Task Scheduling Using Enhanced Sunflower Optimization in Cloud Environment

R. Ilakiya¹, B. Parkavi², R. Anushiya³, Dr. M. Suresh Kumar⁴

^{1, 2, 3} Dept of Information Technology

⁴Professor, Dept of Information Technology ^{1, 2, 3, 4} Sri Rama Krishna Institute of Technology, Coimbatore, Tamil Nadu, India

Abstract- The abstract commences by introducing the importance of task scheduling and the obstacles associated with it. It emphasizes the necessity for effective optimization methods to tackle the complexities arising from elements such as task interdependencies, resource limitations, and everchanging environments. Particle Swarm Optimization (PSO) has emerged as a promising algorithm for addressing the intricacies of task scheduling in cloud computing settings. This abstract provides an overview of PSO's application in cloud task scheduling, underscoring its capability to enhance resource allocation efficiency, adjust to dynamic workloads, and boost overall system performance. By emulating the social interactions of particles within a swarm, PSO dynamically navigates through solution spaces, maintaining a balance between exploration and exploitation to move towards optimal task distributions. This abstract explores the process of PSO in cloud task scheduling, highlighting its adaptability to shifting conditions and its flexibility in optimizing various objectives. Additionally, it underscores the importance of PSO in enhancing resource usage, reducing make-span, and handling the issues related to scalability and effectiveness in cloud computing. Through its iterative refinement procedure, *PSO offers a reliable and scalable solution for task scheduling* in cloud environments, leading to improved performance and service quality for cloud-based applications and services.

Keywords- Task scheduling, completion time, task scheduler, resource information server, virtual machine, meta-heuristic scheduling, enhanced sunflower optimization.

I. INTRODUCTION

Cloud computing is a recent method that offers easy access to shared computer resources like networks, servers, storage, software, and services over the network when needed. It minimizes management efforts and communication with providers while allowing quick provisioning and release. Currently, cloud services are mainly utilized in business settings, focusing on providing IT infrastructure on demand. Cloud computing has the potential to significantly impact various sectors, including search engines, social networks, ebusiness, virtual worlds, and innovations. Nevertheless, it is still in its early stages and requires further exploration. Issues such as the absence of a common programming paradigm, open interfaces, adequate service level agreements, and application portability limit the effectiveness of cloud computing solutions. Neglecting these challenges may lead to consumer reliance on closed, proprietary systems, hindering the mobility of applications that are being transitioned to the cloud. Furthermore, users often provide commercial providers access to their data and applications without a clear quality of service agreement. The primary objective is to establish a framework for evaluating the benefits of cloud computing over traditional IT infrastructure. The emergence of cloud computing providers and the question of portability are driving factors in the development of this field. Many businesses have already integrated cloud computing services into their IT structures.

- The Enhanced Sunflower Optimization (ESFO) algorithm has exhibited significant enhancements in energy consumption and make-span measures compared to other algorithms in task scheduling benchmarks. The ESFO algorithm aims to enhance both exploitation and exploration in solving task scheduling challenges within a cloud environment.
- Evaluation of the ESFO algorithm using various task scheduling benchmarks has showcased its superiority over comparable algorithms, particularly in terms of energy consumption and make-span measures.

II. RELATED WORK

Researchers have made significant progress in recent years in the field of task scheduling within a cloud environment, utilizing various methods and algorithms to improve accuracy resilience.

The research paper by G. Natesan, A. Chokkalingam et al.[1] task scheduling's main goals are to minimize the schedule's objective and schedule the task based on available resources. The grey wolf algorithm was stated in this study to improve system performance and reduce scheduling problems. This method's primary objective is to reduce energy usage as well as make span. The simulation's conclusion indicates that, in contrast to alternative algorithms currently in use, the suggested Mean GWO algorithms produces relatively ample results.

The research led by N. Mansouri, B.M.H. Zade, M.M. Javidi et al.[2] an appropriate arranging tasks strategy is cloud required in settings to offer economical implementations. In order to enhance cloud throughput and load balancing, this paper presents the modified particle swarm optimization (PSO) approach and a fuzzy system as the foundation of the FMPSO hybrid task scheduling algorithm. The FMPSO strategy first considers a roulette wheel selection technique along with, Four modified velocity updating methods have been developed to improve the global search capability. Next, in order to get around some of PSO's shortcomings, like local optima, it makes use of crossover and mutation operators. Lastly, the system of fuzzy inference is employed.

In their study, J. Meshkati, F. Safi-Esfahani [3] a massive electrical Cloud data center energy consumption raises their operating expenses. This demonstrates the value of spending money on energy-saving measures. One way to lower energy consumption is to use meta-heuristic algorithms to move virtual machines around dynamically the right reality nodes. To discover superior options in a search area, meta-heuristic algorithms should discover a middle ground between exploitation and exploration. Exploration is the process of searching a larger area for a solution, whereas exploitation is the process of creating new solutions from pre- existing ones. In biological, The One type of meta-heuristic algorithm is the artificial bee colony (ABC) algorithm. Optimization a signoriented method, its exploration ability.

J. Yang, et al. [4] compared to a traditional distributed system, the task scheduling is more intricate. Based on a cloud computing analysis task scheduling system, created a simplified form, cloud computing in related literature. In contrast to earlier research on task scheduling algorithms for cloud computing as a mathematical tool, the simplified model in this work has as its foundation game theory. The algorithm for task scheduling that takes into take into consideration the stability of the balanced task is suggested, and it is based on the theory of games. The model of task scheduling proposed here is for nodes that compute and is built upon the algorithm for balanced scheduling.

X. Wei, et al.'s research [5] presents a method of task scheduling optimization employing ant colony optimization algorithm in cloud computing to address problems with inconsistent load, sluggish convergence speed, and low virtual machine resource utilization present in earlier methods for optimizing task scheduling. First, An enhanced An algorithm for ant colonies scheduling model to be suggested for preventing The enhancement approach avoiding getting caught up in local optimization. In cloud computing, task scheduling is the foundation for the task scheduling satisfaction function is then created By integrating the three goals of the least amount the length of waiting, the level of resource load balance, and the task completion cost in order find the ideal task scheduling conclusion.

III. METHODOLOGY

The Enhanced Sunflower Optimization algorithm is a sophisticated meta-heuristic method that draws inspiration from the innate behavior of sunflowers. This approach proves highly efficient in tackling optimization challenges across diverse fields.

Problem Formulation

Clearly outlining the task scheduling issue involves identifying the tasks, resources, constraints, and goals for optimization. It is essential to determine if the problem is static or dynamic and if it requires single or multiple criteria optimization.

Encoding

Develop a suitable encoding scheme to represent the solution space. This could involve representing schedules as permutations, binary strings, or other suitable representations depending on the problem characteristics.

Objective Function

The objective function(s) to be optimized, a common objectives in task scheduling include minimizing make-span, minimizing total completion time, maximizing resource utilization, or balancing workload distribution.

Initialization

Initialize the population of sunflowers representing candidate solutions. Ensure that the initial solutions adhere to problem constraints and are diverse to promote exploration.

Enhanced Sunflower Optimization

• Germination Phase: Update the positions of sunflowers based on the traditional sunflower

optimization rules, considering factors such as attraction towards the sun and avoidance of collision with other sunflowers.

- **Pollination Phase:** Enhance the pollination mechanism by introducing probabilistic pollination strategies, such as Levy flight or Gaussian distributions, to explore the solution space more effectively.
- Competition Phase: Incorporate competition mechanisms to encourage diversity and prevent premature convergence. This could involve introducing crowding or tournament selection mechanisms.
- Adaptation Mechanisms: Implement adaptive mechanisms to dynamically adjust algorithm parameters or strategies based on the problem landscape. This could involve adaptive step sizes, mutation rates, or population sizes.
- Local Search: Optionally incorporate local search techniques to exploit promising regions of the solution space. This could involve hill-climbing, simulated annealing, or genetic operators.
- **Termination Criteria:** Define stopping criteria for the optimization process, such as a maximum number of iterations, reaching a certain fitness threshold, or stagnation in the search process.

Solution Extraction

Obtain the optimal solution(s) achieved after the completion of the optimization process. Assess the quality of the solution(s) by evaluating them based on the specified objective function(s).

Performance Evaluation

The effectiveness of the enhanced sunflower optimization algorithm can be assessed by comparing the solutions it produces with those of other metaheuristic algorithms or exact methods. Assess metrics such as solution quality, convergence speed, and scalability.

Parameter Tuning and Sensitivity Analysis

Conducting experiments to adjust parameters is essential in optimizing algorithm settings for a particular task scheduling issue. Sensitivity analysis should be conducted to grasp how alterations in parameters affect the overall performance of the algorithm.

Validation and Testing

Validate the algorithm on benchmark instances and real-world datasets to assess its applicability and effectiveness in practical scenarios.

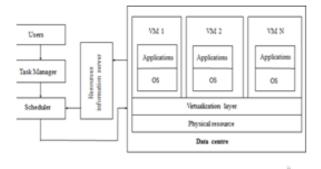
Documentation and Dissemination

Document the methodology, implementation details, experimental results, and insights gained from applying enhanced sunflower optimization to task scheduling. Share findings through research papers, presentations, or openaccess repositories to contribute to the research community.

Feedback

Continuously refine and enhance the algorithm based on feedback, emerging techniques, and new insights gained from further research and experimentation.

Architecture diagram



IV. CURRENT SYSTEM

Numerous scholars have put forth solutions to address the scheduling and resource allocation issue. Tsai put forth an improved differential evolution algorithm-based multi-object approach. The current approach offers a cloud computing time and cost model. An algorithm for load balancing and scheduling that ignores job sizes was proposed by Magukuri. When processing requests, the writers took the server's refresh times into account.

V. PROPOSED SYSTEM

Introducing the ESFO algorithms to resolve the cloud environment's task scheduling problem. To strengthen the exploitation and exploration capacities in ESFO, a new pollination strategy will be implemented. Assessing the suggested ESFO algorithm against several organizing tasks benchmarks to be able to find its advantages and disadvantages. The most crucial step in locating an optimization problem's global optimum is careful solution space exploration. The ESFO algorithm is contrasted with other algorithms, such as Round Robin Optimization and First Come First Serve. An improved version of SFO, known as the ESFO algorithm, is to be established to deal with this issue

VI. EXPERIMENTAL SETUP

I. First Come First Serve (FCFS) Scheduling

First come, first served implies that we should handle the job that was received first, regardless of other properties. We can use our real- time scenario to map this particular situation. The person who joined the line first will receive the ticket when we are waiting in line for movie tickets. Only the second person will be severed. The same approach will be used for scheduling as well. In this context, the issue of work scheduling is being discussed. One CPU can be loaded with multiple jobs, as is common knowledge. Scheduling will be handled by the scheduler. If the scheduler employs the FCFS strategy, the job will be scheduled on the processor to be processed based on which process arrived first. Once a processor has been assigned to a task, it cannot be stopped until the task is finished. We refer to this as non-preemptive scheduling. Pre- emptive scheduling is what we do if we are able to stop. Algorithms with optimal scheduling will reduce average waiting times and turnaround times.

II. Round Robin (RR) Scheduling

One of the most commonly utilized algorithms is undoubtedly the round robin (RR) scheduling algorithm. Each process in this algorithm is allocated a fixed time quantum. Nonetheless, it encounters certain issues, most of which are associated with the time quantum's size. Process response and waiting times tend to increase as the time quantum grows. Conversely, when the time quantum is too small, the CPU is burdened with greater context switches, leading to heightened CPU overhead. This delves into optimization methods for the Round Robin algorithm. Rather than employing a static time quantum, several algorithms have suggested using a dynamic one. Processes in the ready queue have their time quantum determined based on factors like mean, median, and dispersion in relation to their remaining burst time. Additionally, a strategy based on multiple time quanta is recommended. Ultimately, the implementation and results have proven that these algorithms effectively address the issues associated with the traditional Round Robin algorithm, enhancing wait times, response times, and turnaround times.

III. Shortest Job First (SJF)

Scheduling the processes in FCFS scheduling based on the time of their arrival. On the other hand, the SJF scheduling algorithm arranges the processes based on their peak time. The process that has the shortest burst time on the list of processes that are available in the ready queue will be scheduled first in SJF scheduling. However, this algorithm is very hard to implement in the system because it is very hard to predict the burst time required for a process.

IV. Particle Swarm Optimization (PSO)

By iteratively updating the positions and velocities of particles based on their own and neighbours experiences, PSO explore the solution space efficiently and can converge to good solutions for task scheduling problems, PSO's ability to balance exploration(searching the solution space broadly) and exploitation(focusing on promising regions) makes it effective for optimization tasks like task scheduling.

VII. SOFTWARE ENVIRONMENT

CLOUD-SIM

A simulation library called CloudSim provides basic cases for managing different system components like scheduling and provisioning. It also describes users, virtual machines, data centers, and computational resources. It is an open-source, free framework for duplicating the infrastructure and services in cloud computing. Well, it is fully written in Java and is developed by the Clouds Lab group. In order to test a hypothesis before developing software and replicate tests and outcomes, it is utilized to simulate and model an environment for cloud computing.

No money is being invested: There's not setup or upkeep costs when using a simulation tool such as CloudSim.

Easy to use and scalable: can modify specifications by altering a few lines of code, adding or removing resources.

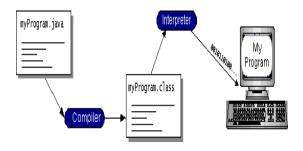
It is possible to assess risks earlier: The utilization of actual test environments within cloud computing confines the ability to experiment to the size of the test environment, posing significant challenges when attempting to replicate results. Through simulation, you can evaluate your product against various test scenarios, addressing issues without any limitations before proceeding to the final deployment.

JDK

The Java programming language stands out for its ability to be compiled and interpreted. Initially, a program

translated into Java byte codes using a compiler, creating codes understood by the Java platform's interpreter. The interpreter parses and executes each Java byte code command on the computer. While interpretation occurs every time the program is run, compilation is a one-time process.

All Java interpreters, whether in Web browsers for applets or development tools, essentially function as the Java Virtual Machine. Java byte codes enable a "write once, run anywhere" capability, allowing programs to be compiled on any platform with a Java compiler into byte codes that can then run on any Java virtual machine.





Java Program

I. First Come First Serve (FCFS)

Output of Proposed System

Is a simple scheduling algorithm that schedules tasks based on their arrival times. In context of task scheduling, accuracy is not typically a metric used to evaluate FCFS, as FCFS does not prioritize tasks based on any criteria other than their arrival order.

	02	SDOCESS.	03	0.6	3051.06	3052.06	6903.92
	0.8	SUCCESS	02	0.5	2490.23	2541.03	4975.26
2	04	SUCCESS	0.3	0.4	1249.03	6903.92	0232.55
	0.6	SUCCESS.	03	0.6	1007.37	4979.26	6066.64
2	30	SUCCESS	02	0.4	714.32	00.2	714.52
	07	STOCKSE	03	04	1256.97	0232.95	5509.02
	OE	success	0.8	0.6	1510.66	3609.02	\$1100.47
	0.8	SUCCESS	02	0.4	2109.51	714.52	2024.03
	10	SUCCESS	02	0.3	776.73	00.2	776.93
	11	SOCCESS	02	0.4	1055.40	2026.03	4401.8
	12	SUCCESS	03	0.6	1551.42	11100.47	12451.09
	1.3	SUCCESS	02	02	1200.41	774.93	1505.54
	14	SUCCESS	02	0.6	\$50.3	6066.64	7424.94
	1.6	SUCCESS	0.3	0.6	3146.9	12651.05	16790.79
	14	SUCCESS	02	0.3	1971.7	00.2	1971.9
	17	SUCCESS	0.2	0.3	2667.08	1971.9	4630.95
	18	SUCCESS	02	05	3144.08	7424.94	10589.02
	1.9	BDCCERB	02	04	2120.79	4601.0	6002.3
	20	STOCCESS.	0.8	0.6	2290.76	16790.79	10097.85
	21	SUCCESS	02	0.4	3456.03	4002.3	10250.33
	22	SUCCESS	0.2	0.4	2694.74	10250.33	12963.06
	23	SUCCESS	03	0.6	4200.42	10097.85	22306.17
	24	SUCCESS	0.2	0.9	529.73	4635.55	55-60.60
	2.6	#DOCESS	02	0.4	1025.39	12963.06	13970.45
	26	SUCCESS	02	0.3	2344.92	1905.84	4320.46
	27	SUCCESS	02	04	642.78	13970.45	14021.23
	2.0	SUCCESS.	02	0.6	349.05	10509.02	10535.67
	29	SUCCESS	02	0.3	2790.77	5540.40	0347.44
		sing FCFS: \$402	100000000000				

Fig.8.1: FCFS Scheduling

II. Round Robin (RR) Scheduling

It can access the performance of the Round Robin algorithm using various criteria such as efficiency, fairness, response time, etc.

	01	SUCCESS	03	03	2443.48	00.1	2443.58
	00	SUCCESS	05	05	2540.83	00.1	2540.93
	21	SUCCESS	02	02	1067.16	2195.58	3262.74
	08	SUCCESS	03	03	1243.97	2443.58	3687.55
	02	SUCCESS	04	04	3835.4	00.1	3835.5
	22	SUCCESS	06	06	2319.49	1551.52	3871.01
	07	SUCCESS	04	04	1079.66	3835.5	4915.15
	03	SUCCESS	05	05	2438.23	2540.93	4979.16
	15	SUCCESS	04	04	1250.65	4915.15	6165.8
	10	SUCCESS	03	03	2629.44	3687.55	6316.99
	26	SUCCESS	06	06	2603.13	3871.01	6474.14
	04	SUCCESS	05	05	1985.73	4979.16	6964.89
	23	SUCCESS	04	04	1666.64	6165.8	7832.44
	11	SUCCESS	03	03	2120.88	6316.99	8437.87
	27 '	SUCCESS	04	04	842.78	7832.44	8675.22
	13	SUCCESS	03	03	543.31	8437.87	8981.18
	06	SUCCESS	05	05	2048.7	6964.89	9013.6
	29	SUCCESS	06	06	2736.53	6474.14	9210.67
	28	SUCCESS	04	04	1039.87	8675.22	9715.09
	16	SUCCESS	03	03	1971.7	8981.18	10952.88
	09	SUCCESS	05	05	2121.1	9013.6	11134.69
	19	SUCCESS	03	03	1135.82	10952.88	12088.7
	17	SUCCESS	05	05	3520.53	11134.69	14655.22
	20	SUCCESS	03	03	3131.47	12088.7	15220.17
	24	SUCCESS	03	03	929.73	15220.17	16149.9
	18	SUCCESS	05	05	3164.88	14655.22	17820.1
	25	SUCCESS	03	03	2062.55	16149.9	18212.45
M	akespan u	sing RR: 7526.6	8010168747				

Fig.8.2: RR Scheduling

III. Shortest Job First (SJF)

To calculate accuracy of Shortest Job First (SJF) scheduling algorithm, by

Accuracy= (Total Waiting Time of SJF)/(Total Waiting Time of Other Algorithm)*100%

2	14	SOCCESS.	0.5	05	650.3	00.1	\$52.4
	06	POCCESS	04	04	1442.72	00.1	1642.02
>	0.0	#DOCKES	0.8	03	2354.40	00.1	2354.50
8	07	SOCCESS	04	04	1075.66	1442.02	2722.47
2	01	ROCCERS	02	02	3160.45	00.1	3160.69
9	02	#DOCKES	0.6	04	3051.04	00.1	3051.14
	2.6	EDOCESS	0.6	05	3544.04	550.4	4103.24
	03	BOCCESS.	02	02	1703.53	3160.59	4072.62
	10	EDOCRES	0.8	0.3	2429.44	2354.50	4904.02
	08	SOCCESS	04	04	2500.04	2722.47	\$290.52
	0.6	BOCCESS.	06	0.6	1755.40	3091.16	5646.63
	21	EDCCRBS	0.6	0.6	395.55	5646.63	€04€.22
	0.6	SOCCESS	02	02	1877.04	4072.52	6450.30
	11	EDOCESS	04	04	1055.40	\$230.62	7005.99
	14	EDCCERS	08	0.8	3004.4	4103.24	7505.64
	20	FOCCESS	0.8	0.3	8131.47	4984.02	0116.45
	24	EDCCERS	04	06	2255.6	6046.22	8341.02
	0.9	EUCCERS	0.2	02	1925.42	6450.30	0376
	26	EQCCESS.	0 B	0.3	1599.10	0115.49	9714.66
	12	EUCCERS	02	02	2471.41	8376	10047.41
	17	EUCCERS	0.4	0.4	3770.22	7045.99	1005€.21
	29	EDCCERS	0.6	0.6	2734.83	0041.02	11078.36
	10	EUCCESS	0.5	05	3164.00	7909.64	11154.52
	27	SUCCESS	0.4	0.4	042.78	10086.21	11690.99
	1.0	EDOCESS	0.2	02	1208.61	10847.41	12056.03
	1.9	EDCCERS	08	05	2314.35	11154.83	13470.07
	2.0	SUCCESS	05	05	249.05	13470.07	13820.72
	2.2	EGCCE88	0.2	02	2105.53	12056.02	14161.68
	2.9	EUCCESS	0.2	02	301.90	14141.55	14543.83
	2.5	SUCCESS	02	02	2633.19	14543.55	17176.73
		sing SJF: 4640. heduler finishe					

Fig.8.3: SJF Scheduling

IV. Sunflower Optimization (SFO)

Analyze the accuracy of the Sunflower Optimization algorithm based on the comparison with benchmarks and any statistical analysis performed with factors like algorithm runtime and scalability in your interpretation.

Locality (LTP): y (LP = 24 MELLEY HINDOWSKI, LIST-5064(11546) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSKI, LIST-5064(11566) HINDOWSK	LATESCH 000000001.4.401747447400 000000001.4.401747447400 00000001.4.401747447400 00000001.4.4017474474010 000000014.5144474021100 000000014.5144745214900 000000014.5145415214900 000000014.5145415214900 000000014.5145415214900 000000014.5145415214900	00000000000000000000000000000000000000	10000000000000000000000000000000000000			
THE I ITELE INTERPOSED CONTRACTORS INTERPOSED	00000000000000000000000000000000000000	0000001748781,220444655000000 0000000773997.9757722997000000 0000002642356.036973.93000000 0000002642357.947545442000000 0000002454427.947545244000000 00000021574427.947545244000000 0000002157425.650055550000000	00000001228.27603463047000 000000000185.20020447100 00000000185.2002047100 00000001207.10001754714000 000000012020.1571482471000 00000000728.444508104000 00000000282.554046422471000			
ETEET 1000000000000000000000000000000000	00000000000000000000000000000000000000	0000001748781,220444655000000 0000000773997.9757722997000000 0000002642356.036973.93000000 0000002642357.947545442000000 0000002454427.947545244000000 00000021574427.947545244000000 0000002157425.650055550000000	00000001228.27603463047000 000000000185.20020447100 00000000185.2002047100 00000001207.10001754714000 000000012020.1571482471000 00000000728.444508104000 00000000282.554046422471000			
0101000000012, ICS790441224442 0101000000007, VETV20202146447 01010000000001, JACTANOSSAND 0101000000001, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 01010000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 01010000000, JACANOSSAND 010100000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 010100000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000, JACANOSSAND 0101000000000000000000000000000000000	00000000000000000000000000000000000000	0000001748781,220444655000000 0000000773997.9757722997000000 0000002642356.036973.93000000 0000002642357.947545442000000 0000002454427.947545244000000 00000021574427.947545244000000 0000002157425.650055550000000	00000001228.27603463047000 000000000185.20020447100 00000000185.2002047100 00000001207.10001754714000 000000012020.1571482471000 00000000728.444508104000 00000000282.554046422471000			
00000000000000000000000000000000000000	00000000000000000000000000000000000000	0000001748781,220444655000000 0000000773997.9757722997000000 0000002642356.036973.93000000 0000002642357.947545442000000 0000002454427.947545244000000 00000021574427.947545244000000 0000002157425.650055550000000	00000000000000000000000000000000000000			
101010000001, 36250048740048 0000000000, 442508058500 0000000000, 442610055500 0000000000, 45644251550 0000000000, 456451550 0000000000, 3569211454200 0000000000, 3569211454200 00000000000, 3569211454200 00000000000, 3569211454200	00000000055.588031891218400 00000000452.13845452207600 000000000475.02865540254000 00000000045.9885540254000 00000000055.9885175676494000 0000000055.88123415875000 00000000558.28123415875000	000000011393.37571203700000 000000423154.09597120300000 000002231231.13965441200000 000000145457.139744224000000 0000001251742.745744224000000 0000001251742.54505550000000	00000000000000000000000000000000000000			
00000000000000000000000000000000000000	00000000451,128457452277600 00000000410,82862574840250 00000000044,54825542254000 00000000055,54825542254000 00000000055,848245251767084000 00000000204,2882144825427400	0000000342356.00687130300000 000002423731.113665442600000 00000245667.547654224000000 000002287462.7567654224000000 000002287462.75668005550000000	00000000000000000000000000000000000000			
00000000000000000000000000000000000000	00000000470.82863378883560 00000000404.54825542254000 00000000052.5489717687884000 000000000455.48224128027500 00000000452.42824128027500	000002512312 113565441400000 00000026354567, 547654224000000 000002257462, 756796526400000 000002225726, 658055550000000	00000001050.15114624133000 000000007729.546555591045000 000000000820.14127542341000 000000000522.655466425241000			
0000000000005,4302452124752580 000000000006,200027766445280 000000000006,3205972116542880 000000000008,1120002077027508 000000000002,044245740021508	00000000044.54825542254000 000000000052.5485125467484000 000000000455.482241250875000 00000000024.288254825427500	0000002645667.547654224000000 000002637462.756790826400000 000002328756.658085550000000	0000000007729.54455559(10)50000 000000000000,7471075421470000 0000000005562.6554643(22)271000			
00000000000000000000000000000000000000	000000000055_3689121667454000 000000000455_4812481580375000 000000000204_2381248529627500	0000001897442.754798524400000 0000001287726.658085550000000	00000000000000,7471070623670000 0000000009262.6954064152670000			
00000000000000,0090971116542000 000000000000,1120002077017500 0000000000002,0646245740617600	000000000455.48124812801975000 000000000204.2881248829427500	0000001328726.658085550000000	000000009262.655406425267000			
88888899999999,1120002877017988 888889999999002,0646245748617688	000000000204.2982646829627600					
000000000000000000000000000000000000000		0000000047114.2233204448000000	00000000007.2204002942200000			
	000000000345.8187518751585000					
PRESERVATION REPARAMENTARY		0000001187741.6521820026000000	\$333000009000, 6835085268 (41000			
	00000000643,680485966570,00000	0000002179145.1942271930000000	0000000011865.7120679961900000			
11111133333334, 4249705504542441	000000000000000000000000000000000000000	0000462729130,1249414315000000	88880000000717.5608580815578888			
0000000000000.9721076046775120	000000000334.6797526556474600	0000000764117.5754368253000000	89999999977866.1452302657435099			
000000000000000000000000000000000000000	000000000332,9644123335469600	000000118945.8419393212000000	0000000005535.224213071411000			
10100000008.5000772067602000	000000000391.7841536459542500	0000001219847.3196899577000000	000000000000000000000000000000000000000			
1011100000002.0045524430303251	000000000284.69475614965664600	0000001257137.8354235291000000	0000000000740.2554250054520000			
F#####################################	0000000000004.4601652500481400	0000001815826.5776431162000000	10101000055598.6582540135421000			
110000000000000000000000000000000000000	000000000000000000000000000000000000000	0000001252778.025110400000000	000000009741.400968710400000			
000000000004.1411861202745650	000000000156.2835214136882800	0000000170895.4168785888000000	000000004826.0193356913250000			
000000000000000000000000000000000000000	0000000000212.0277973944424500	0000000759299.3213128119000000	000000007000.0000904677320000			
Come Mar						
ACTLET	AASTRICUS		20510182002			
100000000000000000000000000000000000000	000000000156.2836216136882999	0000000171899.4148789988000000	000000000000000000000000000000000000000			
		The smaller	st value: 000000000001.0007000199096633			
	00000000000000000000000000000000000000	000000021,700790000 00000021,700790000 00000021,00079000021,00000000000 000000021,000700000 000000021,000000000000000000000000		ULIIJUUUL 1998/98/18/18 000000021 494471184466 000001214 194471846 000001214 194471846 ULIIJUUUUU 299/98/18/18 00000021 4944718466 000001214 19447184 000000214 19447184 ULIIJUUUUU 299/98/18/18 00000021 4944718466 000012141 19447184 000000214 19447184 ULIIJUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU		

Fig.8.4: SFO Scheduling

V. Particle Swarm Optimization (PSO)

Creates a set of test instances representing different task scheduling scenarios. Each test case includes information such as task duration, resource requirements, and constraints.

					meetinctor, and X Sch	nidefatos ava 🛛 📄 Schedulari	for total polaria pana 11	
***		・目・ダを告号は	668 33	- C C O				
3	peckege 050							Q.
a tipe	Coule real scalar							
	25	SUCCESS	05	05	1025.39	5156.24	6181.62	
1	12	SUCCESS	04	04	2471.41	3864.4	6335.81	
1	19	SUCCESS	06	06	1283.74	5314.38	6598.12	
1	18	SUCCESS	02	02	1400.18	5570.26	6970.44	
	13	SUCCESS	0.4	04	1208.61	6335.81	7544.42	
	2.6	SUCCESS	02	02	929.73	6970.44	7900.16	
	14	SUCCESS	04	04	617.62	7544.42	8162.04	
	29	SUCCESS	05	05	2187.54	6181.62	8369.17	
	23	SUCCESS	0.4	04	381.98	8162.04	8544.02	
	20	SUCCESS	06	06	2298.76	6598.12	8896.88	
	21	SUCCESS	.06	06	399.59	8856.88	9296.47	
	27	SUCCESS	06	06	1425.96	9296.47	10722.44	
	26	SUCCESS	0.4	04	2344.92	8544.02	10888,94	
	10888.9399	99999999						
	Best fitne	ss value: 5199.	843366324225					
	Best makes	pan1 2715.21506	98281102					
	PSO.PSO Sc	heduler finishe	d1					
		ESSFUL Itotal t		de V				

Fig.8.5: PSO scheduling

Scheduling	Make span
First Come First	5129.625758533817
Serve(FCFS)	
Shortest Job	4217.043341627036
First(SJF)	
Round Robin (RR)	4006.570314680345
Sunflower	3557.046400653260
Optimization(SFO)	
Particle Swarm	2622.879784299157
Optimization(PSO)	

Model Evaluation result

IX. CONCLUSION

The cloud has gained increasing popularity in business and research circles for its numerous advantages, such as instant self-service, quality service, pay-as-you-go pricing, virtualization, and scalability. Recent studies have

Page | 852

introduced an innovative method for reducing energy consumption in task scheduling. Given that task scheduling is a crucial challenge in cloud computing, this paper has presented an ESFO algorithm that enhances the pollination process of the standard SFO algorithm to efficiently explore a solution space. The ESFO algorithm is proficient at determining an optimal task scheduling solution due to its use of polynomial time complexity. Through comparing the proposed algorithm with others, the outcomes demonstrate a significant enhancement with the ESFO algorithm.

REFERENCES

- G. Natesan, A. Chokkalingam, Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm, ICT Express 5 (2019)
- [2] N. Mansouri, B.M.H. Zade, M.M. Javidi, Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory, Comput. Ind. Eng. 130 (4) (2019)
- [3] J. Meshkati, F. Safi-Esfahani, Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing, J. Super Comput. 75 (5) (2019).
- [4] J. Yang, et al., A task scheduling algorithm considering game theory designed for energy management in cloud computing, Future Gener.Comput. Syst. 105 (2020).
- [5] X. Wei, Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing, J. Ambient Intell. Humaniz. Comput. (2020).
- [6] Shukla, D. K., Kumar, D., & Kushwaha, D. S. (2021). Task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II. Materials Today: Proceedings.
- [7] Azizi, S., Shojafar, M., Abawajy, J., & Buyya, R. (2022). Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi- greedy approach. Journal of network and computer applications, 201, 103333.
- [8] Mangalampalli, S., Swain, S. K., & Mangalampalli, V. K. (2022). Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm. Arabian Journal for Science and Engineering, 47(2), 1821-1830.
- [9] Ibrahim, I. M. (2021). Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(4), 1041-1053.
- [10] Mangalampalli, S., Swain, S. K., & Mangalampalli, V. K.(2021). Prioritized Energy Efficient Task Scheduling Algorithm in Cloud Computing Using Whale

Optimization Algorithm. Wireless Personal Communications, 1-17.

- [11] Mukherjee, D., Nandy, S., Mohan, S., Al- Otaibi, Y. D., & Alnumay, W. S. (2021). Sustainable task scheduling strategy in cloudlets. Sustainable Computing: Informatics and Systems, 30, 100513.
- [12] Emami, H. (2022). Cloud task scheduling using enhanced sunflower optimization algorithm. ICT Express, 8(1), 97-100.
- [13] Ijaz, S., Munir, E. U., Ahmad, S. G., Rafique, M. M., & Rana, O. F. (2021). Energy-makespan optimization of workflow scheduling in fog-cloud computing. Computing, 103(9), 2033-2059.
- [14] D. Ding, et al., Q-learning based dynamic task scheduling for energy efficient cloud computing, Future Gener. Comput. Syst. 108 (2020) 361–371.
- [15] J. Xiao, et al., Game theory-based multi-task scheduling in cloud manufacturing using an extended biogeographybased optimization algorithm, Concurr. Eng. 27 (4) (2019) 314–330.