applied sciences

MDPI

Article

A Discrete Prey–Predator Algorithm for Cloud Task Scheduling

Doaa Abdulmoniem Abdulgader [1], Adil Yousif [2,*] and Awad Ali [2]

[1] Department of Computer Science, Faculty of Computer Science, University of Kassala, Kassala 31113, Sudan; doaahamza@kassalauni.edu.sd
[2] Department of Computer Science, College of Science and Arts-Sharourah, Najran University, Sharourah 68341, Saudi Arabia; aaomar@nu.edu.sa
* Correspondence: ayalfaki@nu.edu.sa

Abstract: Cloud computing is considered a key Internet technology. Cloud providers offer services through the Internet, such as infrastructure, platforms, and software. The scheduling process of cloud providers' tasks concerns allocating clients' tasks to providers' resources. Several mechanisms have been developed for task scheduling in cloud computing. Still, these mechanisms need to be optimized for execution time and makespan. This paper presents a new task-scheduling mechanism based on Discrete Prey–Predator to optimize the task-scheduling process in the cloud environment. The proposed Discrete Prey–Predator mechanism assigns each scheduling solution survival values. The proposed mechanism denotes the prey's maximum surviving value and the predator's minimum surviving value. The proposed Discrete Prey–Predator mechanism aims to minimize the execution time of tasks in cloud computing. This paper makes a significant contribution to the field of cloud task scheduling by introducing a new mechanism based on the Discrete Prey–Predator algorithm. The Discrete Prey–Predator mechanism presents distinct advantages, including optimized task execution, as the mechanism is purpose-built to optimize task execution times in cloud computing, improving overall system efficiency and resource utilization. Moreover, the proposed mechanism introduces a survival-value-based approach, as the mechanism introduces a unique approach for assigning survival values to scheduling solutions, differentiating between the prey's maximum surviving value and the predator's minimum surviving value. This improvement enhances decision-making precision in task allocation. To evaluate the proposed mechanism, simulations using the CloudSim simulator were conducted. The experiment phase considered different scenarios for testing the proposed mechanism in different states. The simulation results revealed that the proposed Discrete Prey–Predator mechanism has shorter execution times than the firefly algorithm. The average of the five execution times of the Discrete Prey–Predator mechanism was 270.97 s, while the average of the five execution times of the firefly algorithm was 315.10 s.

Keywords: cloud computing; job scheduling; optimization; Prey–Predator algorithm

check for updates

Citation: Abdulgader, D.A.; Yousif, A.; Ali, A. A Discrete Prey–Predator Algorithm for Cloud Task Scheduling. *Appl. Sci.* **2023**, *13*, 11447. https://doi.org/10.3390/app132011447

Academic Editors: Ce Li and Bei Guan

Received: 1 August 2023
Revised: 7 October 2023
Accepted: 12 October 2023
Published: 18 October 2023

Copyright: © 2023 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 is a relatively new technology that allows users to offload their computing tasks from personal computers to cloud service providers via the Internet [1,2]. Cloud computing was designed as an alternative infrastructure for physical devices, providing greater flexibility, scalability, and utilization for the computing process [3]. Data centers in cloud computing offer virtualized, remote access to resources anytime and anywhere to meet the users' cloud requirements [4]. Cloud users submit a large volume of tasks that need to be assigned to the proper resources based on the objective function and the scheduling mechanism, as shown in Figure 1. The goal of a task-scheduling algorithm in cloud computing is to effectively match a user's tasks with the appropriate resources provided by cloud providers [5,6]. Scheduling is a challenging and critical process that consists in managing a massive number of virtual machine (VM) requests in a cloud computing

environment. Several cloud-task-scheduling mechanisms have been developed to allocate tasks to resources to reduce their execution time.

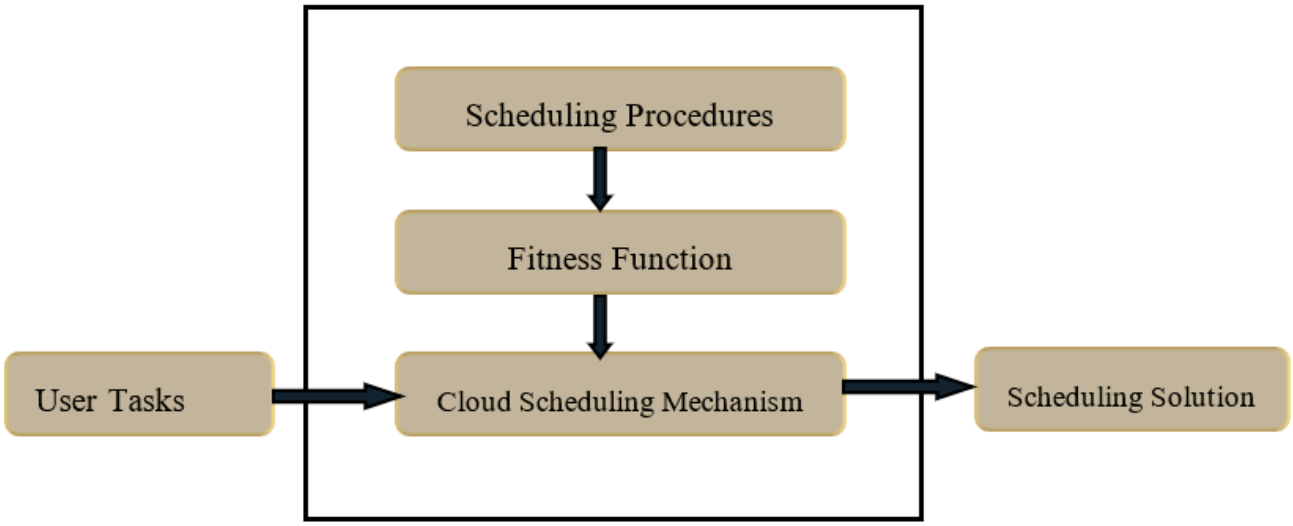

Figure 1. Cloud task-scheduling process.

Several nature-inspired metaheuristics, such as particle swarm optimization, ant colony optimization, and the firefly algorithm, have been employed as optimization techniques for the cloud-task-scheduling problem [7–10]. These methods can find satisfactory solutions but only sometimes optimal ones. When scheduling tasks, allocating tasks to available and appropriate resources should consider various factors, such as cost, minimum completion time, task execution time, deadline, and load balancing [11–13].

In cloud computing, resources can be shared through virtualization. These resources are represented as virtual machines (VMs), each encompassing several computational resources with limited capacities, such as processing power, memory size, network, and bandwidth [4,14]. Tasks in the cloud are represented as a collection of individual atomic tasks submitted by cloud users [15]. Each task has specific requirements, such as the required processing length in millions of instructions (MI). A cloud provider may receive a set of tasks with varying requirements that need to be scheduled for execution in a timely manner while effectively utilizing all available and ready resources [16,17]. The scheduling problem can be formulated as finding the optimal solution to allocate tasks T = {T1, T2, T3, . . ., Tn} to suitable virtual machines VM = {VM1, VM2, . . ., VMn} while considering the requirements of the tasks and the capacities of the resources [11,18].

Several mechanisms have been developed for task scheduling in cloud computing. Still, these mechanisms need to be optimized in terms of execution time and makespan. This paper presents a new task-scheduling mechanism based on the Discrete Prey–Predator mechanism to optimize the task-scheduling process in the cloud environment. The Prey–Predator method is a promising optimization mechanism that emerged to handle challenging problems with huge search space [19]. The inspiration for Prey–Predator is based on the social predating behavior of a predator, such as a lion or a cheetah, to chase prey, such as rabbits or gazelles [19,20]. Surafel Tilahun and Hong Choon Ong developed the Prey–Predator optimization method in 2015 [19]. The Prey–Predator optimization algorithm generates random solutions based on the performance of prey and predators on the objective function.

In Prey–Predator optimization, prey individuals try to be far from predators to save their lives, and predators attempt to catch the weakest prey. Hence, the best prey is the prey with the highest speed among other preys, known as the survival value [19–21]. The proposed Discrete Prey–Predator mechanism assigns each scheduling solution a survival value. The prosed mechanism denotes the maximum surviving value as prey and the minimum surviving value as the predator [19,20].

This paper contains five sections. Section 2 provides reviews of relevant works. Section 3 details the proposed Discrete Prey–Predator scheduling mechanism. Section 4 presents the evaluation and experimentation results. Finally, the conclusion is presented in Section 5.

2. Related Work

The cloud computing environment utilizes numerous task-scheduling algorithms, including heuristic, metaheuristic, and hybrid algorithms, to achieve efficient task execution times with optimized throughput systems. In some studies, heuristic algorithms such as the max–min algorithm and the SJF (Shortest Job First) algorithm have been employed for cloud job scheduling. Mao et al. utilized the max–min algorithm to develop a task allocation algorithm that prioritizes speed and minimizes response time and makespan [22,23]. Furthermore, Elzekiet et al. optimized the max–min algorithm to compute the expected execution time [24]. Mondal et al. proposed an algorithm based on the SJF algorithm to reduce execution and turnaround time by selecting the task with the shortest execution time [25]. Alworafi et al. presented the Shortest Job First (SJF) algorithm to create a scheduling mechanism that reduces completion time while maximizing resource utilization [22].

In metaheuristic mechanisms, authors Guo et al. introduced a task-scheduling mechanism based on particle swarm optimization (PSO) to allocate tasks to available resources to minimize execution time [26]. The implementation of particle swarm optimization (PSO) by Yang et al. focuses on task scheduling by dividing tasks into subtasks and allocating each subtask to a suitable resource using a random order. This approach decreases the fitness value, makespan, and total completion time. Tawfeek et al. presented an algorithm for task scheduling based on the ant colony optimization (ACO) mechanism [27]. The goal of the ACO mechanism is to minimize the makespan [28–30]. Several algorithms utilized the genetic algorithm for optimizing the cloud-task-scheduling process [31]. The researchers in [7] applied the firefly algorithm as an optimization method for scheduling jobs in cloud computing [32]. The firefly mechanism was introduced to minimize the execution times. The study in [33] proposed an Ions Motion Optimization (IMO) algorithm for scheduling cloud jobs. The IMO mechanism consists of two phases, the liquid and the crystal phases. These phases maintain a balance between rapid convergence and avoiding local optima in the algorithm's performance [33].

The authors in [17] converted uncertain factors into interval parameters, effectively and concurrently encompassing factors like makespan, scheduling cost, and task completion rate into the scheduling process. Subsequently, they introduced a new model, the Interval Many-Objective Cloud job Scheduling Optimization (I-MCTSO) model, tailored to replicate real-world cloud computing job-scheduling scenarios.

Evolution optimization has been implemented in several works for cloud task scheduling [34–36]. The study [34] introduced a new job-scheduling methodology for heterogeneous cloud environments, utilizing the innovative Advanced Phasmatodea Population Evolution algorithm. This algorithm expedites the solution discovery process by enhancing the convergence of the closest optimal solutions. Additionally, the study incorporates a restart strategy to safeguard against local optimization, maintaining a harmonious equilibrium between exploration and development capabilities within the algorithm.

The authors in [35] introduced a dual-phase metaheuristic job-scheduling mechanism for cloud computing. Initially, the mechanism employs a clustering method to organize computing nodes into efficient clusters. Each node undergoes training across various utilization levels. Subsequently, they combined the Sparrow Search metaheuristic with the differential evolution (DE) mechanism to enhance the search efficiency for identifying suitable job–resource combinations. A new job scheduler to address the task-scheduling challenges within the cloud computing environment is proposed in [37]. The proposed job-scheduling method is constructed upon two novel enhancements in the conventional differential evolution approach. The initial enhancement focuses on refining the scaling factor, incorporating dynamically generated numerical values based on the current iteration,

thereby enhancing both exploration and exploitation operators. The second enhancement aims to improve the exploitation operator of the classical DE, resulting in improved outcomes with fewer iterations. To establish the effectiveness of HDE, the study implemented multiple tests using the CloudSim simulator, demonstrating the superior performance of the method [37]. The study in [38], introduced a novel hybrid approach for scheduling workflows in computational cloud, integrating genetic algorithm optimization with the differential evolution mechanism. The integration process seeks to explore the impact of heterogeneous computational cloud on workflow job scheduling. The primary objectives of the study are to minimize makespan and to assess the suitability of metaheuristic technology in a distributed environment.

Prey–Predator has been implemented in several research studies [39–41]. The best prey is dedicated to thoroughly exploiting its immediate vicinity, while the predator's task is to traverse the search space in pursuit of promising regions. Ordinary prey, positioned between these extremes, experiences the combined effects of exploration and exploitation behaviors. This algorithm has undergone rigorous testing and has demonstrated its effectiveness in tackling a range of challenges spanning diverse fields, including engineering, tourism, and management. Originally designed for continuous problems, it has become apparent that many real-world issues are not continuous in nature. Consequently, the research in [42] extends the Prey–Predator algorithm to accommodate discrete problems. To assess its efficacy, we employ examination timetabling as a test case. The simulation results, supported by robust statistical analyses, indicate that this approach is on par with the best-performing results documented in the literature for certain benchmark problems.

The research in [39] introduced a model centered around the concept of nature-based system survivability. This model was subjected to simulation and performance assessment, specifically within the context of mobile ad hoc wireless networks. The primary purpose of the survivability model is to ensure that mobile wireless distributed systems can maintain uninterrupted packet delivery throughout their designated missions, even in the face of security attacks. To achieve this, research has developed a communal defense algorithm inspired by prey–predator dynamics.

3. The Proposed Discrete Prey–Predator Algorithm for Cloud Task Scheduling

This section illustrates the steps, mapping, and equations of the proposed Discrete Prey–Predator algorithm for cloud task scheduling.

3.1. Description of the Proposed Mechanism

This study presents an application of the Discrete Prey–Predator algorithm to cloud task scheduling to allocate cloud tasks $N = [t_1, t_2, t_3, \ldots, t_n]$ to the cloud providers' resources $M = [s_1, s_2, s_3, \ldots, s_m]$. The proposed Discrete Prey–Predator algorithm generates an initial set of feasible solutions representing cloud schedules. For each solution member x_i, the algorithm calculates its fitness value, which is defined as the survival value. The calculation of the survival value is based on the encoded objective function of the proposed Discrete Prey–Predator algorithm. The proposed Discrete Prey–Predator algorithm for cloud task scheduling considers the execution time as the objective function for the population solution members (x_i).

In the next step, the proposed algorithm considers member x_i with the minimum survival value as the predator and member x_j with the maximum surviving value as prey. The prey individual that finds a hiding place is the prey individual with the best surviving value, and it is not affected by predators. Other prey individuals in the population move towards prey with better survival values.

3.2. Pseudo-Code for the Proposed Discrete Prey–Predator Algorithm for Cloud Task Scheduling

The following is the pseudo-code of the proposed Discrete Prey–Predator Algorithm 1 for cloud task scheduling:

Algorithm 1 Discrete Prey–Predator Algorithm for Cloud Task Scheduling

Initialize:
$$X_0 = [x_1, x_2, x_3, \ldots, x_m]$$
$$r = \text{iterations_max}$$
Set the initial value of $r = 1$
For each resource, do:
    Define the speed for each cloud resource in $M = [s_1, s_2, s_3, \ldots, s_m]$
End for
For each cloud task, do:
    Define the length of each task in $N = [t_1, t_2, t_3, \ldots, t_n]$
End for
While (t <= iterations_max):
    Determine the survival value for each schedule $SV(x_i)$
        For each $SV(s_i)$ **do**:
            Determine the direction y_i using the equation:
$$y_i = \sum_{j=1}^{p} e^{(SV(x_i))^r - r_{ij}(x_j - x_i)}$$
        End for
        For each Task schedule surviving value i do:
            Determine the movement of the prey schedule using the equation [21]:
$$\text{xschedule}_i = \text{xschedule}_{i+1} + \text{Lmax}(\in 2\left(\frac{y_i}{||y_i||}\right) + \in 3\left(\frac{y_r}{||y_r||}\right))$$
        End for
        For each Task schedule surviving value i do:
            Determine the distance between the positions of prey & predator schedules using the equation:
$$d_1 = ||\text{Scheule}_{\text{predator}} - (x_j + y_r)||$$
$$d_2 = ||\text{Scheule}_{\text{predator}} - (x_j - y_r)||$$
        End for
        For each Task schedule surviving value i do:
            Determine the best prey schedule using the equation:
$$x_{\text{best_prey}} = x_i + \lambda_{\min(\epsilon_4)(\frac{y_i}{||y_i||})}$$
        End for
        $t = t + 1$
End while

3.3. The Proposed Discrete Prey–Predator Method for Cloud Task Scheduling in Detail

This section describes the steps, mapping, and equations for the proposed Discrete Prey–Predator method for cloud task scheduling.

Suppose that

$$N = [t_1, t_2, t_3, \ldots, t_n] \tag{1}$$

is a set of n cloud tasks and

$$M = [s_1, s_2, s_3, \ldots, s_m] \tag{2}$$

is a set of m cloud resource providers.

Suppose that we have 12 tasks $[t_1, t_2, t_3, \ldots, t_{12}]$ with task lengths as stated respectively in Equation (3):

$$\text{Tasks Lengths} = [12, 6, 4, 2, 10, 2, 2, 24, 8, 14, 16, 4] \tag{3}$$

Suppose that we have 6 resources $[s_1, s_2, s_3, s_4, s_5, s_6]$ with resource speeds as stated respectively in Equation (4):

$$\text{Resource Speeds} = [9, 5, 8, 4, 6, 7] \tag{4}$$

The proposed Discrete Prey–Predator algorithm for cloud task scheduling starts the optimization process by initializing of the first random population, $X_0 = [x_1, x_2, x_3, \ldots, x_m]$,

where each x_1 represents a surviving solution. A solution in the population is a cloud task schedule that maps clients' tasks to cloud resource providers. In the mapping process, the dimension index represents the client task index (t_i) and the resource id (s_i).

We start with an intimal population with six random surviving solutions as stated in Equation (5):

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} s_6 & s_2 & s_4 & s_3 & s_1 & s_4 & s_2 & s_6 & s_3 & s_3 & s_6 & s_2 \\ s_3 & s_4 & s_2 & s_1 & s_3 & s_5 & s_3 & s_4 & s_1 & s_1 & s_5 & s_4 \\ s_1 & s_2 & s_3 & s_2 & s_4 & s_6 & s_4 & s_2 & s_3 & s_6 & s_2 & s_4 \\ s_2 & s_1 & s_4 & s_6 & s_2 & s_5 & s_1 & s_3 & s_6 & s_1 & s_1 & s_2 \\ s_3 & s_6 & s_3 & s_2 & s_4 & s_1 & s_5 & s_2 & s_4 & s_4 & s_1 & s_6 \\ s_4 & s_5 & s_2 & s_1 & s_3 & s_6 & s_1 & s_4 & s_2 & s_1 & s_3 & s_5 \end{pmatrix} \quad (5)$$

Each surviving solution in Equation (5) represents a cloud-task-scheduling solution.

The proposed Discrete Prey–Predator algorithm for cloud task scheduling then calculates the fitness of each surviving solution $SV(x_i)$. Equation (6) represents the six surviving solutions and their corresponding fitness values.

$$\begin{pmatrix} SV(x_1) \\ SV(x_2) \\ SV(x_3) \\ SV(x_4) \\ SV(x_5) \\ SV(x_6) \end{pmatrix} = \begin{pmatrix} 15.44 \\ 17.97 \\ 18.72 \\ 15.18 \\ 18.96 \\ 18.60 \end{pmatrix} \quad (6)$$

After calculating the fitness values of each surviving solution $SV(x_i)$, the proposed Discrete Prey–Predator algorithm updates the current solutions by moving according to the equation of the best prey. The algorithm considers each surviving value compared to other surviving values $SV(x_i)$ in Equation (6). Starting from surviving value 1, the best prey for surviving value 1 $SV(x_1)$ is 15.44.

The best prey value for $SV(x_2)$ considering $SV(x_1)$ is 15.88. Using the same calculations, we find that the best prey ($bestPrey_i$) for each surviving value $SV(x_i)$ considering surviving value 1 $SV(x_1)$ is as follows:

$$\begin{pmatrix} bestPrey_1 \\ bestPrey_2 \\ bestPrey_3 \\ bestPrey_4 \\ bestPrey_5 \\ bestPrey_6 \end{pmatrix} = \begin{pmatrix} 15.44 \\ 15.88 \\ 15.44 \\ 16.37 \\ 16.16 \\ 15.50 \end{pmatrix} \quad (7)$$

Surviving value 4 $SV(x_4)$ has the best value, 16.37, so surviving value 1 $SV(x_1)$ moves to surviving value 4 $SV(x_4)$.

For surviving value 2 $SV(x_2)$,

$$\begin{pmatrix} bestPrey_1 \\ bestPrey_2 \\ bestPrey_3 \\ bestPrey_4 \\ bestPrey_5 \\ bestPrey_6 \end{pmatrix} = \begin{pmatrix} 18.33 \\ 17.97 \\ 18.00 \\ 18.35 \\ 18.90 \\ 18.22 \end{pmatrix} \quad (8)$$

Surviving value 5 $SV(x_5)$ has the best value, 18.90, so surviving value 2 $SV(x_2)$ moves to surviving values 5 $SV(x_5)$.

Using the same steps, the Discrete Prey–Predator cloud-task-scheduling algorithm calculates the movement of the remaining four solutions of the initial random population.

The solution with surviving value 5 has maximum fitness and randomly moves in the solution search space.

The new population of the proposed Prey–Predator algorithm after applying the movement equation is described in Equation (9):

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} s_5 & s_1 & s_3 & s_2 & s_1 & s_3 & s_1 & s_5 & s_2 & s_2 & s_5 & s_1 \\ s_2 & s_3 & s_1 & s_1 & s_2 & s_4 & s_2 & s_3 & s_1 & s_1 & s_4 & s_3 \\ s_1 & s_1 & s_2 & s_1 & s_3 & s_5 & s_3 & s_1 & s_2 & s_5 & s_1 & s_3 \\ s_2 & s_1 & s_3 & s_5 & s_1 & s_4 & s_1 & s_2 & s_5 & s_1 & s_1 & s_1 \\ s_3 & s_6 & s_3 & s_2 & s_4 & s_1 & s_5 & s_2 & s_4 & s_4 & s_1 & s_6 \\ s_3 & s_4 & s_1 & s_1 & s_2 & s_5 & s_1 & s_3 & s_1 & s_1 & s_2 & s_4 \end{pmatrix} \quad (9)$$

The proposed Prey–Predator algorithm for cloud task scheduling continues generating new populations for a specific number of iterations or until the surviving fitness values are saturated.

3.4. Benefits of Using the Discrete Prey–Predator Algorithm for Cloud Scheduling

The Discrete Prey–Predator algorithm offers several notable advantages when applied to cloud scheduling problems, setting it apart from its application in more general scheduling contexts, such as those encountered in manufacturing. Cloud environments are characterized by their dynamic and unpredictable nature, marked by fluctuations in resource availability, demand, and network conditions. In contrast to manufacturing scheduling, where resources typically maintain a static profile, the Discrete Prey–Predator algorithm excels in adapting to these ever-shifting scenarios by continuously optimizing task allocation in response to changing conditions. Moreover, cloud environments introduce a unique capability for resource scaling and virtualization, enabling the dynamic allocation of tasks to virtual machines (VMs). This heightened flexibility, which is often absent in manufacturing scheduling characterized by static physical resources, is effectively harnessed by the Discrete Prey–Predator algorithm to optimize task–VM assignments while carefully managing the interplay between workload and resource utilization. Furthermore, cloud scheduling frequently involves multiple objectives, including the minimization of makespan, execution time, and cost. These multi-objective challenges are more pronounced in cloud environments compared with manufacturing scheduling, which typically concentrates on a solitary objective, like minimizing production time. The Discrete Prey–Predator algorithm's intrinsic ability to adeptly handle multi-objective optimization renders it especially well suited for tackling these intricate cloud-scheduling demands. Additionally, the algorithm's adaptability extends to encompassing the presence of heterogeneous resources within cloud data centers, encompassing CPUs, GPUs, and specialized hardware. This resource diversity introduces an additional layer of complexity to scheduling decisions, a challenge that the Discrete Prey–Predator algorithm effectively navigates by considering the unique attributes of each resource type in its scheduling determinations. Finally, cloud environments introduce concerns related to security threats and the imperative for survivability mechanisms, dimensions not commonly encountered in more generalized manufacturing scheduling. Nonetheless, the Discrete Prey–Predator algorithm can be extended to effectively address these security and survivability challenges in the context of cloud scheduling, further enhancing its applicability and versatility for the task at hand. In summation, the Discrete Prey–Predator algorithm's adaptability to the dynamic, multi-objective, and heterogeneous nature of cloud environments, coupled with its potential to accommodate security and survivability considerations, firmly positions it as a robust and adaptable tool for optimizing cloud task scheduling amidst a backdrop of evolving challenges.

4. Performance Evaluations

This section presents the performance evaluation of the proposed Discrete Prey–Predator algorithm for cloud task scheduling. The evaluation is divided into two parts.

The programming of the proposed algorithm using the Java platform is the first part. The second part is developing a cloud simulation environment using the CloudSim 3.0-5.0 tool.

CloudSim is designed to simulate the characteristics and operations of cloud computing environments. While it may not precisely mirror every facet of an authentic cloud system, it offers a closely mimicked and well-regulated setting for research experiments. This capability empowers researchers to investigate and assess cloud-centric algorithms, such as task scheduling, within a precisely controlled context. CloudSim facilitates the repetition of experiments in a uniform setting. This capacity for repeated trials is pivotal in guaranteeing the credibility of outcomes and in showcasing the resilience of one's algorithm proposal. CloudSim grants researchers the ability to manage and fine-tune diverse parameters and circumstances, streamlining the process of isolating particular variables and sharpening their impacts on a system's performance. This careful management of variability serves to strengthen the credibility of CloudSim experimental findings.

4.1. Java Platform Evaluation

This part of the study investigates a simple scenario based on simulating the proposed Discrete Prey–Predator method using the Java platform. The Java platform scenario consists of two cases: In the first one, the number of tasks is ten, and the number of resources is five. In the second case, the number of tasks is twenty, and the number of resources is ten. In these two scenarios, each iteration's survival value is determined by selecting the population's maximum survival value in that iteration.

4.2. The First Scenario

As mentioned in the previous section, the number of tasks in this scenario is ten, and the number of resources is five. The simulation results are presented in Table 1 and visualized in Figure 2. The survival value of the proposed Discrete Prey–Predator cloud-task-scheduling mechanism represents the execution time.

Table 1. The survival values of ten iterations in the first scenario.

Iteration	1	2	3	4	5	6	7	8	9	10
Survival value	0.05	0.05	0.06	0.07	0.07	0.07	0.07	0.07	0.07	0.07

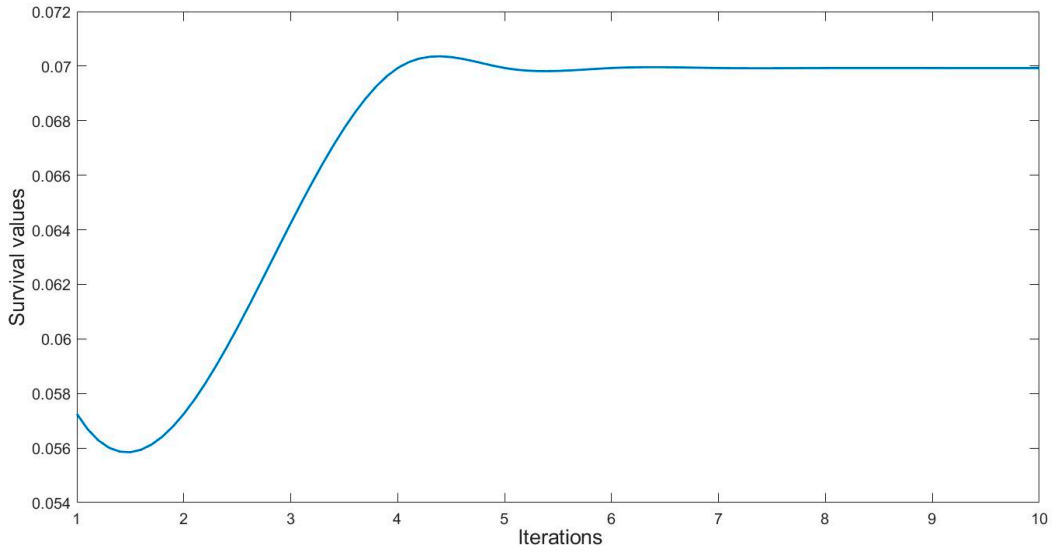

Figure 2. The survival values of ten iterations in the first scenario.

As detailed in Table 1 and shown in Figure 2, the resulting Java platform survival value is 0.057 in the first iteration and increases to 0.064 in the second iteration. The survival

value increases during the simulation iterations and settles at 0.070 in the last iteration of the simulation.

4.3. The Second Scenario

In the second scenario of the Java-based simulation, the number of tasks is twenty, and the number of resources is ten. The results of the simulation survival value are presented in Table 2 and Figure 3.

Table 2. The survival values of ten iterations in the second scenario.

Iteration	1	2	3	4	5	6	7	8	9	10
Survival value	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.0285	0.07

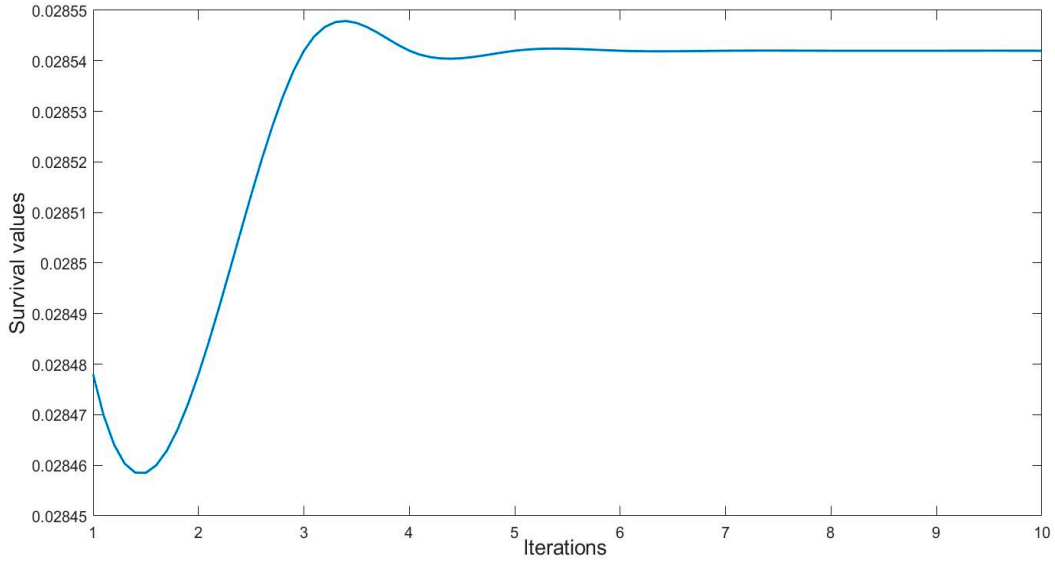

Figure 3. The survival values of ten iterations in the second scenario.

Table 2 and Figure 3 describe the resulting survival values of the proposed Discrete Prey–Predator cloud-task-scheduling mechanism. The survival value starts from 0.028478 in the first iteration and gradually increases until it settles at 0.028542.

4.4. The CloudSim Simulation Results

The second phase of the proposed Discrete Prey–Predator cloud-task-scheduling mechanism is based on the simulation of cloud environments using the CloudSim simulator [43]. Four simulation scenarios are studied in this phase, as presented in the following subsections.

4.5. The First CloudSim Simulation Scenario

The first CloudSim scenario consists of 70 tasks and 30 resources. The execution time results are presented in Table 3 and Figure 4.

Table 3. The execution times of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the first scenario.

Iteration	1	2	3	4	5	6	7	8	9	10
Survival value	261	248	238	227	224	191	191	191	191	165

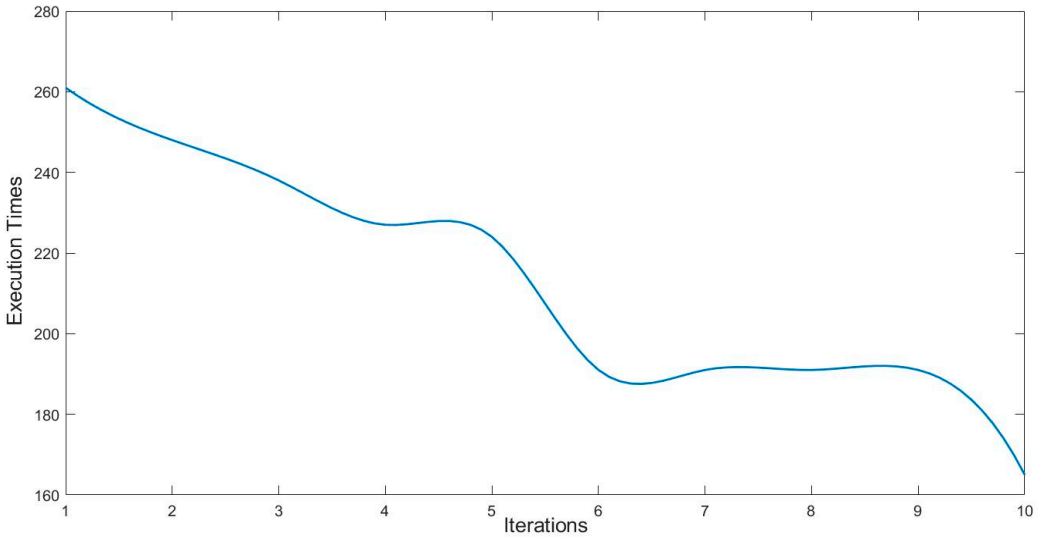

Figure 4. The execution times of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the first scenario.

As shown in Figure 4 and Table 3, the execution time of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the first scenario is 261 in the first iteration of the optimization process. From the second iteration, the execution time of proposed Prey–Predator mechanism for cloud task scheduling decreases gradually until it reaches 165 in the last iteration.

4.6. The Second CloudSim Simulation Scenario

The second CloudSim scenario consists of 150 tasks and 80 resources. The execution time results are presented in Table 4 and Figure 5.

Table 4. The execution times of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the second scenario.

Iteration	1	2	3	4	5	6	7	8	9	10
Survival value	596	582	474	455	455	455	455	437	437	423

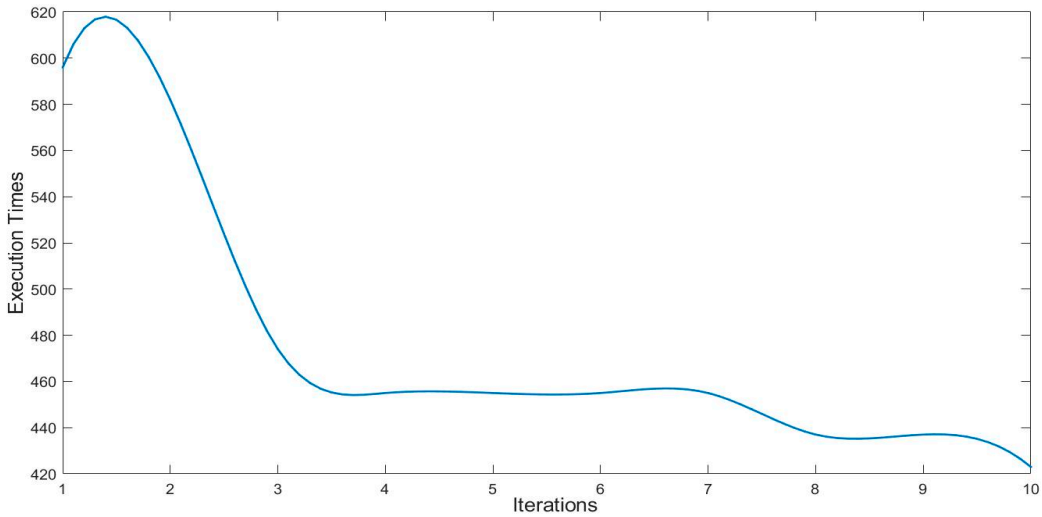

Figure 5. The execution times of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the second scenario.

As shown in Figure 5 and Table 4, the execution time of the proposed Discrete Prey–Predator mechanism for cloud task scheduling in the first scenario is 596 in the first iteration of the optimization process.

In the second iteration, the execution time of the proposed Discrete Prey–Predator mechanism for cloud task scheduling decreases to 582 and continues decreasing steadily until it reaches 423 in the last iteration.

4.7. The Third CloudSim Simulation Scenario

In this scenario, a simulation comparison is conducted between the proposed Discrete Prey–Predator mechanism for cloud task scheduling and the firefly algorithm for cloud task scheduling [7,32,44]. The execution times of the simulation of the proposed Discrete Prey–Predator optimization, a random solution, and firefly algorithm are presented in Table 5 and Figure 6. This simulation considers 90 tasks and 50 resources.

Table 5. A comparison between the Discrete Prey–Predator algorithm, a random solution, and firefly algorithm.

Scheduling Mechanism	Execution Time
Random solution	386
Firefly algorithm	247
Prey–Predator	238

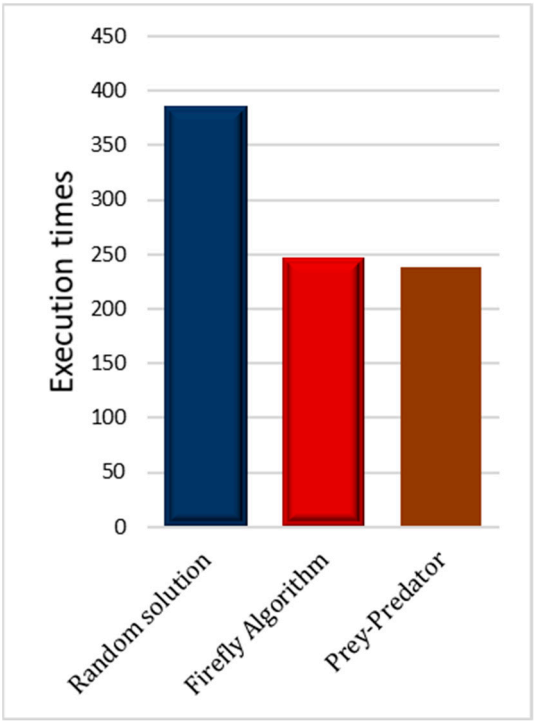

Figure 6. A comparison between Discrete Prey–Predator algorithm, a random solution, and firefly Algorithm.

Table 5 and Figure 6 show the comparison of the execution times of the proposed Discrete Prey–Predator algorithm, a random solution, and firefly algorithm for cloud task scheduling. The random solution has the longest execution time, 386 s, and the proposed Prey–Predator algorithm has the shortest execution time, 238 s. The firefly algorithm performs better than the random solution, with 274 s execution time.

4.8. The Fourth CloudSim Simulation Scenario

The fourth scenario is a comparative experiment between the proposed Discrete Prey–Predator algorithm for cloud task scheduling and the firefly algorithm. Several experiments are conducted with a different number of tasks and resources. The results of the fourth scenario are presented in Table 6 and Figure 7.

Table 6. The execution times of the proposed Discrete Prey–Predator algorithm and the firefly algorithm for cloud task scheduling.

Experiment No.	1	2	3	4	5
Number of tasks	20	60	100	120	150
Number of resources	10	30	50	80	100
Firefly algorithm	41	262	394	376	500
Prey–Predator algorithm	28	203	339	353	430

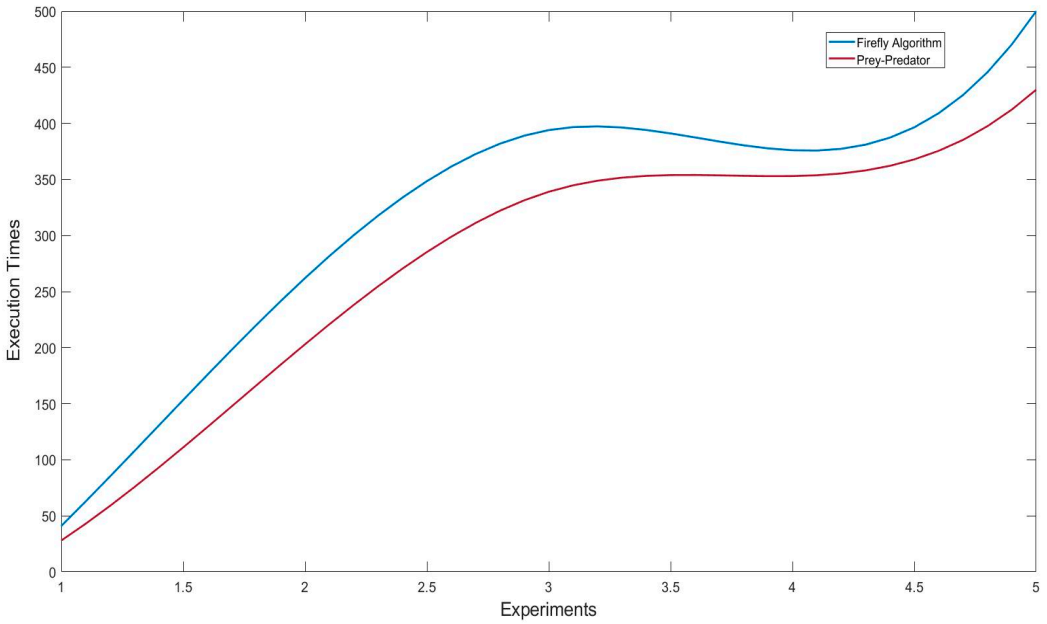

Figure 7. The comparison of simulation results in the fourth scenario.

As shown in Table 6 and Figure 7, the execution time of the Discrete Prey–Predator algorithm is 28 s in the first experiment, which is 41 s less than the execution time of the firefly algorithm. However, in the subsequent experiments and when the number of resources and tasks is increased, the proposed Discrete Prey–Predator algorithm has shorter execution times than the firefly algorithm. Furthermore, as shown in Table 6, the average execution time of the proposed Discrete Prey–Predator algorithm is 270 s less than the average execution time of the firefly algorithm, which is 315 s. Hence, the experiment results reveal that the proposed Discrete Prey–Predator algorithm for cloud task scheduling outperforms the firefly algorithm in the case of execution time.

4.9. Limitation of Performance Evaluation Works

One limitation of the performance evaluation is that it relies on simulations within the CloudSim environment. While CloudSim is a valuable tool for simulating computational cloud scenarios, it is not an accurate representation of real-world cloud environments.

In order to overcome these limitations and substantiate the applicability of our research in a real computation cloud platform, we propose pilot deployment that involves shifting from simulation to actual implementation within a genuine cloud environment. This

approach would facilitate the direct observation of the algorithm's performance under real-world conditions.

4.10. Results Discussion

In the first scenario studied on the Java platform, we examined the performance of the Discrete Prey–Predator algorithm with a modest task–resource configuration of ten tasks and five resources. The survival values representing execution times displayed an initial value of 0.057 in the first iteration, progressively increasing to 0.070 by the last iteration. This trend signifies an iterative improvement in execution time as the algorithm converged towards optimal task-scheduling solutions. In the second Java-based simulation scenario, we escalated the complexity by increasing the task–resource configuration to twenty tasks and ten resources. Similarly to the first scenario, the survival values began from 0.028478 in the first iteration and gradually improved, ultimately stabilizing at 0.028542. These results reinforce the algorithm's ability to adapt to different task–resource scenarios and continuously enhance execution times.

Moving to the CloudSim simulation phase, the first scenario involved 70 tasks and 30 resources. The execution times revealed a consistent pattern, with initial execution times of 261 in the first iteration decreasing steadily to 165 by the tenth iteration. This demonstrates that the Discrete Prey–Predator algorithm effectively optimized task scheduling, achieving reduced execution times over successive iterations. In the second CloudSim scenario, which featured a more substantial workload of 150 tasks and 80 resources, the algorithm exhibited similar behavior. Execution times started from 596 in the first iteration and progressively decreased to 423 in the last iteration. This scenario reinforced the scalability and efficiency of the algorithm, as the algorithm effectively handled a larger task–resource configuration while still optimizing execution time. In the second CloudSim scenario, which featured a more substantial workload of 150 tasks and 80 resources, the algorithm exhibited similar behavior. Execution times started from 596 in the first iteration and progressively decreased to 423 in the last iteration. This scenario reinforced the scalability and efficiency of the algorithm, as the algorithm effectively handled a larger task–resource configuration while still optimizing execution time. In the third CloudSim scenario, we compared the Discrete Prey–Predator algorithm to the firefly algorithm for cloud task scheduling. The results clearly indicated that the proposed algorithm outperformed both the random solution and the firefly algorithm in terms of execution time. This comparative analysis validated the superiority of the Discrete Prey–Predator algorithm in achieving efficient task scheduling within a cloud environment. In the fourth scenario, we conducted a comprehensive comparative experiment between the Discrete Prey–Predator algorithm and the firefly algorithm across various task–resource configurations. The results demonstrated that the Discrete Prey–Predator algorithm consistently outperformed the firefly algorithm as the task and resource numbers increased. The average execution times further confirmed the algorithm's superiority, with an average execution time of 270 for the Discrete Prey–Predator algorithm compared with 315 for the firefly algorithm. These findings illustrate the algorithm's scalability and effectiveness in optimizing execution time even in more demanding cloud environments.

5. Conclusions

This paper proposes a new task-scheduling mechanism based on the Discrete Prey–Predator algorithm to minimize the execution time of the task-scheduling process in cloud computing. The study started by mapping cloud task scheduling with the Discrete Prey–Predator algorithm. The proposed mechanism generates an initial random population of solutions of schedules. Each solution has a fitness value representing the Discrete Prey–Predator algorithm's survival value. The survival value corresponds to the execution times of the population schedules. The fitness function aims to minimize the execution times of the schedules. To evaluate the proposed Discrete Prey–Predator, the study conducted a simulation using the CloudSim tool. Several scenarios were implemented in the evaluation

process. The simulation results found that the proposed Discrete Prey–Predator cloud-task-scheduling mechanism decreases the execution times gradually when increasing the number of iterations. The execution time reduction process continues for several iterations; then, the execution time is saturated and remains constant. Furthermore, the study conducted a simulation to compare the proposed Discrete Prey–Predator mechanism with the firefly algorithm. The simulation results revealed that the Discrete Prey–Predator mechanism outperformed the firefly algorithm for cloud job scheduling.

Author Contributions: Conceptualization, D.A.A.; Methodology, D.A.A. and A.Y.; Original draft preparation, D.A.A.; Review and editing, A.Y. and A.A.; Visualization, D.A.A. and A.Y.; Supervision, A.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deputy for Research and Innovation, Ministry of Education, Kingdom of Saudi Arabia, through a grant (NU/IFC/2/SERC/-/18) under the institutional funding committee at Najran University, Kingdom of Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Authors would like to acknowledge the support of Deputy for Research and Innovation, Ministry of Education, Kingdom of Saudi Arabia, for this research through a grant (NU/IFC/2/SERC/-/18) under the institutional funding committee at Najran University, Kingdom of Saudi Arabia.

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

References

1. Qian, L.; Luo, Z.; Du, Y.; Guo, L. Cloud computing: An overview. In Proceedings of the Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, 1–4 December 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 626–631.
2. Dillon, T.; Wu, C.; Chang, E. Cloud computing: Issues and challenges. In Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, Australia, 20–23 April 2010; pp. 27–33.
3. Kim, W. Cloud computing: Today and tomorrow. *J. Object Technol.* **2009**, *8*, 65–72. [CrossRef]
4. Malik, M.I.; Wani, S.H.; Rashid, A. Cloud Computing-Technologies. *Int. J. Adv. Res. Comput. Sci.* **2018**, *9*. [CrossRef]
5. Zain, A.M.; Yousif, A. Chemical reaction optimization (CRO) for cloud job scheduling. *SN Appl. Sci.* **2020**, *2*, 53. [CrossRef]
6. Suliman, Y.M.; Yousif, A.; Bashir, M.B. Shark Smell Optimization (SSO) Algorithm for Cloud Jobs Scheduling. In Proceedings of the International Conference on Computing, Bali, Indonesia, 19–22 April 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 71–80.
7. Esa, D.; Yousif, A. Scheduling jobs on cloud computing using firefly algorithm. *Int. J. Grid Distrib. Comput.* **2016**, *9*, 149–158. [CrossRef]
8. Esa, D.I.; Yousif, A. Glowworm Swarm Optimization (GSO) for Cloud Jobs Scheduling. *Int. J. Adv. Sci. Technol.* **2016**, *96*, 71–82. [CrossRef]
9. Singh, R.M.; Awasthi, L.K.; Sikka, G. Towards metaheuristic scheduling techniques in cloud and fog: An extensive taxonomic review. *ACM Comput. Surv.* **2022**, *55*, 1–43. [CrossRef]
10. Rahimikhanghah, A.; Tajkey, M.; Rezazadeh, B.; Rahmani, A.M. Resource scheduling methods in cloud and fog computing environments: A systematic literature review. *Clust. Comput.* **2022**, *25*, 911–945. [CrossRef]
11. Iftikhar, S.; Ahmad, M.M.M.; Tuli, S.; Chowdhury, D.; Xu, M.; Gill, S.S.; Uhlig, S. HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. *Internet Things* **2023**, *21*, 100667. [CrossRef]
12. Attiya, I.; Abd Elaziz, M.; Abualigah, L.; Nguyen, T.N.; Abd El-Latif, A.A. An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud. *IEEE Trans. Ind. Inform.* **2022**, *18*, 6264–6272. [CrossRef]
13. Nabi, S.; Ahmad, M.; Ibrahim, M.; Hamam, H. AdPSO: Adaptive PSO-based task scheduling approach for cloud computing. *Sensors* **2022**, *22*, 920. [CrossRef]
14. Xing, Y.; Zhan, Y. Virtualization and cloud computing. In *Future Wireless Networks and Information Systems*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 1, pp. 305–312.
15. Masdari, M.; Zangakani, M. Efficient task and workflow scheduling in inter-cloud environments: Challenges and opportunities. *J. Supercomput.* **2020**, *76*, 499–535. [CrossRef]
16. Aloboud, E.; Kurdi, H. Cuckoo-inspired Job Scheduling Algorithm for Cloud Computing. *Procedia Comput. Sci.* **2019**, *151*, 1078–1083. [CrossRef]

17. Zhang, Z.; Zhao, M.; Wang, H.; Cui, Z.; Zhang, W. An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty. *Inf. Sci.* **2022**, *583*, 56–72. [CrossRef]
18. Krishna, A.V.; Ramasubbareddy, S.; Govinda, K. Task Scheduling Based on Hybrid Algorithm for Cloud Computing. In *International Conference on Intelligent Computing and Smart Communication 2019*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 415–421.
19. Tilahun, S.L.; Ong, H.C. Prey-predator algorithm: A new metaheuristic algorithm for optimization problems. *Int. J. Inf. Technol. Decis. Mak.* **2015**, *14*, 1331–1352. [CrossRef]
20. Zhang, H.; Yuan, M.; Liang, Y.; Liao, Q. A novel particle swarm optimization based on prey–predator relationship. *Appl. Soft Comput.* **2018**, *68*, 202–218. [CrossRef]
21. Li, Y.; Li, Y.; Liu, Y.; Cheng, H. Stability analysis and control optimization of a prey-predator model with linear feedback control. *Discret. Dyn. Nat. Soc.* **2018**, *2018*, 4945728. [CrossRef]
22. Yeboah, T.; Odabi, I.; Hiran, K.K. An integration of round robin with shortest job first algorithm for cloud computing environment. In Proceedings of the International Conference on Management, Communication and Technology, Prague, Czech Republic, 20–22 April 2015; pp. 1–5.
23. Mohammed, A.; Yousif, A. Enhanced Job Scheduling Algorithm for Cloud Computing Using Shortest Remaining Job First (SRJF). *Int. J. Comput. Sci. Manag. Stud.* **2015**, *15*, 65–68.
24. Elzeki, O.; Reshad, M.; Elsoud, M.A. Improved max-min algorithm in cloud computing. *Int. J. Comput. Appl.* **2012**, *50*, 22–27.
25. Alworafi, M.A.; Dhari, A.; Al-Hashmi, A.A.; Darem, A.B. An improved SJF scheduling algorithm in cloud computing environment. In Proceedings of the 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), Mysuru, India, 9–10 December 2016; pp. 208–212.
26. Chen, H.; Guo, W. Real-time task scheduling algorithm for cloud computing based on particle swarm optimization. In Proceedings of the Cloud Computing and Big Data: Second International Conference, CloudCom-Asia 2015, Huangshan, China, 17–19 June 2015; Revised Selected Papers 2; Springer: Berlin/Heidelberg, Germany, 2015; pp. 141–152.
27. Tawfeek, M.A.; El-Sisi, A.; Keshk, A.E.; Torkey, F.A. Cloud task scheduling based on ant colony optimization. In Proceedings of the 2013 8th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egypt, 26–28 November 2013; pp. 64–69.
28. Wang, Y.; Zuo, X.; Wu, Z.; Wang, H.; Zhao, X. Variable neighborhood search based multiobjective ACO-list scheduling for cloud workflows. *J. Supercomput.* **2022**, *78*, 18856–18886. [CrossRef]
29. Alsaidy, S.A.; Abbood, A.D.; Sahib, M.A. Heuristic initialization of PSO task scheduling algorithm in cloud computing. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 2370–2382. [CrossRef]
30. Nanjappan, M.; Albert, P. Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e5517. [CrossRef]
31. Velliangiri, S.; Karthikeyan, P.; Xavier, V.A.; Baswaraj, D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Eng. J.* **2021**, *12*, 631–639. [CrossRef]
32. Bacanin, N.; Zivkovic, M.; Bezdan, T.; Venkatachalam, K.; Abouhawwash, M. Modified firefly algorithm for workflow scheduling in cloud-edge environment. *Neural Comput. Appl.* **2022**, *34*, 9043–9068. [CrossRef]
33. Hassan, M.; Yousif, A. Cloud job scheduling with ions motion optimization algorithm. *Eng. Technol. Appl. Sci. Res* **2020**, *10*, 5459–5465. [CrossRef]
34. Zhang, A.-N.; Chu, S.-C.; Song, P.-C.; Wang, H.; Pan, J.-S. Task scheduling in cloud computing environment using advanced phasmatodea population evolution algorithms. *Electronics* **2022**, *11*, 1451. [CrossRef]
35. Khaleel, M.I. Efficient job scheduling paradigm based on hybrid sparrow search algorithm and differential evolution optimization for heterogeneous cloud computing platforms. *Internet Things* **2023**, *22*, 100697. [CrossRef]
36. Abd Elaziz, M.; Xiong, S.; Jayasena, K.; Li, L. Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowl. Based Syst.* **2019**, *169*, 39–52. [CrossRef]
37. Abdel-Basset, M.; Mohamed, R.; Abd Elkhalik, W.; Sharawi, M.; Sallam, K.M. Task Scheduling Approach in Cloud Computing Environment Using Hybrid Differential Evolution. *Mathematics* **2022**, *10*, 4049. [CrossRef]
38. Saif, F.A.; Latip, R.; Derahman, M.; Alwan, A.A. Hybrid meta-heuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment. In *Proceedings of the Future Technologies Conference*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 16–43.
39. Akinwale, A.; Olajubu, E.A.; Aderounmu, G.A. A Prey-Predator Defence Mechanism For Ad Hoc On-Demand Distance Vector Routing Protocol. *CIT J. Comput. Inf. Technol.* **2023**, *30*, 67–83. [CrossRef]
40. Khan, N.A.; Sulaiman, M.; Seidu, J.; Alshammari, F.S. Mathematical Analysis of the Prey-Predator System with Immigrant Prey Using the Soft Computing Technique. *Discret. Dyn. Nat. Soc.* **2022**, *2022*, 1241761. [CrossRef]
41. Lu, Y.; Li, M. Unconditionally convergent and superconvergent FEMs for nonlinear coupled time-fractional prey–predator problem. *Comput. Appl. Math.* **2023**, *42*, 111. [CrossRef]
42. Tilahun, S.L. Prey-predator algorithm for discrete problems: A case for examination timetabling problem. *Turk. J. Electr. Eng. Comput. Sci.* **2019**, *27*, 950–960. [CrossRef]

43. Buyya, R.; Ranjan, R.; Calheiros, R.N. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In Proceedings of the High Performance Computing & Simulation, Leipzig, Germany, 21–24 June 2009; pp. 1–11.
44. Mangalampalli, S.; Karri, G.R.; Elngar, A.A. An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization. *Sensors* **2023**, *23*, 1384. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.