  |
| 300       | 304.43  | 260.67  | 16.79           | 4.5288               | $3.01 \times 10^{-5}$  |
| 400       | 429.11  | 363.38  | 18.09           | 6.6084               | $1.33 \times 10^{-8}$  |
| 500       | 571.23  | 504.68  | 13.19           | 5.8260               | $2.63 \times 10^{-7}$  |
| 600       | 677.50  | 588.24  | 15.17           | 5.7192               | $3.94 \times 10^{-7}$  |
| 700       | 821.29  | 693.86  | 18.37           | 8.6257               | $5.53 \times 10^{-12}$ |
| 800       | 949.67  | 814.97  | 16.53           | 6.3993               | $2.97 \times 10^{-8}$  |
| 900       | 1036.75 | 894.78  | 15.87           | 7.2671               | $1.04 \times 10^{-9}$  |
| 1000      | 1201.66 | 1047.84 | 14.68           | 10.099               | $2.15 \times 10^{-14}$ |

**Table 7.** Comparison of the makespans of DSOS and eDSOS for the dataset with the right-half distribution.

| Task Size | DSOS    | eDSOS  | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value        |
|-----------|---------|--------|-----------------|----------------------|------------------------|
| 100       | 85.92   | 77.00  | 11.58           | 2.4293               | 0.01825                |
| 200       | 189.86  | 150.41 | 26.23           | 9.3064               | $4.16 \times 10^{-13}$ |
| 300       | 281.57  | 236.94 | 18.84           | −1.0000              | 0.32150                |
| 400       | 370.46  | 318.64 | 16.26           | 5.1559               | $3.19 \times 10^{-6}$  |
| 500       | 484.99  | 402.29 | 20.56           | 8.3335               | $1.70 \times 10^{-11}$ |
| 600       | 597.84  | 484.65 | 23.35           | 9.1862               | $6.55 \times 10^{-13}$ |
| 700       | 685.77  | 564.03 | 21.58           | 8.1428               | $3.53 \times 10^{-11}$ |
| 800       | 804.90  | 660.39 | 21.88           | 10.5800              | $3.69 \times 10^{-15}$ |
| 900       | 887.43  | 738.52 | 20.16           | 11.1340              | $5.00 \times 10^{-16}$ |
| 1000      | 1006.27 | 863.99 | 16.47           | 18.5930              | $2.20 \times 10^{-16}$ |

**Table 8.** Comparison of the makespans of DSOS and eDSOS for the uniformly distributed dataset.

| Task Size | DSOS    | eDSOS   | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value       |
|-----------|---------|---------|-----------------|----------------------|-----------------------|
| 100       | 107.40  | 97.80   | 9.82            | 2.7944               | 0.00704               |
| 200       | 202.37  | 187.10  | 8.16            | 2.2366               | 0.02917               |
| 300       | 317.81  | 282.52  | 12.49           | 3.7330               | 0.00043               |
| 400       | 445.96  | 381.36  | 16.94           | 6.0255               | $1.2 \times 10^{-7}$  |
| 500       | 573.27  | 512.68  | 11.82           | 4.5433               | $2.9 \times 10^{-5}$  |
| 600       | 721.87  | 607.13  | 18.90           | 8.2421               | $2.4 \times 10^{-11}$ |
| 700       | 833.74  | 713.53  | 16.85           | 6.9794               | $3.2 \times 10^{-9}$  |
| 800       | 983.97  | 839.26  | 17.24           | 8.2634               | $2.2 \times 10^{-11}$ |
| 900       | 1107.71 | 953.91  | 16.12           | 10.519               | $4.6 \times 10^{-15}$ |
| 1000      | 1229.35 | 1049.98 | 17.08           | 11.448               | $2.2 \times 10^{-16}$ |

**Table 9.** Comparison of the response times of DSOS and eDSOS for the normally distributed dataset.

| Task Size | DSOS   | eDSOS  | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value       |
|-----------|--------|--------|-----------------|----------------------|-----------------------|
| 100       | 16.04  | 11.23  | 42.83           | 4.1806               | $9.9 \times 10^{-5}$  |
| 200       | 35.24  | 24.29  | 45.08           | 5.1731               | $3.0 \times 10^{-6}$  |
| 300       | 44.14  | 38.52  | 14.59           | 1.5192               | 0.1341                |
| 400       | 59.51  | 54.13  | 9.94            | 0.4537               | 0.6517                |
| 500       | 75.43  | 73.71  | 2.33            | 2.1002               | 0.0401                |
| 600       | 88.28  | 79.75  | 10.70           | 11.448               | $2.2 \times 10^{-16}$ |
| 700       | 111.15 | 96.23  | 15.50           | 3.4483               | 0.00106               |
| 800       | 125.91 | 113.61 | 10.83           | 1.4226               | 0.1602                |
| 900       | 147.24 | 126.87 | 16.06           | 2.1341               | 0.0371                |
| 1000      | 159.07 | 141.32 | 12.56           | −0.4574              | 0.6491                |

**Table 10.** Comparison of the response times of DSOS and eDSOS for the dataset with the left-half distribution.

| Task Size | DSOS   | eDSOS  | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value       |
|-----------|--------|--------|-----------------|----------------------|-----------------------|
| 100       | 15.39  | 9.48   | 62.34           | 4.3641               | $5.32 \times 10^{-5}$ |
| 200       | 28.54  | 21.15  | 34.94           | 4.2642               | $7.49 \times 10^{-5}$ |
| 300       | 40.40  | 35.61  | 13.45           | 1.9637               | 0.0544                |
| 400       | 54.68  | 48.62  | 12.46           | 2.0785               | 0.0421                |
| 500       | 76.88  | 66.47  | 15.66           | 3.2315               | 0.0020                |
| 600       | 90.54  | 76.43  | 18.46           | 4.0290               | 0.0002                |
| 700       | 99.93  | 91.28  | 9.48            | 1.4371               | 0.1561                |
| 800       | 123.01 | 108.87 | 12.99           | 0.9288               | 0.3568                |
| 900       | 129.10 | 121.30 | 6.43            | -0.5184              | 0.6062                |
| 1000      | 145.81 | 132.59 | 9.97            | 0.6189               | 0.5384                |

**Table 11.** Comparison of the response times of DSOS and eDSOS for the dataset with the right-half distribution.

| Task Size | DSOS  | eDSOS | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value |
|-----------|-------|-------|-----------------|----------------------|-----------------|
| 100       | 9.83  | 6.47  | 51.93           | 3.3613               | 0.0014          |
| 200       | 19.71 | 14.68 | 34.26           | 3.8736               | 0.0003          |
| 300       | 29.92 | 25.27 | 18.40           | 3.6055               | 0.0007          |
| 400       | 37.21 | 36.87 | 0.92            | 0.1791               | 0.8585          |
| 500       | 49.30 | 44.14 | 11.69           | 2.4222               | 0.0186          |
| 600       | 58.33 | 56.93 | 2.46            | 0.6615               | 0.5109          |
| 700       | 70.99 | 65.15 | 8.96            | 2.2098               | 0.0311          |
| 800       | 82.20 | 75.22 | 9.28            | -1.0545              | 0.2960          |
| 900       | 95.87 | 84.25 | 13.79           | 0.5383               | 0.5925          |
| 1000      | 99.02 | 90.76 | 9.10            | 1.3714               | 0.1755          |

**Table 12.** Comparison of the response times of DSOS and eDSOS for the uniformly distributed dataset.

| Task Size | DSOS   | eDSOS  | Improvement (%) | <i>t</i> -Test Value | <i>p</i> -Value      |
|-----------|--------|--------|-----------------|----------------------|----------------------|
| 100       | 12.22  | 9.14   | 33.70           | 3.355                | 0.0014               |
| 200       | 23.16  | 19.90  | 16.38           | 2.071                | 0.0428               |
| 300       | 40.59  | 31.22  | 30.01           | 4.189                | $9.7 \times 10^{-5}$ |
| 400       | 49.02  | 45.35  | 8.09            | 2.079                | 0.0421               |
| 500       | 63.39  | 56.59  | 12.02           | 2.818                | 0.0066               |
| 600       | 74.11  | 69.72  | 6.30            | 1.594                | 0.1165               |
| 700       | 87.29  | 80.95  | 7.83            | 2.135                | 0.0370               |
| 800       | 99.64  | 91.51  | 8.88            | 0.732                | 0.4670               |
| 900       | 117.27 | 103.70 | 13.09           | 0.821                | 0.4150               |
| 1000      | 128.05 | 115.48 | 10.89           | 0.683                | 0.4975               |

## 7. Conclusions

In this paper, an enhanced discrete variant of the metaheuristic symbiotic organism search algorithm was designed and implemented. The discrete variant of this strategy was inspired by the symbiotic interactions displayed by organisms in an ecosystem. Hence, the strategy imitates the different types of symbiotic interactions (mutualism, commensalism, and parasitism) in order to improve the quality of a particular objective function.

The proposed strategy was simulated with the CloudSim toolkit to schedule independent tasks. The makespans and response times of VMs were measured, and eDSOS was

shown to be better than the benchmark, DSOS. The improvement of the performance of eDSOS compared to DSOS increased in most instances when the search space expanded.

The best improvements in terms of the minimisation of the makespan with eDSOS were 13.68% to 20.05%, 14.68% to 18.37%, 11.58% to 26.23%, and 8.16% to 18.90% less than those with DSOS for the normal, left-half, right-half, and uniform distributions, respectively. Similarly, the best improvements in terms of the minimisation of the response time with eDSOS were 02.23% to 45.08%, 06.43% to 62.34%, 0.92% to 51.93% and 06.30% to 33.70% less than those with DSOS for the normal, left-half, right-half, and uniform distributions, respectively. Thus, the results of the two-sample *t*-test showed that eDSOS clearly outperformed DSOS through almost the entire process, and did so more glaringly in the later state of the search process.

The diversification of the mutualism stage, as well as its mutual benefit factor mechanism, allowed eDSOS to search for better solutions in new regions of the search space. In addition, reducing the range of random numbers for  $r_3$  provides the strategy with an optimum solution in terms of the convergence rate at the commensalism stage. In the parasitism stage, the parasite vector method aids in preventing premature convergence through the addition of perturbation in the ecosystem. The unique features of the basic SOS that are considered its advantage are its benefit factor, random number generation, and parasite vector mechanism. In the global and local search, a crucial role is played by these mechanisms. In addition to its global and local search ability, the SOS is parameterless, and it is easy to implement features, which is also regarded as an advantage.

The utilisation of eDSOS for other discrete optimisation problems and the use of more performance metrics can likely be tasks for future work. Additionally, hybrid and multi-objective versions of SOS can be used in cloud environments while considering other factors when scheduling tasks. The application of SOS to the scheduling of workflow and the use container orchestration can also be researched in the future.

**Author Contributions:** Conceptualization, S.S. and A.M.; methodology, all authors; software, S.S., A.M., and M.A.; validation, all authors; formal analysis, S.S., A.M., and M.A.; investigation, S.S.; resources, all authors; data curation, all authors.; writing—original draft preparation, S.S.; writing—review and editing, A.M., M.A., A.A., and F.H.A.; visualization, S.S. and M.A.; supervision, A.M., A.A., and F.H.A.; funding acquisition, RMC, Universiti Putra Malaysia. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Research Management Centre (RMC) of the Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia.

**Data Availability Statement:** Normal dataset: <http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0158229.s001> (accessed on 4 May 2021), left-half dataset: <http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0158229.s002> (accessed on 4 May 2021), right-half dataset: <http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0158229.s003> (accessed on 4 May 2021), uniform dataset: <http://www.plosone.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pone.0158229.s004> (accessed on 4 May 2021).

**Acknowledgments:** We are grateful to the staff members of the Department of Communication Technology and Networks of the Universiti Putra Malaysia.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

|       |   |
|-------|---|
| SOS   | Symbiotic Organism Search                   |
| DSOS  | Discrete Symbiotic Organism Search          |
| eDSOS | Enhanced Discrete Symbiotic Organism Search |
| ACO   | Ant Colony Optimisation                     |
| PSO   | Particle Swarm Optimisation                 |

|      |                                 |
|------|---------------------------------|
| GA   | Genetic Algorithm               |
| QoS  | Quality of Service              |
| SLA  | Service-Level Agreement         |
| IT   | Information Technology          |
| EC2  | Elastic Computing Cloud         |
| VM   | Virtual Machine                 |
| VMM  | Virtual Machine Monitor         |
| MIPS | Million Instructions per Second |
| ETC  | Expected Time to Compute        |
| CB   | Cloud Broker                    |
| CIS  | Cloud Information Service       |
| HPC  | High Performance Computing      |
| CPU  | Central Processing Unit         |

## References

- Durao, F.; Carvalho, J.F.S.; Fonseca, A.; Garcia, V.C. A systematic review on cloud computing. *J. Supercomput.* **2014**, *68*, 1321–1346. [CrossRef]
- Avram, M. Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective. *Procedia Technol.* **2014**, *12*, 529–534. [CrossRef]
- de Assunção, M.D.; di Costanzo, A.; Buyya, R. A cost-benefit analysis of using cloud computing to extend the capacity of clusters. *Clust. Comput.* **2010**, *13*, 335–347. [CrossRef]
- EC2, A. Amazon EC2. 2010. Available online: <https://aws.amazon.com/ec2/> (accessed on 29 May 2021).
- Tsai, C.W.; Rodrigues, J.J.P.C. Metaheuristic Scheduling for Cloud: A Survey *IEEE Syst. J.* **2014**, *8*, 279–291. [CrossRef]
- Aceto, G.; Botta, A.; De Donato, W.; Pescapè, A. Cloud monitoring: A survey. *Comput. Netw.* **2013**, *57*, 2093–2115. [CrossRef]
- Abdullahi, M.; Ngadi, M.A.; Dishing, S.I.; Abdulhamid, S.M.; eel Ahmad, B.I. An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. *J. Netw. Comput. Appl.* **2019**, *133*, 60–74. [CrossRef]
- Gabi, D.; Ismail, A.; Zainal, A.; Zakaria, Z.; Abraham, A. Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. *Neural Comput. Appl.* **2018**, *30*, 1845–1863. [CrossRef]
- Abdullahi, M.; Ngadi, M.A.; Abdulhamid, S.M. Symbiotic Organism Search optimization based task scheduling in cloud computing environment. *Future Gener. Comput. Syst.* **2016**, *56*, 640–650. [CrossRef]
- Garey, M.R.; Johnson, D.S. *Computer and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: New York, NY, USA, 1979.
- Ming, G.; Li, H. An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing. *Int. J. Innov. Res. Comput. Commun. Eng. (An Iso Certif. Organ.)* **2012**, *32972*, 217–223.
- Bhoi, U.; Ramanuj, P.N. Enhanced max-min task scheduling algorithm in cloud computing. *Int. J. Appl. Innov. Eng. Manag.* **2013**, *2*, 259–264.
- Munir, E.U.; Li, J.; Shi, S. QoS sufferage heuristic for independent task scheduling in grid. *Inf. Technol. J.* **2007**, *6*, 1166–1170. [CrossRef]
- Abdullahi, M.; Ngadi, M.A. Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. *PLoS ONE* **2016**, *11*, e0158229. [CrossRef]
- Wu, L.; Wang, Y.J.; Yan, C.K. Performance Comparison of Energy-Aware Task Scheduling with GA and CRO Algorithms in Cloud Environment. *Appl. Mech. Mater.* **2014**, *596*, 204–208. [CrossRef]
- Zhao, C.; Zhang, S.; Liu, Q.; Xie, J.; Hu, J. Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing. In Proceedings of the 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing, Beijing, China, 24–26 September 2009; pp. 1–4. [CrossRef]
- Tao, F.; Feng, Y.; Zhang, L.; Liao, T.W. CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Appl. Soft Comput. J.* **2014**, *19*, 264–279. [CrossRef]
- Zhu, Y.; Liu, P. Multi-dimensional constrained cloud computing task scheduling mechanism based on genetic algorithm. *Int. J. Online Eng.* **2013**, *9*, 15–18. [CrossRef]
- Zheng, S.M.; Gao, Z.N.; Wei, W.; Miao, Z.; Shao, R.M. Grid task scheduling genetic algorithm based on cloud model. *J. Univ. Electron. Sci. Technol. China* **2012**, *41*, 911–915.
- Lu, J.; Hu, W.; Shen, H.; Li, Y.; Liu, J. Particle swarm algorithm based task scheduling for many-core systems. In Proceedings of the 2017 12th IEEE Conference on Industrial Electronics and Applications, ICIEA 2017, Siem Reap, Cambodia, 18–20 June 2017; pp. 1860–1864. [CrossRef]
- Dordaie, N.; Navimipour, N.J. A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. *ICT Express* **2017**, *4*, 199–202. [CrossRef]

22. Zuo, X.; Zhang, G.; Tan, W. Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 564–573. [[CrossRef](#)]
23. Xu, A.; Yang, Y.; Mi, Z.; Xiong, Z. Task scheduling algorithm based on PSO in cloud environment. In Proceedings of the 2015 IEEE 12th International Conference on Ubiquitous Intelligence and Computing, 2015 IEEE 12th International Conference on Advanced and Trusted Computing, 2015 IEEE 15th International Conference on Scalable Computing and Communications, Beijing, China, 10–14 August 2015; pp. 1055–1061. [[CrossRef](#)]
24. Wang, M.; Zeng, W. A comparison of four popular heuristics for task scheduling problem in computational grid. In Proceedings of the 2010 6th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2010, Chengdu, China, 23–25 September 2010; pp. 1–4. [[CrossRef](#)]
25. Liu, Z.; Wang, X. A PSO-based algorithm for load balancing in virtual machines of cloud computing environment. *Lect. Notes Comput. Sci. (Incl. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinform.)* **2012**, *7331 LNCS*, 142–147. [[CrossRef](#)]
26. Ramezani, F.; Lu, J.; Hussain, F. Task Scheduling Optimization in Cloud Computing Applying Multi-Objective Particle Swarm Optimization. *Serv. Oriented Comput.* **2013**, *8274*, 237–251.
27. Popov, V. Particle Swarm Optimization Technique for Task-Resource Scheduling for Robotic Clouds. *Appl. Mech. Mater.* **2014**, *565*, 243–246. [[CrossRef](#)]
28. Netjinda, N.; Sirinaovakul, B.; Achalakul, T. Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization. *J. Supercomput.* **2014**, *68*, 1579–1603. [[CrossRef](#)]
29. Chitra, S.; Madhusudhanan, B.; Sakthidharan, G.R.; Saravanan, P. Local Minima Jump PSO for Workflow Scheduling in Cloud Computing Environments. *Adv. Comput. Sci. Eng.* **2014**, 1225–1234. [[CrossRef](#)]
30. Bilgaiyan, S.; Sagnika, S.; Das, M. Workflow scheduling in cloud computing environment using Cat Swarm Optimization. In Proceedings of the Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014, Gurgaon, India, 21–22 February 2014; pp. 680–685. [[CrossRef](#)]
31. Xue, S.; Li, M.; Xu, X.; Chen, J. An ACO-LB algorithm for task scheduling in the cloud environment. *J. Softw.* **2014**, *9*, 466–473. [[CrossRef](#)]
32. Tong, Z.; Li, K.; Xiao, Z.; Qin, X. H2ACO: An optimization approach to scheduling tasks with availability constraint in heterogeneous systems. *J. Internet Technol.* **2014**, *15*, 115–124. [[CrossRef](#)]
33. Sun, W.; Zhang, N.; Wang, H.; Yin, W.; Qiu, T. PACO: A period ACO based scheduling algorithm in cloud computing. In Proceedings of the 2013 International Conference on Cloud Computing and Big Data, CLOUDCOM-ASIA 2013, Fuzhou, China, 16–19 December 2013; pp. 482–486. [[CrossRef](#)]
34. Rajagopalan, A.; Modale, D.R.; Senthilkumar, R. Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm. In *Advances in Decision Sciences, Image Processing, Security and Computer Vision. Learning and Analytics in Intelligent Systems*; Springer: Cham, Switzerland, 2020; Volume 4, pp. 678–687. [[CrossRef](#)]
35. Chen, X.; Cheng, L.; Liu, C.; Liu, Q.; Liu, J.; Mao, Y.; Murphy, J. A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems. *IEEE Syst. J.* **2020**, *14*, 3117–3128 [[CrossRef](#)]
36. Velliangiri, S.; Karthikeyan, P.; Arul Xavier, V.; Baswaraj, D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Eng. J.* **2021**, *12*, 631–639. [[CrossRef](#)]
37. Liu, H.; Abraham, A.; Hassaniien, A.E. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Gener. Comput. Syst.* **2010**, *26*, 1336–1343. [[CrossRef](#)]
38. Qureshi, M.S.; Qureshi, M.B.; Fayaz, M.; Zakarya, M.; Aslam, S.; Shah, A. Time and Cost Efficient Cloud Resource Allocation for Real-Time Data-Intensive Smart Systems. *Energies* **2020**, *13*, 5706. [[CrossRef](#)]
39. Gabi, D.; Ismail, A.S.; Zainal, A.; Zakaria, Z.; Al-Khasawneh, A. Hybrid Cat Swarm Optimization and Simulated Annealing for Dynamic Task Scheduling on Cloud Computing Environment. *J. Inf. Commun. Technol.* **2018**, *3*, 435–467.
40. Khalid, A.; Aslam, S.; Aurangzeb, K.; Haider, S.I.; Ashraf, M.; Javaid, N. An efficient energy management approach using fog-as-a-service for sharing economy in a smart grid. *Energies* **2018**, *11*, 3500. [[CrossRef](#)]
41. Abdullahi, M.; Ngadi, M.A.; Dishing, S.I. Chaotic Symbiotic Organisms Search for Task Scheduling Optimization on Cloud Computing Environment. In Proceedings of the 2017 6th ICT International Student Project Conference (ICT-ISPC), Johor, Malaysia, 23–24 May 2017; pp. 1–4.
42. Kaur, S.; Verma, A. An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment. *Int. J. Inf. Technol. Comput. Sci.* **2012**, *4*, 74–79. [[CrossRef](#)]
43. Yang, Z.; Qin, X.; Li, W.; Yang, Y. Optimized Task Scheduling and Resource Allocation in Cloud Computing using PSO based Fitness Function. *Inf. Technol. J.* **2013**, *12*, 7090–7095. [[CrossRef](#)]
44. Zhan, S.; Huo, H. Improved PSO-based Task Scheduling Algorithm in Cloud Computing. *J. Inf. Comput. Sci.* **2012**, *13*, 3821–3829.
45. Ge, Y.; Wei, G. GA-based task scheduler for the cloud computing systems. In Proceedings of the 2010 International Conference on Web Information Systems and Mining, WISM 2010, Sanya, China, 23–24 October 2010; Volume 2, pp. 181–186. [[CrossRef](#)]
46. Pandey, S.; Wu, L.; Guru, S.M.; Buyya, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Proceedings of the International Conference on Advanced Information Networking and Applications, AINA, Perth, WA, Australia, 20–23 April 2010; pp. 400–407. [[CrossRef](#)]

47. Verma, A.; Kaushal, S. Bi-Criteria Priority based Particle Swarm Optimization workflow scheduling algorithm for cloud. In Proceedings of the 2014 Recent Advances in Engineering and Computational Sciences, RA ECS 2014, Chandigarh, India, 6–8 March 2014; pp. 6–8. [\[CrossRef\]](#)
48. Wang, J.; Li, F.; Zhang, L. QoS preferenceawareness task scheduling based on PSO and AHP methods. *Int. J. Control. Autom.* **2014**, *7*, 137–152. [\[CrossRef\]](#)
49. Han, C.; Zhou, G.; Zhou, Y. Binary Symbiotic Organism Search Algorithm for Feature Selection and Analysis. *IEEE Access* **2019**, *7*, 166833–166859. [\[CrossRef\]](#)
50. Zhang, B.; Sun, L.; Yuan, H.; Lv, J.; Ma, Z. An improved regularized extreme learning machine based on symbiotic organisms search. In Proceedings of the 2016 IEEE 11th Conference on Industrial Electronics and Applications, ICIEA 2016, Hefei, China, 5–7 June 2016; pp. 1645–1648. [\[CrossRef\]](#)
51. Tran, D.H.; Luong, D.L.; Chou, J.S. Nature-inspired metaheuristic ensemble model for forecasting energy consumption in residential buildings. *Energy* **2020**, *191*, 116552. [\[CrossRef\]](#)
52. Liu, D.; Li, H.; Wang, H.; Qi, C.; Rose, T. Discrete symbiotic organisms search method for solving large-scale time-cost trade-off problem in construction scheduling. *Expert Syst. Appl.* **2020**, *148*, 113230. [\[CrossRef\]](#)
53. Tran, D.H.; Cheng, M.Y.; Prayogo, D. A novel Multiple Objective Symbiotic Organisms Search (MOSOS) for time-cost-labor utilization tradeoff problem. *Knowl. Based Syst.* **2016**, *94*, 132–145. [\[CrossRef\]](#)
54. Prayogo, D.; Cheng, M.Y.; Wong, F.T.; Tjandra, D.; Tran, D.H. Optimization model for construction project resource leveling using a novel modified symbiotic organisms search. *Asian J. Civ. Eng.* **2018**, *19*, 625–638. [\[CrossRef\]](#)
55. Cheng, M.Y.; Prayogo, D.; Tran, D.H. Optimizing Multiple-Resources Leveling in Multiple Projects Using Discrete Symbiotic Organisms Search. *J. Comput. Civ. Eng.* **2016**, *30*, 287–290. doi:10.1061/(ASCE)CP.1943-5487.0000512. [\[CrossRef\]](#)
56. Tejani, G.G.; Savsani, V.J.; Patel, V.K.; Mirjalili, S. Truss optimization with natural frequency bounds using improved symbiotic organisms search. *Knowl. Based Syst.* **2018**, *143*, 162–178. [\[CrossRef\]](#)
57. Banerjee, S.; Chattopadhyay, S. Power Optimization of Three Dimensional Turbo Code Using a Novel Modified Symbiotic Organism Search (MSOS) Algorithm. *Wirel. Pers. Commun.* **2017**, *92*, 941–968. [\[CrossRef\]](#)
58. Yalçın, E.; Çam, E.; Taplamacıoğlu, M.C. A new chaos and global competitive ranking-based symbiotic organisms search algorithm for solving reactive power dispatch problem with discrete and continuous control variable. *Electr. Eng.* **2020**, *102*, 573–590. [\[CrossRef\]](#)
59. Duman, S. Symbiotic organisms search algorithm for optimal power flow problem based on valve-point effect and prohibited zones. *Neural Comput. Appl.* **2017**, *28*, 3571–3585. [\[CrossRef\]](#)
60. Prasad, D.; Mukherjee, V. A novel symbiotic organisms search algorithm for optimal power flow of power system with FACTS devices. *Eng. Sci. Technol. Int. J.* **2016**, *19*, 79–89. [\[CrossRef\]](#)
61. Das, S.; Bhattacharya, A. Symbiotic organisms search algorithm for short-term hydrothermal scheduling. *Ain Shams Eng. J.* **2016**. [\[CrossRef\]](#)
62. Guha, D.; Roy, P.K.; Banerjee, S. Symbiotic organism search algorithm applied to load frequency control of multi-area power system. *Swarm Evol. Comput.* **2017**, *33*, 46–67. [\[CrossRef\]](#)
63. Kahraman, H.T.; Dosoglu, M.K.; Guvenc, U.; Duman, S.; Sonmez, Y. Optimal scheduling of short-term hydrothermal generation using symbiotic organisms search algorithm. In Proceedings of the 4th International Istanbul Smart Grid Congress and Fair, ICSG 2016, Istanbul, Turkey, 20–21 April 2016. [\[CrossRef\]](#)
64. Saha, D.; Datta, A.; Das, P. Optimal coordination of directional overcurrent relays in power systems using Symbiotic Organism Search Optimisation technique. *IET Gener. Transm. Distrib.* **2016**, *10*, 2681–2688. [\[CrossRef\]](#)
65. Verma, S.; Saha, S.; Mukherjee, V. A novel symbiotic organisms search algorithm for congestion management in deregulated environment. *J. Exp. Theor. Artif. Intell.* **2017**, *29*, 59–79. [\[CrossRef\]](#)
66. Balachennaiah, P.; Suryakalavathi, M. Real Power Loss minimization using symbiotic organisms search algorithm. In Proceedings of the 12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015, New Delhi, India, 17–20 December 2015; Volume 4, pp. 1–6. [\[CrossRef\]](#)
67. Jamunaa, D.; Hasoon, F.N.; Mahanti, G. Symbiotic organisms search optimisation algorithm for synthesis of phase-only reconfigurable concentric circular antenna array with uniform amplitude distribution. *Int. J. Electron. Lett.* **2020**, *8*, 460–471. [\[CrossRef\]](#)
68. Gharehchopogh, F.S.; Shayanfar, H.; Gholizadeh, H. A comprehensive survey on symbiotic organisms search algorithms. *Artif. Intell. Rev.* **2020**, *53*, 2265–2312. [\[CrossRef\]](#)
69. Tejani, G.G.; Savsani, V.J.; Patel, V.K. Adaptive symbiotic organisms search (SOS) algorithm for structural design optimization. *J. Comput. Des. Eng.* **2016**, *3*, 226–249. [\[CrossRef\]](#)
70. Talatahari, S. Symbiotic organisms search for optimum design of frame and grillage systems. *Asian J. Civ. Eng. (BHRC)* **2016**, *17*, 299–313.
71. Kanimozhi, G.; Rajathy, R.; Kumar, H. Minimizing energy of point charges on a sphere using symbiotic organisms search algorithm. *Int. J. Electr. Eng. Inform.* **2016**, *8*, 29–44. [\[CrossRef\]](#)
72. Eki, R.; Vincent, F.Y.; Budi, S.; Redi, A.P. Symbiotic Organism Search (SOS) for Solving the Capacitated Vehicle Routing Problem. *Appl. Soft Comput.* **2015**, *9*, 873–877.

73. Vincent, F.Y.; Redi, A.P.; Yang, C.L.; Ruskartina, E.; Santosa, B. Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. *Appl. Soft Comput. J.* **2017**, *52*, 657–672. [[CrossRef](#)]
74. Wang, Y.; Wu, Y.; Xu, N. Discrete symbiotic organism search with excellence coefficients and self-escape for traveling salesman problem. *Comput. Ind. Eng.* **2019**, *131*, 269–281. [[CrossRef](#)]
75. Dib, N. Synthesis of antenna arrays using symbiotic organisms search (SOS) algorithm. In Proceedings of the 2016 IEEE Antennas and Propagation Society International Symposium, APSURSI 2016, Fajardo, PR, USA, 26 June–1 July 2016; pp. 581–582. [[CrossRef](#)]
76. Dib, N.I. Design of Linear Antenna Arrays with Low Side Lobes Level Using Symbiotic Organisms Search. *Prog. Electromagn. Res. B* **2016**, *68*, 55–71. [[CrossRef](#)]
77. Dosoglu, M.K.; Guvenc, U.; Duman, S.; Sonmez, Y.; Kahraman, H.T. Symbiotic organisms search optimization algorithm for economic/emission dispatch problem in power systems. *Neural Comput. Appl.* **2018**, *29*, 721–737. [[CrossRef](#)]
78. Secui, D.C. A modified Symbiotic Organisms Search algorithm for large scale economic dispatch problem with valve-point effects. *Energy* **2016**, *113*, 366–384. [[CrossRef](#)]
79. Guvenc, U.; Duman, S.; Dosoglu, M.K.; Kahraman, H.T.; Sonmez, Y.; Yilmaz, C. Application of Symbiotic Organisms Search Algorithm to solve various economic load dispatch problems. In Proceedings of the 2016 International Symposium on INnovations in Intelligent SysTems and Applications, INISTA 2016, Sinaia, Romania, 2–5 August 2016; pp. 1–7. [[CrossRef](#)]
80. Sonmez, Y.; Kahraman, H.T.; Dosoglu, M.K.; Guvenc, U.; Duman, S. Symbiotic organisms search algorithm for dynamic economic dispatch with valve-point effects. *J. Exp. Theor. Artif. Intell.* **2017**, *29*, 495–515. [[CrossRef](#)]
81. Tiwari, A.; Pandit, M. Bid based economic load dispatch using symbiotic organisms search algorithm. In Proceedings of the 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016, Coimbatore, India, 17–18 March 2016; pp. 1073–1078. [[CrossRef](#)]
82. Rajathy, R.; Taraswinee, B.; Suganya, S. A novel method of using symbiotic organism search algorithm in solving security-constrained economic dispatch. In Proceedings of the IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2015, Nagercoil, India, 19–20 March 2015; pp. 1–8. [[CrossRef](#)]
83. Do, D.T.; Lee, D.; Lee, J. Material optimization of functionally graded plates using deep neural network and modified symbiotic organisms search for eigenvalue problems. *Compos. Part B Eng.* **2019**, *159*, 300–326. [[CrossRef](#)]
84. Cheng, M.Y.; Prayogo, D. Symbiotic Organisms Search: A new metaheuristic optimization algorithm. *Comput. Struct.* **2014**, *139*, 98–112. [[CrossRef](#)]
85. Khorram, E.; Zarei, H. Multi-objective optimization problems with Fuzzy relation equation constraints regarding max-average composition. *Math. Comput. Model.* **2009**, *49*, 856–867. [[CrossRef](#)]
86. Loo, S.M.; Wells, B.E. Task Scheduling in a Finite-Resource, Reconfigurable Hardware/Software Codesign Environment. *INFORMS J. Comput.* **2006**, *18*, 151–172. [[CrossRef](#)]
87. Demiroz, B.; Topcuoglu, H.R. Static task scheduling with a unified objective on time and resource domains. *Comput. J.* **2006**, *49*, 731–743. [[CrossRef](#)]
88. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *1*, 23–50. [[CrossRef](#)]