Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

An Improved SARSA Learning Hyper-Heuristic Algorithm For Task Scheduling in Cloud Computing

Arvind Upadhyay[1,2,*], Ramesh Thakur[3], Archana Thakur[4], Kavita Thorat Upadhyay[2]

[1]Institute of Engineering & Technology, Devi Ahilya Vishwavidyalaya, Khandwa Road, Indore Madhya Pradesh, India 452007
[2] IPS Academy, Institute of Engineering & Science, A. B. Road, Indore, Madhya Pradesh, India 452012
[3]International Institute of Professional Studies, Devi Ahilya Vishwavidyalaya, Khandwa Road, Indore Madhya Pradesh, India 452007
[4]School of Computer Science & IT, Devi Ahilya Vishwavidyalaya, Khandwa Road, Indore Madhya Pradesh, India 452007
*Corresponding Author: upadhyayarvind10@gmail.com

Abstract- The increasing computational demands in modern cloud environments highlight the critical need for efficient task scheduling strategies. Task scheduling, a well-known NP-complete problem, seeks to optimize the allocation of tasks to resources while meeting predefined objectives such as minimizing completion time. This paper introduces a hyper-heuristic algorithm, Improved State Action Reward State Action (ISARSA), which uses reinforcement learning to dynamically regulate a pool of meta-heuristics, including Genetic Algorithm (GA), Cuckoo Search (CS), and Particle Swarm Optimization (PSO). SARSA, is an on-policy, high-level decision maker that is trained to prefer the most appropriate low-level heuristic based on the present system state and realized rewards, and the chosen meta-heuristic carries out the actual task-VM mapping optimization. The study is carried out in two phases: the first part involves simulation-based experiments demonstrating that ISARSA has a smoother learning behavior, adaptability, and more stable convergence in comparison to the traditional deterministic, heuristic, and Q-learning-based schedules; the second part is the implementation of the framework in a Hadoop cluster, where the same performance trends are observed and prove the robustness and practical applicability of the proposed framework.

Keywords: Cloud Computing, Hyper-heuristic, Nature Inspired Algorithms, NP-complete problems, GA,CS,PSO.

1. Introduction

The parallel and distributed commuting is nolongera choice because of its processing capabilities of information that is generated by today's information centric world. The blast of information has shaken the foundationsof centralized computing as they are inefficient in handling today's computing demand. This situation depicts the necessity of the decentralized resources which are essentially parallel and distributed. The best example of these systems is cloud computing environment. A "cloud" of computing resources is nothing more than various computing resources which are available on dynamic requirements by the end users[1-3]. This definition gives raise to various models of cloud computation[4]e.g. Infrastructure as a Service (IaaS).

Although cloud computing offers significant benefits, it is accompanied by various challenges, among which efficient task scheduling remains a fundamental research problem. Scheduling can be considered at various levels and with various constraints, but here our concentration is on how to schedule "tasks" or "jobs" submitted by end users. Tasks are the fundamental unit of work in cloud which are assigned to the various available machines. The problem of task scheduling basically "demands a schedule of tasks which can be given to the cloud such that some predefined conditions are met or in scientific terms optimized". This problem can be presented as an optimization problem and then a solution to that optimization problem can be proposed.

But, there exists no algorithm which can be used to solve this problem in polynomial time because the cloud scheduling problem, in general is NP complete [5]. In this case researches propose solutions in the form of heuristics which are algorithms that can produce good solutions fast.A few heuristic solutions to the cloud scheduling problem can be found in[6-7] .While heuristics are problem specific there are also meta-heuristicswhich are problem independent and hence more general than heuristicswhich are also tested on the scheduling problem. For example Ant Colony Optimization (ACO)[8-13],Particle Swarm Optimization[14-20], Genetic Algorithm[21-25]etc. are a few interesting meta-heuristics that have achieved better than heuristics performance on various problems. Moreover, we also have the so called hybrid meta-heuristics[26–32] which are hybrid of the advantages of nn number of meta-heuristics, usually n = 2.

Hybrid meta-heuristic approaches aim to combine the strengths of multiple meta-heuristic algorithms to achieve better optimization performance. However, these combinations are typically problem-specific and are manually designed by human experts. To overcome these limitations, there is a growing need to automate the selection and combination of optimization strategies, making them problem-independent, more robust, performance-oriented, time-efficient, fault-tolerant, and cost-effective.Researchers have developed even more advanced algorithms which can take many advantages of the important properties of many algorithmssimultaneously. These algorithms are hyper-heuristics [33-34]. Hyper-heuristics are quite new research methodology and even now there remains much to contribute to this methodology from many perspectives.

Therefore, here in this paper we give the following contributions:

(1) We solve the cloud scheduling problem which is NP-Complete.
(2) We solve the aforementioned problem by proposing a hyper-heuristic employing Improved State Action Reward State Action (ISARSA) algorithm.
(3) ISARSA introduces a learning strategy in our proposed hyper-heuristic.
(4) We have chosen meta-heuristics quite interestingly as each of them are established in literature.

With these contributions our hypothesis is that the proposed ISARSA hyper-heuristic algorithm will outperform/comparable to other compared algorithms most of the time.

The remainder of this paper is organized as follows. Section II presents the background of the study, including the hyper-heuristic methodology and the algorithms employed. Section III describes the proposed learning hyper-heuristic algorithm. Section IV details the implementation methodology and presents a comparative performance evaluation of the proposed approach against existing state-of-the-art algorithms. Section V discusses and analyzes the experimental results. Finally, Section VI concludes the paper and outlines potential directions for future research.

2. Background

Literature review

Machine Learning (ML) algorithms have become ubiquitous, achieving remarkable accuracy across diverse problem domains[35-40]. These algorithms can serve as heuristics, meta-heuristics, or hyper-heuristics, offering adaptable and efficient solutions. This section reviews the state-of-the-art research most relevant to our proposed approach, focusing on advancements in these categories in the conscious tabular format.

Category	Reference	Problem Addressed	Approach	Metrics Used	Comparison to ISARSA
Heuristic	[6]	Task scheduling and resource allocation in the cloud.	Developed a Multi-facetedheuristic-based scheduler for task prioritization and resource optimization.	Turnaround Time, Response Time	ISARSA differently dynamically adapting strategies via reinforcement learning for varying workload conditions.
	[41]	Meeting Deadlines in Cloud Workflows.	The proposed algorithm implements a proactive task replication strategy to mitigate the impact of performance variations and potential delays.	Likelihood of Meeting Deadlines, Total Execution Time	ISARSA integrates heuristic strengths more dynamically providing better scalability and adaptability.
Meta-Heuristic	[42]	Cloud task scheduling	This approach integrates a new encoding mechanism into the GWO GWOEM (Grey	Makespan	ISARSA dynamically selects from multiple meta-heuristics rather than relying on fixed meta-

			Wolf Optimization) algorithm to address the limitations of traditional evolutionary algorithms in cloud task scheduling.		heuristics, enhancing robustness.
	[43]	Task Scheduling in Cloud Computing	Hybrid Optimization	Makespan, Computational Cost, Reliability, Predicted Energy Consumption, Resource Utilization	ISARSA's key advantage lies in its ability to dynamically adapt and learn, making it more robust and responsive to the ever-changing conditions of cloud environments.
	[44]	Task Scheduling in Cloud Computing	The proposed algorithm incorporates a multi-adaptive learning strategy to enhance the performance of the traditional Particle Swarm Optimization (PSO) algorithm (MLPSO).	Makespan, Load Balancing,	MALPSO aims to improve the performance of a single meta-heuristic (PSO), while ISARSA leverages the strengths of multiple meta-heuristics through dynamic selection and adaptation.
	[45]	Task Scheduling in Cloud Computing.	Hybrid Genetic Algorithm and Gravitational Search Algorithm.	Makespan, Resource Utilization. Energy Consumption, Throughput	ISARSA, leverages a broader range of meta-heuristics and employs reinforcement learning to dynamically adapt to the specific characteristics of the cloud environment.
	[46]	Task Scheduling in Cloud Computing	This novel algorithm combines the Jaya algorithm and Synergistic Swarm Optimization (SSO) to leverage their strengths.	Accuracy, Solution Quality, Convergence Speed	ISARSA is more advantageous in dynamic and complex environments where adaptability and the ability to leverage multiple meta-heuristics are crucial.
	[38]	Workflow Scheduling in Quantum-based Hybrid Clouds	Multi-objective Quantum-inspired Genetic Algorithm (MQGA)	Makespan, Energy Consumption	MQGA focuses on enhancing the performance of a single meta-heuristic (genetic algorithm) by

					incorporating quantum-inspired concepts. ISARSA emphasizes dynamic adaptation and the intelligent selection of multiple meta-heuristics.
Hyper-Heuristic	[34]	Scientific workflow cost optimization in cloud computing.	Proposed a cost-aware hyper-heuristic framework integrating diverse meta-heuristics.	Cost Optimization, Workflow Completion Time	ISARSA emphasizes task completion time which potentially improves cost, optimizing resource utilization dynamically for real-world scenarios like Hadoop.
	[33]	General hyper-heuristic framework for cloud task scheduling.	Proposed a general-purpose hyper-heuristic method combining low-level heuristics.	Makespan	ISARSA improves upon this by using reinforcement learning to drive heuristic selection dynamically, achieving higher efficiency and adaptability.
	[47]	Efficient Workflow Scheduling in Cloud Computing	The proposed method leverages evolutionary algorithms to automatically discover and refine scheduling rules.	Makespan, Execution Cost	The evolutionary rule discovery approach focuses on evolving effective scheduling rules through evolutionary computation. ISARSA emphasizes dynamic adaptation and the intelligent selection of meta-heuristics.

Observations

1). Heuristics:

- Useful for specific problems but lack adaptability for broader scenarios like dynamic cloud workloads.
- ISARSA's learning mechanism overcomes this limitation by dynamically switching between heuristics based on task requirements.

2). Meta-Heuristics:

- Strong optimization techniques for specific problem types (e.g., PSO for discrete problems).
- Hybrid approaches often suffer from static design. ISARSA offers flexibility by dynamically selecting and integrating algorithms during runtime.

3). Hyper-Heuristics:

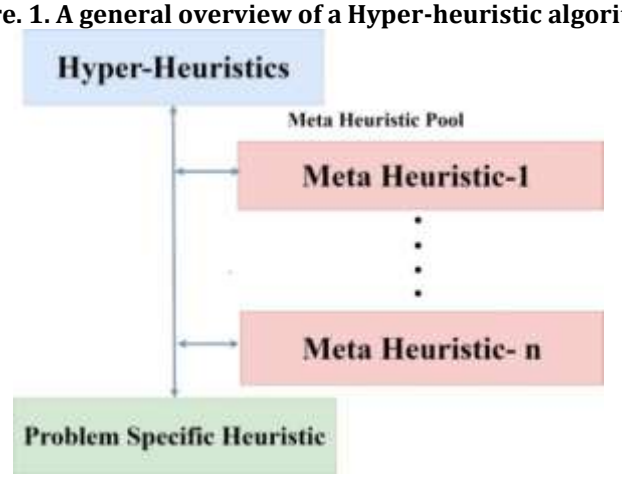
- Most existing hyper-heuristic approaches focus on static selection or combination of algorithms.
- ISARSA's reinforcement learning-based dynamic adaptation ensures better performance across varying workloads.

Hyper-heuristic methodology

Here, we describe the hyper-heuristic methodology, State Action Reward State Action (SARSA), Cuckoo Search (CS), Particle swarm optimization (PSO) and Genetic algorithm (GA).SARSA is a Reinforcement Learning (RL) algorithm while others are Nature Inspired Algorithms (NIAs) [48-50], which are quite popular.This section also motivates and leads us to our proposed aim.

Consider figure 1 which demonstrates the working of a hyper-heuristic in general: Figure 1, has the following elements:

Figure. 1. A general overview of a Hyper-heuristic algorithm



1) Hyper-heuristic: It is an algorithm that regulates other algorithms in order to generate a problem specific strategy which will then yield near optimal solution to the problem at hand. The regulation can be either selecting any one of the algorithms available at the meta-heuristic pool or creating a new heuristic based on the available algorithm in the pool. It is often called high level-heuristic.

2) Meta-heuristics: Often called low-level heuristic algorithms which work on the specific problem directly. Typically, n numbers of meta-heuristics are taken as a meta-heuristic pool which is made available to the hyper-heuristic to work on.

3) Specific problem: It is some real word problem formulated mathematically as an optimization problem. Hyper-heuristics are generally suitable for "hard" problems i.e. NP-complete or NP-hard.

From the above description our strategy is as follows:

1) We improve the SARSA algorithm to get Improved State Action Reward State Action and apply it as our hyper-heuristic.

2) We choose CS, PSO and GA as our meta-heuristic in the pool.

3) The specific problem of our interest is cloud scheduling.

State Action Reward State Action (SARSA)

Reinforcement learning (RL) is the branch of machine learning in which programs learn by interacting with the real-world environment. One novel idea of RL is the Temporal Difference (TD) learning which is a combination of Monte Carlo (MC) methods and Dynamic Programming (DP). That is in TD, like MC learning happens with raw experience without a model of the environment while like DP estimates are updated with respect to learned estimates without waiting for final outcome. In the subject of TDlearningone cannot overlook SARSA[51], [52] which is basically a quintuple (s, a,r, s', a') (s, a, r, s', a') where:

s= original state, a = original action, r = reward observed in the following state, s' = new state, a' = new action.

$a' = new\ action$The Q-value (Q(s,a)) is just the values of state and action pair observed by the algorithm. All of the Q-valuesare available in as a table called Q-table. Q-valueswhich are fundamental object are updated continuously while it is not necessarily the case that the maximum reward is always used to update the Q-values, instead following the policya new action reward can be selected. Hence SARSA is on-policy temporal difference learning (on policy TD-learning). This algorithm is completely described as:

Algorithm-I (SARSA)
1: Input a policy π (that uses the action value function).
2: Initialize Q(s,a)← $0Q(s,a)$← 0for all s∈$s_t$$s \in S_t$ and a ∈ A_t.
3: loop: for each episode
4: Initialize en
vironment to provides.
5: do:
6: Choose a froms $s$using $\pi\pi$ while ties must be broken randomly.
7: Take action aand observer ands'.
8: Using π choosea'froms' while ties must be broken randomly.
9: Q(s,a) ← Q(s,a)+a[r+γQ(s',a')-Q(s,a)].$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$.
10: s←s'$s \leftarrow s'$.
11: a←a'← a'.
12: while $s$is not terminal.

Let, Q-tables are initialized with:

(i) the states $\mathbf{s \in s_t}$, where s_t is the set of all states and

(ii) action$\mathbf{a \in A_t}$, where A_t is the set of all actions**.**

Then, the update rule is:

$$\left(\frac{Q(s,\ a)}{new\ Q\ value}\right) \leftarrow \left(\frac{Q(s,\ a)}{current\ Q\ value}\right) + \left(\frac{\alpha}{learning\ rate\ used\ to\ balance\ exploration\ and\ exploitation}\right)\left[\left(\frac{r}{reward}\right) + \left(\frac{\gamma}{discount}\right)\left(\frac{Q(s',\ a')}{observed\ future\ value}\right) - \left(\frac{Q(s,\ a)}{current\ value}\right)\right].$$

This update rule has a few hyper-parameters which can be described as follows:

(i) α is the learning rate as described with the update rule is between 0 and 1 where high values learn faster but performs less averaging.

(ii) γ discounts the expected return, it models the fact that the algorithm must decide on immediate rewards rather than future rewards.

The action value function isπ(a|s, Q(s,a)).

Please note that SARSA chooses an action before it updates the action-value function (steps 6-7). This behavior of SARSA can be very advantageous in some situation although in others it can bedisadvantageous or marginal. Although, we can enumerate the advantages of choosing SARSA in the Proposed Setting:

Dynamic and On-Policy Learning:
SARSA's on-policy nature means it updates the action-value function based on the action derived from the current policy. This allows SARSA to adapt dynamically to changing states and policies, which are crucial in task scheduling where the environment is highly dynamic, and task arrivals, can be unpredictable.

Stochastic Environment Suitability:
Cloud environments often exhibit stochastic behavior due to variable resource availability, workload fluctuations, and heterogeneous systems. SARSA's reliance on observed transitions (s, a, r, s', a') ensures robust learning even in such uncertain conditions.

Smooth Policy Evolution:
SARSA updates Q-values based on the current policy rather than directly optimizing for the maximum reward. This characteristic makes it less prone to drastic policy shifts, ensuring stable and gradual improvements—a key benefit in multi-task scheduling scenarios where abrupt policy changes could disrupt resource allocation.

Better Exploration-Exploitation Trade-Off:
SARSA inherently balances exploration and exploitation through its learning rate (α) and discount factor (γ). In task scheduling, this ensures that underexplored actions (e.g., scheduling strategies) are still considered, improving overall system performance in the long run.

Alignment with Real-Time Scheduling:
SARSA updates Q-values incrementally after every action, making it well-suited for real-time systems like cloud task scheduling. This incremental learning aligns with the continuous flow of tasks and decision-making required in cloud platforms.

Customizability for Multi-Objective optimization:

SARSA allows easy incorporation of customized reward functions that align with task scheduling objectives, such as minimizing makespan, maximizing resource utilization, and balancing workloads. This flexibility is key in tailoring the algorithm to the specific requirements of cloud environments.

Integration with Meta-Heuristics:

SARSA's capability to guide policy-based action selection aligns well with meta-heuristic algorithms like GA, PSO, and CS. By using SARSA as a hyper-heuristic controller, the system can dynamically decide which meta-heuristic to use based on real-time feedback, optimizing task scheduling performance.

Reduced Computational Overhead Compared to Off-Policy Methods:

SARSA's reliance on the current policy reduces the computational overhead of evaluating all possible actions (as in Q-learning). This efficiency is particularly valuable in large-scale cloud systems with numerous tasks and resources.

Genetic Algorithm (GA)

GA[49], [53] is a NIA and the inspiration of this algorithm is the process if natural selection in nature. It is a well-known algorithm and has a demonstrated successful history[53] of applications. Therefore, we are not disusing GA in detail.

Cuckoo Search (CS)

CS [49], [50]is another NIA inspired by the reproduction habit of Cuckoo bird. Since, we are interested in the algorithm the said habit of the cuckoo bird can be idealized as follows:

(1) We assume an idealized environment in which there exist a predefined number of cuckoos. Each cuckoo can lay only one egg at a time and then the laid egg is carefully inserted into some randomly chosen nest.

(2) Best nest or eggs survives to form the next generations of expectedly better solutions than the previous generations.

(3) There are a finite number of host birds, the original owners of the respective nests, who can discover the laid cuckoo's egg with probability p ϵ (0,1). In this case the nest/egg can be abandoned.

One point to note here is the fact that there is no distinction between the words cuckoo, nest, egg or solution from the perspective of our implemented algorithm.

Particle Swarm Optimization (PSO)

PSO[54] is again a NIA which is inspired by the swarm behavior of species in nature. Like GA, it is also a well-known algorithm and forms the subject of swarm intelligence.

The heart of PSO is the velocity and position updates of particles. With respect to any iteration t,let there aren particles in the system. Then for thei^{th}particle:

(i) Velocityupdate is performed using the formula:

$$v_i^{t+1} = v_i^t + \alpha\epsilon_1[g^* - x_i^t] + \beta\epsilon_2[x_i^{*(t)} - x_i^t]$$

,(1)

Here, ϵ_1 and $\epsilon_2$are random vectors,g* is current global best and$x_t^{*(t)}$ is the current particle's individual best.

(ii) Position update is performed using the formula:

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

(2)

3. Problem Formulation & Proposed Algorithm

Cloud task scheduling is developed as an optimization problem where a group of heterogeneous Virtual Machines (VMs) is to implement a group of independent user tasks with the goal to achieve optimality in the overall system performance at minimum cost associated with resource usage. Where T={t_1,t_2,…t_n} is the set of tasks, and VM= {vm_1,vm_2,…,vm_m}is the set of available virtual machines. Each task $\mathbf{t_i}$ is described by its computational workload, and the time needed to run the task, and each virtual machine $\mathbf{vm_j}$is described by a given processing capacity in Million Instructions Per Second (MIPS), and the memory and bandwidth resources.

1. Task Execution and Completion Time

The completion time of task t_i on VM $vm_j$is modeled as:

$$C_{ij} = \frac{L_i}{P_j} \quad (3)$$

where:

- L_i workload length of task t_i (in Million Instructions),
- P_j = computational capacity of vm_j (in MIPS),
- C_{ij} = execution time of task t_i on VM vm_j.

The finish time of a VM after processing a batch of tasks is:

$$FT_j = \sum_{t_i \in VM_j} C_{ij} \quad (4)$$

2. Objective Function

The objective of the cloud scheduler is to determine an execution sequence for these tasks that minimizes a given performance metric.

Flowtime: It is the total finishing times of all the tasks.

$Flowtime = \sum_{i \in task} F_i$, i denote the finishing time of task i. (5)

Now, the scheduling objective is to minimize flowtime:

$min\ s\ \sum_i F_i$,subject to $t_i \rightarrow vm_j$ (6)

3 State Space and Load Conditions

To incorporate workload adaptiveness, system states are classified according to VM load:

S={s_1,s_2,s_3}

s_1=low load,s_2=medium load,s_3=high load

The load of VM $vm_j$is defined as:

$$\lambda_j = \frac{\sum_{t_i \in VM_j} L_i}{P_j} \quad (7)$$

4. Action Space (Meta-Heuristic Selection)

The high-level decision layer selects one heuristic at each iteration:

A={a_1,a_2,a_3}

a_1=GA, a_2=CS, a_3=PSO

5. Reinforcement Learning Model

The action–value function is updated using the temporal-difference rule

$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$ (8)

where α denotes the learning rate and γ is the discount factor.

where $\alpha_t = 1 - 0.9 \left(\frac{t}{t_{max}}\right)$ (9)

is time dependent learning rate, γ is discount factor for future reward, $\mathbf{r_t}$is reward from executing a heuristic.

The Improved SARSA based Hyper-Heuristic (ISARSA) framework as shown in figure 2 is developed as a two-layer adaptive scheduling structure in the cloud environment. Under this model, the ISARSA agent acts as a high-level decision controller, whereas a metaheuristic pool which is a combination of Genetic Algorithm (GA), Cuckoo Search (CS), and Particle Swarm Optimization (PSO) are low-level optimizers. The controller picks a suitable metaheuristic depending on the state and workload in the systems.

ISARSA has an on-policy reinforcement learning mechanism, in which the state-action policy is updated in real time using performance feedback. The framework as shown in figure 2, has the ability to adaptively converge and stabilize scheduling behaviour under dynamic cloud workloads by continuously refining its decision strategy using reward-based learning, which has the benefit of leveraging the complementary capabilities of GA, CS and PSO.

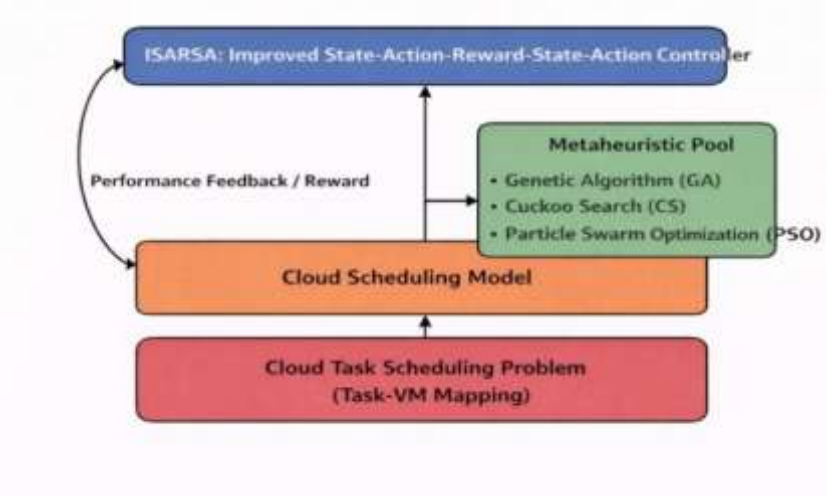
The proposed improvements to the SARSA algorithm are as follows:

(i) Let the set of states be$S = \{s_1, s_2, s_3\}$ and the set of actions be $A = \{a_1, a_2, a_3\}$.

Since we have used 3NIAs in our meta-heuristic or NIA pool the possible states and actions are 3. Here s_1=low load, s_2=medium load, s_3=high load, a_1=GA, a_2=CS, a_3=PSO .

(ii) The learning rate α at iteration t is calculated as $1 - \left(0.9 * \frac{t}{t_{max}}\right)$ where t_{max} is the maximum iteration.

Figure. 2: framework of proposed ISARSA for cloud task scheduling.



(iii) Reward is given by our reward function r_f:

$r_f = \{+1\ if\ f_{new} > f_{best}\ 0\ if\ f_{new} = f_{best}\ -1\ if\ f_{new} < f_{best}$.

(10)

f_{new}= new reward,

f_{best}= best reward ever observed.

(iv) In ISARSA the following two polices were implemented (π):

(1) $Greedy$ policy: It is defined as: choose the action with the highest value calculated as: $arg\ arg\ Q(s, a_k)$, $k = 1,2,3$ ($k$stands for the k^{th}action.

(11)

Basically, it says that always choose the best reward i.e. with +1 value.

(2) $\in -greedy$policy: It consist of two parts:

(a) If less than ρ% probability exists then use $\in$% probability for selecting actions. Compute the index k of the selected actions as:

$k = \lceil 3 * rand(\ ) \rceil$, here $rand()$ value is [0 1].

(12)

(b) If ρ% probability exists then: use the roulette-wheel selection to choose the k^{th}action. The probability to choose the k^{th}action for $k = 1,2,3$ is calculated according to the following:

$$probablity\ (a_k) = \frac{exp\ (Q(s,a_k))}{\sum\ exp\ (Q(s,a_k))}$$

(13)

Pseudo code of Proposed ISARSA Algorithm

Input:

Task set T, vm set VM,

States S={s1,s2,s3}(low, medium, high load),

Actions A={a1,a2,a3} (GA, CS, PSO)

Initialize:

Q(s,a)=0for all **s∈S,a∈A**

Set learning parameters **γ**, exploration probability **ρ** and $\mathbf{t_{max}}$

Initialize best fitness $\mathbf{f_{best}} \leftarrow -\infty$
Randomly select the initial state $\mathbf{s_t}$
While $(t \leq t_{max})$ do
1. Observe current state $\mathbf{s_t}$.
2. Update learning rate

$$\alpha_t = 1 - 0.9\left(\frac{t}{t_{max}}\right)$$

3. Action (Metaheuristic) Selection
Generate random number $\mathbf{p \in [0,1]}$
If $p < \rho$ then
Select action a_t randomly (Exploration)
Else
Select action a_t using roulette-wheel selection based on
probablity$(a_k) = \frac{\exp(Q(s,a_k))}{\sum \exp(Q(s,a_k))}$(Exploitation)
4. Execute Selected Metaheuristic
If $a_t = a_1$ then
Execute **GA** scheduling
Else if$a_t = a_2$ then
Execute **CS** scheduling
Else
Execute **PSO** scheduling
5. Evaluate scheduling solution and compute fitness f_{new}
6. Reward Calculation
If $f_{new} > f_{best}$then
$r_t = +1$, $f_{best} = f_{new}$
Else if$f_{new} = f_{best}$then
$r_t = 0$,
Else
$r_t = -1$,
7. Determine next state s_{t+1}
8. Select next action a_{t+1} using the same policy (Step 3)
9. Update Q-value (SARSA rule)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t[r_t + \gamma Q(Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

10. State Transition
$s_t \leftarrow s_{t+1}$, $t \leftarrow t + 1$
End While
Output:Optimized Q-table and adaptive metaheuristic selection policy (GA/CS/PSO)

The ISARSA can be explained in steps as follows:
1.Initialization: Initialization of the Q-table is done with all the workload states and metaheuristic actions, and this allows the learning process to be unbiased.
2. State Observation: The existing load in the cloud is reduced to a discrete form (low, medium, or high load), which gives a condensed view of the system conditions. 3.Learning Rate Update: To speed up early learning, an adaptive learning rate is applied to allow consistent convergence in subsequent iterations.
4. Action Selection: ISARSA trades exploration and exploitation by employing a probabilistic policy which is a combination of random selection and softmax-based roulette-wheel selection.
5. Metaheuristic Execution: The chosen action will trigger the respective metaheuristic (GA, CS, or PSO) to produce a solution to the task scheduling.
6. Reward Computation: This is rewarded according to improvement in the schedule performance, which guides policy refinement.
7. State Transition: The system moves to a new work load state that indicates new resource conditions.
8. Q-Value Update: The SARSA rule is used to update the Q-table, which is based on immediate reward and future course of action.
9. Iteration: This process is repeated until convergence, which causes an adaptive metaheuristic selection policy.

4. Implementation and Results

The proposed ISARSA-based scheduling framework performance was evaluated by performing large-scale experiments with the CloudSim simulator and then testing with a real-world cloud platform based on Hadoop. Section 1 gives the results of the simulation and the analysis discussions, and Section 2 give the experimental configuration and performance performances that were realized in the real cloud environment.

The CloudSim toolkit has been used to simulate a realistic cloud infrastructure that has heterogeneous virtual machines, changing task arrival patterns, and dynamic resource management policies. The traditional methods of scheduling (FCFS, Fair Scheduler, and HHSA) and the suggested ISARSA hyper-heuristic were run in the same simulation environment to guarantee equal and objective results of performance analysis. The key performance indicators involved in the experimental analysis included: makespan, average flow time, and throughput, resource utilization indicators which included CPU, memory, and VM utilization. Workload traces (WC98 data) to represent realistic operational requirements were included to allow a thorough evaluation of the flexibility and effectiveness of the ISARSA framework to dynamic cloud workloads.

1. Performance Evaluation of Proposed ISARSA Framework on cloudsim

Experimental Setup

Table 1: Parameters used in experiment

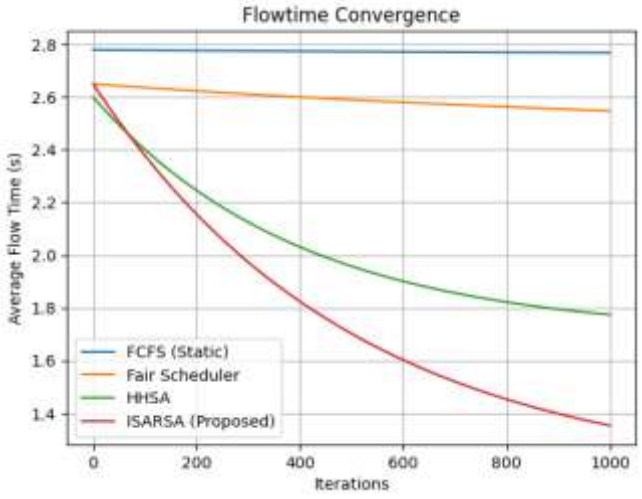
Parameter	Configuration
Dataset	WC98 Web Server Trace
Number of Tasks	1,000 – 10,000
VM Configuration	Heterogeneous (MIPS: 500–2000)
Learning Parameters	($\epsilon = 0.1$), ($\gamma = 0.8$), (t_{max}=1000)
Heuristic Pool	GA, CS, PSO
Performance Objective	Minimize (F) (flow time)

The parameters used in the experiments are shown in table 1. Evolution of the learning process is characterized by exploration intensive behavior during early stages, whereas, during the approach to convergence, the learning process is characterized by exploitation directed policy execution.

Convergence Behavior

The convergence of flow time of various scheduling algorithms is shown in figure 3. The FCFS does not have any significant improvement because it is a static approach and the Fair Scheduler gains only a minimal improvement because it focuses on fairness rather than optimization. HHSA is more efficient in selecting flowtime by using heuristics but saturates early without learning state-actions. On the other hand, ISARSA optimizes the flowtime through the continuous optimization of its policy through on-policy reinforcement learning, which leads to stable and progressive convergence.

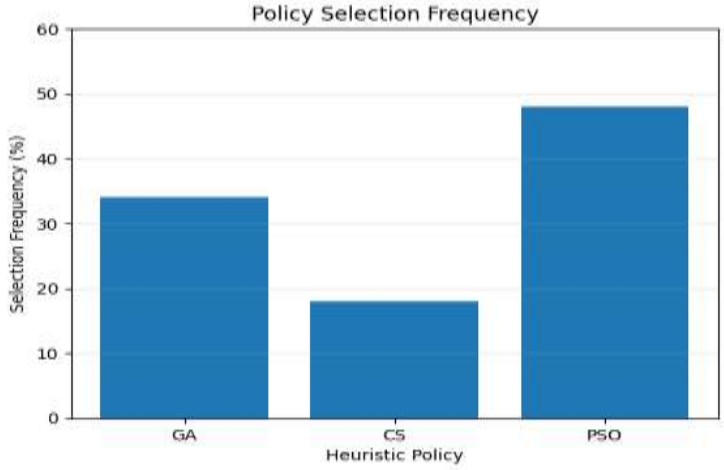
Figure 3. Flow time convergence comparison for FCFS, Fair Scheduler, HHSA and the proposed ISARSA.



Policy Selection & Learning Dynamics

Figure 4 gives the frequency of policy selection of the heuristic operators on execution. PSO is chosen the most, which means that it has a high and stable rate of rewards in different load conditions. The selection frequency of GA is moderate because it has a balanced exploration ability and steady performance. Conversely, CS is the least preferred option because it has a slower convergence rate and lower instantaneous reward, which is not beneficial when using on-policy reinforcement learning.

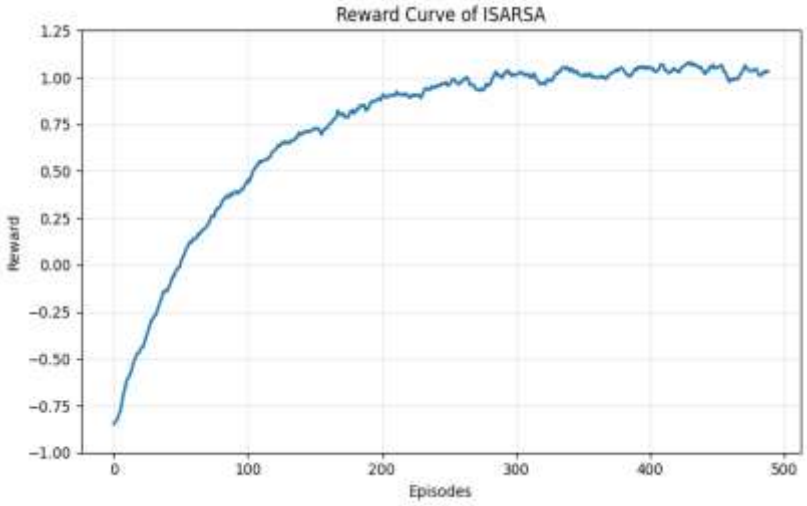
Figure 4: Policy selection frequency validating learning-based heuristic prioritization under varying load.



Reward Curve Analysis

The reward curve is used to indicate the dynamics of the ISARSA algorithm in terms of learning over training episodes. The values of rewards are initially low and highly fluctuating because of the prevalent exploration where the agent often chooses the suboptimal actions in order to explore the environment. The reward gradually grows as the training advances, which means that the agent is acquiring more efficient scheduling policies and refining the choice of actions in the light of the experience gathered. The smooth increase without sharp peaks is typical of on-policy learning, in which the changes are conservative and stability is valued Figure 5 therefore proves that ISARSA smoothly approaches a stable policy where the rewards are always higher thus effective learning and at the same time performance improvement with time.

Figure 5: Reward curve of ISARSA across episodes, showing exploration, transition, and stability phases.

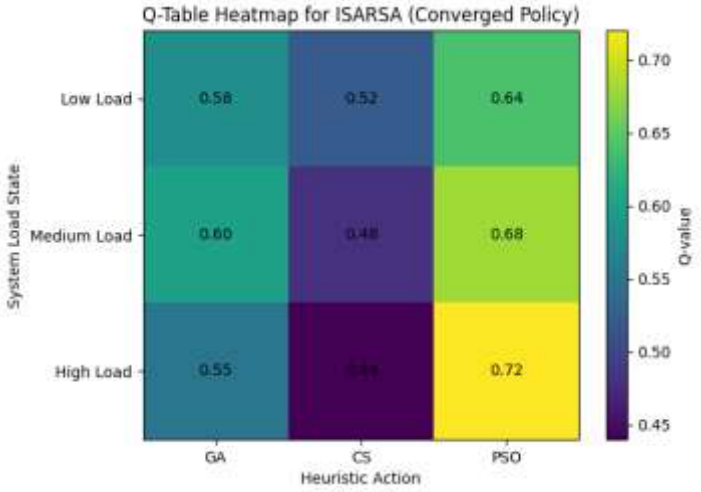


Q-Table Heatmap (Policy Convergence Behavior)

The stabilized action-value estimates obtained by ISARSA at convergence are represented by the Q-table heatmap as an estimate of the expected long-term reward and not immediate performance. The largest Q-

values of PSO are observed in all states of the load since the method is always able to lower the flowtime during congestion, and the medium size of the values is a direct result of the updates of SARSA on policy, which restricts the optimism overestimation. GA has intermediate and fairly constant Q-values, meaning that it is robust to different load conditions, whereas CS gets the lowest Q-values because it converges more slowly and its short-term rewards are less predictable. Figure 6 of these learned action preferences proves that ISARSA learns a balanced and stable schedule policy with conservative reinforcement learning.

Figure 6: Q-Table heatmap showing action preferences of the ISARSA agent across different workload states.



Key Findings

The suggested ISARSA framework is significantly better than FCFS, Fair Scheduler, and HHSA in flowtime and throughput, indicating the usefulness of on-policy reinforcement learning in scheduling tasks in clouds. Results of flowtime convergence prove that ISARSA is able to optimize in a manner that is stable and gradual, not exhibiting any unrealistic convergence behavior but at the same time delivering the lowest objective value of any compared methods.

The reward convergence curve confirms that ISARSA is stable in its learning process because it does not exhibit oscillatory or erratic behavior as it gradually switches between exploration and exploitation.

The Q-table heat map shows the existence of well-separated but conservative action-value estimates with PSO preferred when the load is medium and high-load, and GA preferred when load is low, which proves the adaptive and load-aware decision-making. The fact that learned Q-values are consistent with heuristic selection frequency and performance metrics at the system level confirms the strength and internal consistency of the proposed ISARSA framework.

2.Performance Evaluation of Proposed ISARSA Framework on Hadoop

The following algorithms of scheduling are applied on the Hadoop platform:

(i) FIFO,

(ii) Fair Scheduler, and

(iii) HSSA (Hyper-Heuristic Scheduling Algorithm) proposed in [33].

YARN has native support of the first two schedulers and HSSA is a sophisticated hyper-heuristic algorithm proposed by Tsai et.al. in [33]. Based on their work, we introduce ISARSA, which includes a distinct learning ability with the use of on-policy reinforcement learning. In order to make a fair and meaningful comparison, the experimental setup is meant to be similar to a configuration reported in[33]. Once the scheduling algorithms are selected, proper benchmark programs and datasets are selected and run on the Hadoop platform to perform the performance. We used test programs as tabulated in table 2:

Table 2: Test programs to be used for comparisons

Programs	Availability
grep program	Standard MapReduce string matching.
Log Analyzer	Standard MapReduce way to log files analysis.
TeraSort	Standard MapReduce sort.

The dataset we plan to use with the programs in table 2 is tabulated in table 3.

Table 3: Dataset used with programs

Programs	**Dataset**
Grep program	Random generation of 100 bytes of data per line.
Log Analyzer	Available online about Worldcup1998. [55]
TeraSort	Random generation of 100 bytes of data per line, but large datasets are used with burst (of size 5GB, 7GB and 10GB)

A Hadoop cluster is configured in MATLAB using four nodes. The map slots of each node are allocated four slots, which result in sixteen map slots per node to execute a task. It has one node known as the master (ResourceManager) that is charged with the responsibility of submitting jobs, coordination of scheduling, and aggregation of results, and the rest are known as worker nodes that handle the execution of map tasks.

Experimental Setup for Hadoop

The hyper-parameter details of various algorithms in the meta-heuristic pool are as follows:

1)For GA	
Properties	**Values/Descriptions**
Population Size	50
Fitness (F) criteria	Eq. 6
Probabilities of crossover p_c	0.8
Probabilities of mutation p_m	0.2
Crossover type	Single-point
Mutation type	Single-point
2)For CS	
Properties	**Values/Descriptions**
Population Size	50
Objective function	Eq. 6
p_a	0.25
3)For PSO	
Properties	**Values/Descriptions**
Inertia weight	0.9
Objective function	Eq. 6
Acceleration coefficient	1.0
The hyper-parameter details for ISARSAis:	
Properties	**Values/Descriptions**
State space S	$\{s_1, s_2, s_3\}$ (low, medium, high load)
Action space	$\{a_1, a_2, a_3\}$ for {GA, CS, PSO}
α	0.75
γ	0.65
$\in$	> 0
policy (π)	Greedy + $\in$ −gereedy

The results of experiments performed on the three benchmark programs are tabulated in table 4

Table 4: Results of comparing ISARSA with FIFO, Fair Scheduler and HSSA.

Algorithm performance (Flow Time in seconds)					
Task		**FIFO**	**Fair Scheduler**	**HSSA**	**ISARSA**

Grep	Average	704	708	707	707
	Best	703	710	703	700
	Worst	705	706	711	705
Log Analyzer	Average	1175	1153	1143	1128
	Best	1150	1147	1137	1122
	Worst	1201	1159	1150	1134
TeraSort	Average	910	830	866	832
	Best	870	860	830	760
	Worst	951	950	901	905

5. Result Analysis

Task scheduling in clouds is a process of maximizing several performance measures, among them being flowtime, resource use, and throughput. This part compares the performance of ISARSA with the traditional schedulers (FIFO, Fair Scheduler and HSSA) and natural-inspired hybrid metaheuristics in different workloads and datasets, with the emphasis on the adaptability in terms of learning and optimization efficiency. The findings demonstrate that ISARSA is always able to achieve lower flowtime in comparison to FIFO, Fair Scheduler and HSSA in all the cases. FIFO will not work because it has a poor scheduling policy which is a static policy, whereas the Fair Scheduler will not improve much because it has a constraint of fairness which will limit optimization.

HSSA experiences heuristic switching and experiences early saturation due to the lack of explicit state action learning. Conversely, ISARSA uses on-policy reinforcement learning to repeatedly improve decisions regarding scheduling, which results in consistent and long-term flowtime reduction

Rather, ISARSA is dynamically adjusted heuristic preference depending on the system load, and so generalizes workloads effectively. The gains are especially significant in the case of the workload of the Log Analyzer. Even though the absolute flowtime difference between advanced algorithms is relatively low-value, i.e. near-optimal performance in real-world cloud constraints, the steady increase in ISARSA indicates that the algorithm gains come not due to heuristics or random chance but due to systematic learning in policies.

6. Conclusions

In this paper, it is proposed to use a better SARSA based scheduling framework (ISARSA) that can be utilized to achieve more stability in learning and reliability in decision-making in the dynamic cloud task scheduling environment. ISARSA adopts on-policy reinforcement learning framework in which the difference in values is directly linked to the scheduled action performed. This close connection between learning and implementation strengthens the trustworthiness of the policies under uncertain conditions and frequently changing conditions of the system.

Simulation shows that ISARSA is always optimal to minimize the flowtime and to maximize the throughput when compared to FIFO, Fair Scheduler and HSSA. Compared to the off-policy Q-learning procedures, ISARSA exhibits smooth convergence of rewards as well as moderate estimates of the action value, meaning that optimistic overestimation is also absent. It is observed that the obtained Q-tables and reward curves demonstrate that ISARSA develops structured, state-aware scheduling policies through gradual instead of quick or intense convergence to stabilize an extremely dynamic cloud environment.

The experiment results with Hadoop also prove the possibility of the ISARSA in the practical implementation environment, including task queuing delays, slot contention, and scheduling overhead. Despite these limitations, the ISARSA has reduced or equivalent flowtime when compared to traditional schedulers and advanced hybrid metaheuristics. This increase in performance is particularly evident in workloads that are computation-intensive such as Log Analyzer where it can be observed that ISARSA can be used to fine-tune heuristic preference based on the workload characteristics rather than pre-defined hybrid strategies.

Overall, ISARSA may be regarded as the powerful evolution of the learning-based cloud scheduling as it is based on on-policy learning and has the awareness of execution. ISARSA improvement does not involve changing the basic update rule of SARSA, but combining it with load-aware state modelling, action selection in heuristic mode and reward structure in line with the aims of the cloud performance. ISARSA balances the learning stability and adaptive decision-making and consequently offers a stable point of intelligent scheduling in dynamically adjusted cloud infrastructures between theoretical reinforcement learning and practical cloud resource management.

7. References

1. Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches," ACM Comput. Surv., vol. 47, no. 4, pp. 1–33, Jul. 2015, https://doi.org/10.1145/2788397 .
2. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Futur. Gener. Comput. Syst., vol. 25, no. 6, pp. 599–616, Jun. 2009, https://doi.org/10.1016/j.future.2008.12.001.
3. R. Buyya, S. K. Garg, and R. N. Calheiros, "SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions," in 2011 International Conference on Cloud and Service Computing, IEEE, Dec. 2011, pp. 1–10. https://doi.org/10.1109/CSC.2011.6138522.
4. R. Aluvalu and L. Muddana, "A Survey on Access Control Models in Cloud Computing," 2015, pp. 653–664. https://doi.org/10.1007/978-3-319-13728-5_73.
5. D.Garey, M.R. and Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st ed. New York: W. H. Freeman & Co, 1979. [Online]. Available: https://dblp.org/rec/books/fm/GareyJ79
6. M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," J. Cloud Comput., vol. 7, no. 1, p. 4, Dec. 2018, https://doi.org/10.1186/s13677-018-0105-8.
7. S. Ghanbari, M. Othman, M. R. Abu Bakar, and W. J. Leong, "Multi-objective method for divisible load scheduling in multi-level tree network," Futur. Gener. Comput. Syst., vol. 54, pp. 132–143, Jan. 2016, https://doi.org/10.1016/j.future.2015.03.015.
8. K. Rajakumari, M. V. Kumar, G. Verma, S. Balu, D. Kumar Sharma, and S. Sengan, "Fuzzy Based Ant Colony Optimization Scheduling in Cloud Computing," Comput. Syst. Sci. Eng., vol. 40, no. 2, pp. 581–592, 2022, https://doi.org/10.32604/csse.2022.019175.
9. S. S. Pabboju and A. Thondepu, "Effective Task Scheduling in Cloud Computing using Improved BAT Algorithm," Int. J. Eng. Trends Technol., vol. 70, no. 10, pp. 271–276, Oct. 2022, https://doi.org/10.14445/22315381/IJETT-V70I10P226.
10. H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," Optik (Stuttg)., vol. 258, p. 168677, May 2022, https://doi.org/10.1016/j.ijleo.2022.168677.
11. N. Alharbe, M. A. Rakrouki, and A. Aljohani, "An Improved Ant Colony Algorithm for Solving a Virtual Machine Placement Problem in a Cloud Computing Environment," IEEE Access, vol. 10, pp. 44869–44880, 2022, https://doi.org/10.1109/ACCESS.2022.3170103.
12. J. He, "Cloud Computing Load Balancing Mechanism Taking into Account Load Balancing Ant Colony Optimization Algorithm," Comput. Intell. Neurosci., vol. 2022, pp. 1–10, Apr. 2022, https://doi.org/10.1155/2022/3120883.
13. L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, "A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing," IEEE Access, vol. 3, pp. 2687–2699, 2015, https://doi.org/10.1109/ACCESS.2015.2508940.
14. A. S. A. Beegom and M. S. Rajasree, "Integer-PSO: a discrete PSO algorithm for task scheduling in cloud computing systems," Evol. Intell., vol. 12, no. 2, pp. 227–239, Jun. 2019, https://doi.org/10.1007/s12065-019-00216-7.
15. M. Kumar and S. C. Sharma, "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint," Sustain. Comput. Informatics Syst., vol. 19, pp. 147–164, Sep. 2018, https://doi.org/10.1016/j.suscom.2018.06.002.
16. S. S and KirankumariPatil, "Hybrid Genetic Algorithm and Modified-Particle Swarm Optimization Algorithm (GA-MPSO) for Predicting Scheduling Virtual Machines in Educational Cloud Platforms," Int. J. Emerg. Technol. Learn., vol. 17, no. 07, pp. 208–225, Apr. 2022, https://doi.org/10.3991/ijet.v17i07.29223.
17. S. Nabi and M. Ahmed, "PSO-RDAL: particle swarm optimization-based resource- and deadline-aware dynamic load balancer for deadline constrained cloud tasks," J. Supercomput., vol. 78, no. 4, pp. 4624–4654, Mar. 2022, https://doi.org/10.1007/s11227-021-04062-2.
18. J. Balusamy and M. Karunakaran, "Hybridization of immune with particle swarm optimization in task scheduling on smart devices," Distrib. Parallel Databases, vol. 40, no. 1, pp. 85–107, Mar. 2022, https://doi.org/10.1007/s10619-021-07337-y.
19. N. E. Nwogbaga, R. Latip, L. S. Affendey, and A. R. A. Rahiman, "Attribute reduction based scheduling algorithm with enhanced hybrid genetic algorithm and particle swarm optimization for optimal device selection," J. Cloud Comput., vol. 11, no. 1, p. 15, Dec. 2022, https://doi.org/10.1186/s13677-022-00288-4.

20. G. Wei, "Quadratic Particle Swarm Optimisation Algorithm for Task Scheduling Based on Cloud Computing Server," J. Inf. Knowl. Manag., vol. 22, no. 01, Feb. 2023, https://doi.org/10.1142/S0219649222500678.
21. X. Xia, H. Qiu, X. Xu, and Y. Zhang, "Multi-objective workflow scheduling based on genetic algorithm in cloud environment," Inf. Sci. (Ny)., vol. 606, pp. 38–59, Aug. 2022, https://doi.org/10.1016/j.ins.2022.05.053.
22. L. Imene, S. Sihem, K. Okba, and B. Mohamed, "A third generation genetic algorithm NSGAIII for task scheduling in cloud computing," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 9, pp. 7515–7529, Oct. 2022, https://doi.org/10.1016/j.jksuci.2022.03.017.
23. S. Chakraborty and K. Mazumdar, "Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 4, pp. 1552–1568, Apr. 2022, https://doi.org/10.1016/j.jksuci.2022.02.014.
24. Y. Xie, Y. Sheng, M. Qiu, and F. Gui, "An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling," Eng. Appl. Artif. Intell., vol. 112, p. 104879, Jun. 2022, https://doi.org/10.1016/j.engappai.2022.104879.
25. H. Materwala, L. Ismail, R. M. Shubair, and R. Buyya, "Energy-SLA-aware genetic algorithm for edge–cloud integrated computation offloading in vehicular networks," Futur. Gener. Comput. Syst., vol. 135, pp. 205–222, Oct. 2022, https://doi.org/10.1016/j.future.2022.04.009.
26. M. Nanjappan and P. Albert, "Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment," Concurr. Comput. Pract. Exp., vol. 34, no. 7, Mar. 2022, doi: 10.1002/cpe.5517.
27. S. Srichandan, T. Ashok Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm," Futur. Comput. Informatics J., vol. 3, no. 2, pp. 210–230, Dec. 2018, https://doi.org/10.1016/j.fcij.2018.03.004.
28. A. GhorbanniaDelavar and Y. Aryan, "HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems," Cluster Comput., vol. 17, no. 1, pp. 129–137, Mar. 2014, https://doi.org/10.1007/s10586-013-0275-6.
29. M. Hosseini Shirvani and R. NoorianTalouki, "A novel hybrid heuristic-based list scheduling algorithm in heterogeneous cloud computing environment for makespan optimization," Parallel Comput., vol. 108, p. 102828, Dec. 2021, https://doi.org/10.1016/j.parco.2021.102828.
30. S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," Futur. Gener. Comput. Syst., vol. 68, pp. 331–345, Mar. 2017, https://doi.org/10.1016/j.future.2016.10.014.
31. M. Kaur, S. Kadam, and N. Hannoon, "RETRACTED ARTICLE: Multi-level parallel scheduling of dependent-tasks using graph-partitioning and hybrid approaches over edge-cloud," Soft Comput., vol. 26, no. 11, pp. 5347–5362, Jun. 2022, https://doi.org/10.1007/s00500-022-07048-1.
32. A. Chhabra, K.-C. Huang, N. Bacanin, and T. A. Rashid, "Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic," J. Supercomput., vol. 78, no. 7, pp. 9121–9183, May 2022, https://doi.org/10.1007/s11227-021-04199-0.
33. C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A Hyper-Heuristic Scheduling Algorithm for Cloud," IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 236–250, Apr. 2014, https://doi.org/10.1109/TCC.2014.2315797.
34. E. N. Alkhanak and S. P. Lee, "A hyper-heuristic cost optimisation approach for Scientific Workflow Scheduling in cloud computing," Futur. Gener. Comput. Syst., vol. 86, pp. 480–506, Sep. 2018, https://doi.org/10.1016/j.future.2018.03.055.
35. A. Sharma, "Nature inspired algorithms with randomized hypercomputational perspective," Inf. Sci. (Ny)., vol. 608, pp. 670–695, Aug. 2022, https://doi.org/10.1016/j.ins.2022.05.020.
36. A. Sharma, "Stochastic nonparallel hyperplane support vector machine for binary classification problems and no-free-lunch theorems," Evol. Intell., vol. 15, no. 1, pp. 215–234, Mar. 2022, https://doi.org/10.1007/s12065-020-00503-8.
37. A. Sharma, "Algorithms simulating natural phenomena and hypercomputation," in 2016 IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS), IEEE, Mar. 2016, pp. 1–6. https://doi.org/10.1109/SCEECS.2016.7509337.
38. M. Hussain, L.-F. Wei, F. Abbas, A. Rehman, M. Ali, and A. Lakhan, "A multi-objective quantum-inspired genetic algorithm for workflow healthcare application scheduling with hard and soft deadline constraints in hybrid clouds," Appl. Soft Comput., vol. 128, p. 109440, Oct. 2022, https://doi.org/10.1016/j.asoc.2022.109440.

39. Z. Tong, X. Deng, H. Chen, J. Mei, and H. Liu, "QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment," Neural Comput. Appl., vol. 32, no. 10, pp. 5553–5570, May 2020, https://doi.org/10.1007/s00521-019-04118-8.
40. D. Fernández-Cerero, J. A. Troyano, A. Jakóbik, and A. Fernández-Montes, "Machine learning regression to boost scheduling performance in hyper-scale cloud-computing data centres," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 6, pp. 3191–3203, Jun. 2022, https://doi.org/10.1016/j.jksuci.2022.04.008.
41. R. N. Calheiros and R. Buyya, "Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 7, pp. 1787–1796, Jul. 2014, https://doi.org/10.1109/TPDS.2013.238.
42. X. Huang, M. Xie, D. An, S. Su, and Z. Zhang, "Task scheduling in cloud computing based on grey wolf optimization with a new encoding mechanism," Parallel Comput., vol. 122, p. 103111, Nov. 2024, https://doi.org/10.1016/j.parco.2024.103111.
43. P. J. Varma and S. R. Bendi, "Multiobjective hybrid Al-Biruni Earth Namib Beetle Optimization and deep learning based task scheduling in cloud computing," Sustain. Comput. Informatics Syst., vol. 44, p. 101053, Dec. 2024, https://doi.org/10.1016/j.suscom.2024.101053.
44. P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," J. Ambient Intell. Humaniz. Comput., vol. 14, no. 4, pp. 4313–4327, Apr. 2023, https://doi.org/10.1007/s12652-023-04541-9.
45. I. Behera and S. Sobhanayak, "HTSA: A novel hybrid task scheduling algorithm for heterogeneous cloud computing environment," Simul. Model. Pract. Theory, vol. 137, p. 103014, Dec. 2024, https://doi.org/10.1016/j.simpat.2024.103014.
46. L. Abualigah et al., "Improved synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing," Sustain. Comput. Informatics Syst., vol. 43, p. 101012, Sep. 2024, https://doi.org/10.1016/j.suscom.2024.101012.
47. T. Zaki, Y. Zeiträg, R. Neves, and J. R. Figueira, "A cooperative coevolutionary genetic programming hyper-heuristic for multi-objective makespan and cost optimization in cloud workflow scheduling," Comput. Oper. Res., vol. 172, p. 106805, Dec. 2024, https://doi.org/10.1016/j.cor.2024.106805.
48. X.-S. Yang, "Nature-inspired optimization algorithms: Challenges and open problems," J. Comput. Sci., vol. 46, p. 101104, Oct. 2020, https://doi.org/10.1016/j.jocs.2020.101104.
49. X.-S. Yang, Nature-Inspired Optimization Algorithms. Elsevier, 2014. https://doi.org/10.1016/C2013-0-01368-0.
50. X.-S. Yang, S. Deb, S. Fong, X. He, and Y.-X. Zhao, "From Swarm Intelligence to Metaheuristics: Nature-Inspired Optimization Algorithms," Computer (Long. Beach. Calif)., vol. 49, no. 9, pp. 52–59, Sep. 2016, https://doi.org/10.1109/MC.2016.292.
51. R. P. . Rao, "Reinforcement Learning: An Introduction; R.S. Sutton, A.G. Barto (Eds.); MIT Press, Cambridge, MA, 1998, 380 pages, ISBN 0-262-19398-1, $42.00," Neural Networks, vol. 13, no. 1, pp. 133–135, Jan. 2000, https://doi.org/10.1016/S0893-6080(99)00098-2.
52. A. L. C. Ottoni, E. G. Nepomuceno, M. S. de Oliveira, and D. C. R. de Oliveira, "Tuning of reinforcement learning parameters applied to SOP using the Scott–Knott method," Soft Comput., vol. 24, no. 6, pp. 4441–4453, Mar. 2020, https://doi.org/10.1007/s00500-019-04206-w.
53. V. A, Nature-Inspired Optimization Algorithms, 1st ed. Chapman & Hall, 2022. [Online]. Available: https://www.routledge.com/Nature-Inspired-Optimization-Algorithms/A/p/book/9780367503291?srsltid=AfmBOooyP0H-xuzkXvfSmzGQ8VayL2lOCqNGYsLJr-1_BxsyhFAxVJNH
54. J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.
55. World Cup '98 Web Site Logs, The Internet Traffic Archive, Lawrence Berkeley National Laboratory. [Online]. Available: https://ita.ee.lbl.gov/html/contrib/WorldCup.html.