International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

OPTIMIZATION OF CLOUD SCHEDULING USING Q-LEARNING HYPERHEURISTIC ALGORITHM

Arvind Upadhyay[a,b,*], Ramesh Thakur[c], Archana Thakur[d],Kavita Upadhyay[e]
[a]IPS Academy, Institute of Engineering & Science, Indore, M.P., India
[b]Institute of Engineering and Technology, Devi Ahilya University, Indore, M.P., India
[c]International Institute of Professional Studies, Devi Ahilya University, Indore, M.P., India
[d]School of Computer Science and IT, Devi Ahilya University, Indore, M.P., India
[e]IPS Academy, Institute of Engineering & Science, Indore, M.P., India

*arvindupadhyay@ipsacademy.org
[c]r_thakur@rediffmail.com
[d]archana227@gmail.com
[e]Kavita.thorat@gmail.com

Abstract

The challenge of producing an optimal task schedule in cloud environments holds significant importance for both the research community and industry. This scheduling problem is computationally complex and is generally classified as NP-complete, and in many cases, NP-hard. Heuristic and metaheuristic algorithms are used to solve optimization problems. Heuristic algorithms use the process of trial and error to find the answer. Metaheuristic algorithms, on the other hand, identify the answer at a higher level. Therefore, we have proposed a better solution to this problem using Reinforcement Learning based Q-learning algorithm which is extended to act as a hyper-heuristic controlling and helping various evolutionary algorithms which are acting as meta-heuristics. It is hypothesized that this strategy will provide us with a near optimal schedule for a real cloud environment. That is, concretely we propose a hyper-heuristic based on Q-learning which drives various evolutionary algorithms HSA, CS, BAT algorithm and as a result the system outputs a schedule which requires as minimum time as practically possible when executed on a real cloud. We test the proposed algorithm on a real world system Hadoop directly for reliable results. For comparisons we use the state of art scheduling algorithms (deterministic, hyper-heuristic and meta-heuristics) and our results demonstrates a significant performance gain over 80% of comparisons.

Keywords:- Optimization ,Cloud Computing, Meta-heuristics, Hyper-heuristic, HSA, CS, BAT

1. Introduction

Information in today's world has become an issue not only because of its production but also because of its usage. The classical centralized resources of computing equipments which are backbone of many successful innovations have now become a kind of limitations in this highly

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
incendiary situation of information explosion. There is an urgent need of decentralized computing resources and this demand is fulfilled by cloud environment [1]. Cloud computing (as software as a service, on-demand computing, or the Internet as a platform) is simply a collection of computing resources that are made available to end users based on their changing needs. [1–3]. This definition gives raise to various models of cloud computation [4].

Now-a-days, as cloud has become extremely popular and useful fundamental computing environment, but cloud computing has its own issues and problems. The most fundamental problem which arises is of *"task scheduling"* [5,62]. Tasks are the fundamental unit of work in cloud which is assigned to the various available machines. The problem of task scheduling basically "demands a sequence of tasks which can be given to the cloud such that some predefined conditions are met or in scientific terms optimized". This is in fact an optimization problem where predefined condition(s) can be any of the concerned condition(s) and can wary dynamically with demand [6].

Moreover, the cloud scheduling problem is NP-Complete [21], meaning that no algorithm can accomplish this task in polynomial time. Every algorithm reported in literature, has to evaluate each and every candidate solution if one has to guarantee an optimal solution. This property of NP-completeness of the scheduling problem motivates us to design an algorithm which will be as close to optimal as possible.

Presently, to solve the scheduling problem specifically and NP-complete problems generally, the following solutions are explored in the literature. First considering heuristics which are used extensively to come up with *good solutions fast* are applied to our problem of cloud scheduling [7, 23, 24]. Moreover, the second option meta-heuristics; defined to be more general than heuristics and are problem independent, are used to perform better on a selected performance measure (e.g. resources, time etc.) by cloud experts to solve the scheduling problem. To mentioning a few successful meta-heuristics for solving various NP-complete problems reported in literature are Bat Algorithm (BA) [8, 9, 31-35], Particle Swarm Optimization [36-40], Whale Optimization Algorithm [10-17], Cuckoo Search Algorithm [18-20], Cat Swarm Optimization Algorithm [24-30] etc. To achieve even better performance, various hybrid meta-heuristics are developed in e.g. [41-45].

However, with these advanced scheduling techniques there is still an urgent need and high urge of solving the scheduling (NP-complete) problem in a more robust, performance-centric and timer efficient manner. This further motivates us. To achieve these goal researchers had developed even more advanced algorithms which can take many advantages of the important properties of many other algorithms simultaneously. These algorithms are hyper-heuristics. There are a limited number of hyper-heuristics proposed in the literature for e.g. [5, 46, 47]. Although, Hyper-heuristics are recent research methodologies and much remains to contribute to these methodologies from many perspectives, they are very promising.

Therefore, motivated by the above facts, here in this paper we provide the following novel contributions:

(1) Solution to the cloud scheduling problem which is NP-Complete (first motivation).

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

(2) As a solution we presented a hyper-heuristic.
(3) A learning strategy is introduced in our proposed hyper-heuristic.
(4) We select our meta-heuristics to be employed very carefully to get the maximum advantages
(5) The proposed hyper-heuristic is able to take advantages of meta-heuristics quite efficiently.

We implement our proposed algorithm and all of the algorithms used for comparisons on the real system *HADOOP* avoiding simulations software completely. This will make our results directly applicable to the real cloud.

The organization of this paper is as follows: section 2 describes the background required to understand hyper-heuristics and other used algorithms. Section 3 describes our proposed learning algorithm while section 4 gives the implementation details and results of the performed experimentation. Section 5 analyzes the results obtained in section 4, while section 6 concludes the work with future directions.

2. Background

In this section, we describe the hyper-heuristics methodology in general, Q-learning, Harmonic Search Algorithm, Cuckoo Search and Novel Bat algorithm. In simple words here we are interested in Nature Inspired Algorithms (NIAs) whose definition is quite informal in the literature, but see [14], which are becoming very popular recently.

2.1. Hyper-heuristics

Figure 1 demonstrates the working of a hyper-heuristics in general. Terminologies are defining as follow:

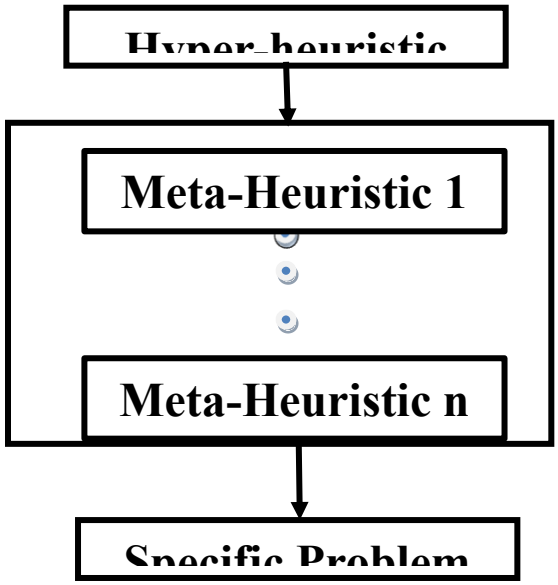

Figure: 1 A general overview of a Hyper-heuristic algorithm

1) Hyper-heuristic: It is essentially an algorithm which works on other working algorithms with the aim of creating a novel problem specific algorithm or selecting any one of the working algorithm at a time for specific problem with respect to some performance measure (e.g. time).

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2) Meta-heuristic: It is essentially an algorithm which works on the specific problem directly. The most important property is it is problem independent. (In our Hyper-heuristic environment typically, there are n numbers of meta-heuristic available (collectively called a pool) on which a hyper-heuristic works for example.)

3) Specific problems: As implied by their name a problem is just a mathematical formulation of the concerned situation. Hyper-heuristics are generally suitable for "hard" problems. (here, we are interested in the problem of cloud scheduling)

Figure 1, demonstrates what we need to design a hyper-heuristic algorithm. In General, our idea is:-

1) To modify the Q-learning algorithm and then employ it as a hyper-heuristic.
2) The Harmony Search Algorithm, Cuckoo Search and Bat algorithm are used as meta-heuristic.
3) The cloud scheduling is the specific problem to solve.

Hyper-heuristic will work on meta-heuristic to generate a problem specific heuristic which will then works on the cloud scheduling problem to generate a schedule (a sequence) of tasks to be executed in order in cloud environment.

2.1. Q-learning

Reinforcement Learning (RL) [51] provides a way that can be used to get closer to the way how human solve some problem of interest. One concept that is novel to RL is temporal difference (TD) learning whose one incarnation is the Q-learning algorithm. This algorithm is attributed to the impetus gained by RL methods in terms of mathematical analysis (of convergence).

This algorithm is described as:

Algorithm-I (Q-learning)

1: Initialize $Q(s,a) \leftarrow 0$ for all $s \in S$ and $a \in A$.
2: While (s is not terminal)
3: Initialize environment to provide s.
4: Choose a from s using π while ties must be broke randomly.
5: Take action a and observe r and s'.
6: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \arg\max_{a_s \in A_t} Q(s', a_t) - Q(s,a)]$.
7: $s \leftarrow s'$.

Here, S is the set of all possible states and A is the set of all possible actions and s, a is such that $s \in S$ and $a \in A$. Now, consider the update rule:

$$\begin{pmatrix} Q_{t+1}(s_t,a_t) \\ \text{new Q value} \end{pmatrix} \leftarrow \begin{pmatrix} Q_t(s_t,a_t) \\ \text{current Q value} \end{pmatrix} + \begin{pmatrix} \alpha_t \\ \text{learning rate used to balance exploration and exploitation} \end{pmatrix} \left[\begin{pmatrix} r_{t+1} \\ \text{reward} \end{pmatrix} + \begin{pmatrix} \gamma \\ \text{discount} \end{pmatrix} \begin{pmatrix} \arg\max_{a_s \in A_t} Q_t(s_{t+1},a) \\ \text{estimate of (optimal)future value} \end{pmatrix} - \begin{pmatrix} Q_t(s_t,a_t) \\ \text{current value} \end{pmatrix} \right] \quad (1)$$

For a given state this rule chooses the best action by implementing a simple policy. $Q(.)$ is the action-value function (providing the expected return for any action). Moreover, one must overwrite the estimate iteratively, i.e. it is an online update rule. But, the estimate or guess is just a guess,

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
anyway, and the algorithm must know how accurate it is with respect to what actually happened or with reality. This situation is modeled mathematically by considering everything in square bracket ([.]) after α. This also implements temporal-difference one-step look-ahead in which the algorithm is allowed to guess one step forward and decide on the best action. This is one of the biggest advantages as the agent can create a policy as oracle i.e. as predicting future.

The $arg\max_{a_s \in A_t}(.)$, is implemented in the algorithm so as commending it to scene over all possible actions in some state and choose the best possible. Although, clear from algorithm-I the following explanations are just included for rituals.

(1) The algorithm starts with a policy (e.g. greedy) which provides the probability of which action to take.
(2) Then, a data structure (we use a map) is initialized to store action-value pair estimate.
(3) The algorithm iterates over all episodes (each step of each episode), by initializing the starting state.
(4) A decision on an action is done through the external policy function, while the chosen action is that of highest probability.
(5) Then, the action chosen is taken and the reward and next state are both observed.
(6) The current estimate (action-value) is updated as the new behavior is observed in the given state and action.
(7) The next state is set finally. This process is iterated again until the environment signals the state is terminal.

Now, it can be seen that Q-learning is advantageous in those situations which are highly dynamic and work on a black box function. We have formulated a version of Q-learning as a hyper-heuristic controlling various meta-heuristics so that their different advantages are unified.

2.2. Harmony Search Algorithm (HSA)

HSA [48] is a meta-heuristic inspired by the harmony i.e. the process of joining individual sounds so as to form a composition. To create a new composition, from a musician's point of view these three processes are important:

1) Play something already known to be in harmony.
2) Play something (not exactly but) around a known music.
3) Compose something new.

Formalizing the above concepts, we get the following correspondence:

1) usage of harmony memory,
2) pitch adjusting, and
3) Randomization.

These are defined-respectively as:

1) The harmony accepting rate $r_{accept} \in [0,1]$ describes formally how harmony memory is used. If too high then only known good harmonics are selected and if too low then best harmonics are explored less. Typically $r_{accept} = 0.7 \sim 0.95$ is used.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2) Pitch adjusting is typically a random walk and in fact it corresponds to a local search performed as $(x\ is\ a\ solution)$: $x_{new} = x_{old} + b_p(2 * rand - 1)$,
where $rand$ is a random number drawn from the uniform distribution $[0, 1]$ and b_p is the bandwidth which controls the local range of the pitch adjustment (r_{pa} is the pitch adjustment rate typically equals 0.1 ~ 0.5 to control the degree of adjustment).
3) Randomization is the global search of HSA and its true probability is $P_{random} = 1 - r_{accept}$.
Equipped with above facts the HSA can be outlined as:

Algorithm-II (HSA)

1: Define the objective function.
2: Generate the initial solutions (harmony).
3: Define pitch adjusting rate and pitch limit.

4: Define r_{accept}.
5: Perform initial evaluation.
6:While ($Number\_of\_iterations$ is less than N)
7: New harmonics generation (with respect to best harmonics).
8: Adjust pitch.
9: if (rand > r_{accept}).
10: choose an existing harmonic randomly.
11: else if (rand > r_{pa}).
12: adjust pitch randomly.
13: else.
14: generate new harmony by randomization.
15: End if.
16: Accept the new harmony.
17: End while.
18: Find the current best.
19: Output.

2.3. Cuckoo Search (CS)

The Cuckoo bird's mating behavior served as inspiration for Cuckoo Search [49]. The said habit of the cuckoo bird can be idealized as follows:

(1) An idealized environment is being supposed.
(2) The idealized habitat has a set number of cuckoos, each of which can lay a single egg at a time and neatly place that egg into a nest that is picked at random.
(3) Best nest or eggs survive to form the next generations.
(4) There are a finite number of predefined host birds which are the owners of the nests and can discover the terrorizing cuckoo's egg with probability $p \in (0,1)$. In this case the nest/egg can be abandoned.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
The algorithm does not distinguish between the terms "cuckoo," "nest," "egg," and "solution." The CS is as defined in Algorithm-III:

Algorithm-III (CS)

1: Articulate the objective function.
2: Create the first group of options.
3:While (*Number_of_iterations* is less than *N*)
4: Get a solution randomly.
5: Levy flight should be used to generate a solution.
6: Evaluate it for its objective value (say f_i).
7: Choose a solution randomly (say j and let its objective value is f_j).
8: if ($f_i < f_j$)
9: replace j by i.
10: end
11: Abandon worst nests.
12: Generate new nests.
13: preserve best solutions.
14: Find the current best by ranking the solutions.
15: End while.
16: Output.

To describe the mathematical terms let, in the following x_j^t and x_k^t are two distinct solutions selected randomly, H is a Heaviside function, ε is a small random number (from uniform distribution), s is the step size, $\otimes$ is the entry-wise product of two vectors and α is the scaling factor.

Local Search: It is performed using the following:

$$x_i^{t+1} = x_i^t + \alpha * s \otimes H(p_a - \varepsilon) \otimes \left(x_j^t - x_k^t\right). \quad (2)$$

Global search: It is performed using the following (*Levy flight*):

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda). \quad (3)$$

Here,

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin\left(\pi/2\right)}{\pi} \frac{1}{s^{1+\lambda}}. \quad (4)$$

This completes our second meta-heuristic.

2.4. Bat Algorithm (BA)

The bat algorithm [49] is inspired by the bat behavior i.e. their echolocation behavior, and therefore it is also a meta-heuristic. Bats use this behavior to search for their prey. For the purpose of the algorithm three idealized rules are extracted and described as follow:

1) All the bats use echolocation for forging.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2) Bats fly randomly with specific velocity v_i at specific location x_i. The sound they emit has frequency $\boldsymbol{f_i}$ (or wavelength $\frac{1}{\boldsymbol{f_i}}$) which they can adjust. The rate of emission is $\boldsymbol{r_i} \in [0,1]$.
3) The loudness of their sound pluses can also vary from low $A_{\boldsymbol{min}}$ to high $A_{\boldsymbol{max}}$.

Then, converting the above three rules into an algorithm:

Algorithm-IV (BA)

1: Define the objective function.
2: Initialize the bat population, x_i, v_i, f_i, r_i and A_i i for i^{th} bat.
3: Evaluate the objective.
4:While ($\boldsymbol{Number\_of\_iterations}$ is less than $\boldsymbol{N}$)
5: generate new solutions (eqs. 5-7) by adjusting frequency.
6: for each solution update velocity and location.
7: if (rand > $\boldsymbol{r_i}$)
8: From the available options, choose the best solution.
9: perform the local search around it (eq. 8).
10: End if.
11: Perform a global search by random flight.
12: if (rand < $\boldsymbol{A_i}$ & $\boldsymbol{f(x_i) < f(x_*)}$)
13: Accept the new solutions.
14: Increase $\boldsymbol{r_i}$ and reduce $\boldsymbol{A_i}$ (eqs. 9 & 10).
15: End if.
16: Rank the bat and find the current best $\boldsymbol{x_*}$
17: End while
18: Output.

The new solutions are produced using (at iteration $t+1$ for the i^{th} bat):

(1) $f_i = f_{min} + (f_{max} - f_{min})\beta$, where $\beta \in [0,1]$ is a random vector. (5)

(2) $v_i^{t+1} = v_i^t + (x_i^t - \boldsymbol{x}_*)f_i$, $\boldsymbol{x}_*$ is the iteration best and v_i is the velocty (6)

(3) $x_i^{t+1} = x_i^t + v_i^{t+1}$. Here x_i is the position (7)

(4) Local search is performed using:

$$x_{new} = x_{old} + s_f \lambda_t A_{mean}^t \quad (8)$$

Here, $\lambda_t \in$ normal distribution $N(0,1)$,
$s_f =$ scaling factor, and
$A_{mean}^t =$ average loudness.

(5) Loudness update formula:

$$A_i^{t+1} = \gamma A_i^t, \text{ here } \gamma \text{ is a constant.} \quad (9)$$

(6) Pulse rate update formula:

$$r_i^{t+1} = r_i^0[1 - \exp(-\alpha t)], \text{ here } \alpha \text{ is a constant.} \quad (10)$$

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
All of the meta-heurustics HSA, CS and BA have distinct advantages and are sutabale for different kinds of problem in general. To get the benefit of all of these algorithms, we will propose a method that have the ability to combine the advantages in a interesting way.

3. Proposed algorithm

The proposed algorithm is constructed incrementally as follows:

3.1. Problem Definition

Let there is n number of tasks in the set $task = \{T_1, T_2, \dots, T_n\}$ (the numbering of tasks here is arbitrary for e.g. on the first come basis), then the problem of scheduling these tasks on cloud is to find a *sequence* of tasks such that the concerned *performance metric* is maximized or minimized. There are many kinds of performance metric, a few of which are:

Makespan: It denotes the final task's completion time, mathematically:

$$makespan = max_{i \in task}\{F_i\} \tag{11}$$
$$F_i = \text{final task's completion time.}$$

(1) ***Flowtime:*** It is the sum of the finishing time of all of the tasks, mathematically:

$$\text{Flowtime} = \sum_{i \in task} F_i \tag{12}$$
$$F_i = \text{finishing time of task } i.$$

By choosing the performance metric as Flowtime the cloud scheduling problem is to minimize this metric i.e. we are concerned with finding a schedule (sequence) of given tasks $\{T_1, T_2, \dots, T_n\}$ such that the sum of the finishing time of all of the tasks is minimum. Mathematically,

$$\min_{s} \quad \sum_i F_i \tag{13}$$

where $s = \{T_i |\ 1 \leq i \geq n\}$ is the generated sequence of tasks by the scheduling algorithm. This (e.q. (13)) forms our problem definition.

3.2 Meta-heuristics

As defined previously meta-heuristics are themselves methodology that is independent of any underlying problem. But, there are some major issues with them:

(1) There are many meta-heuristic algorithms available in literature, and it is difficult to find the perfect algorithm for any given problem.
(2) Can we provably achieve the optimal solution in polynomial time? This question has no answer, considering the cloud scheduling to be NP-complete [21].

Meta-heuristics are very promising technique which practically (not theoretically) proven to be very effective towards solving NP-complete problems. The following established meta-heuristics is being used to work with in this research.

(1) HSA
(2) CS
(3) BA

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The function they optimize is eq. (13). The algorithms (HSA, CS and BA) minimizes the cumulative Flowtime, given a set of n tasks by finding a sequence $s = \{T_i | \ 1 \leq i \geq n\}$ as defined before. In detail, for all HSA, CS and BA:

(1) The objective function is eq.(13).

(2) Although it is obvious, but the word solution means a sequence $s = \{T_i | \ 1 \leq i \geq n\}$.

(3) The other concert settings are defined in tables 1 respectively.

All these meta-heuristics are coordinated by our proposed hyper-heuristic that we describe next.

3.3 Modified Q-Learning Hyper-heuristic (MQLH)

Previously we had described major issues with meta-heuristics here we further enumerate the challenges with them which further motivates us for our proposed algorithm:

(1) There is no mathematical theory of meta-heuristics.

(2) Every meta-heuristic has its own advantages and disadvantages, but on which function some meta-heuristic is advantageous is far from obvious.

(3) In essence meta-heuristics are same considering all of the functions [56].

(4) There is no procedure to choose meta-heuristic from a pool such that they can be advantageous for someone's problem's purpose.

Considering the above challenges, we take the refuge to the reinforcement learning paradigm which can be very helpful here. Flow of this work is as follows (see figure 1):

(1) The Q-learning is used as a representative of RL which will work as hyper-heuristic itself. To achieve this objective, modifications will be done in Q-learning. Armed with these modifications this hyper-heuristic will co-ordinate the chosen meta-heuristics.

(2) HSA, CS and BA are meta-heuristics used in this work.

(3) Meta-heuristics directly works on the tasks that are needed to be scheduled on cloud environment.

(4) As a result, it is hypothesized that we achieved not only get a near optimal schedule but also near optimal use of selected meta-heuristics.

Figure 1 can be elaborated in more details in figure 2. Out of the four steps outlined we have already demonstrated (2) and (3) in previous sections formally while the (4) one will be validated by experiments. We explain the (1) now:

Consider algorithm-I which demonstrates Q-learning. In order to apply it as hyper-heuristic to control our meta-heuristics (HSA, CS and BA) so that they can work on the problem of cloud scheduling (eq. (13)) it needs to be modified. We propose the following modifications:

(i) Let $S = \{s_1, s_2, s_3\}$ and $A = \{a_1, a_2, a_3\}$. The subscript 3 is used because we have 3 meta-heuristics algorithms {HSA,CS,BA} and states will be s1=low load, s2=medium load, s3=high load.

α_t is the learning rate defined as $1 - \left(0.9 * \frac{t}{t_{max}}\right)$ where t denotes the current iteration and t_max represents the maximum number of iterations. (14)

(ii) The reward function is defined as $r_f = \begin{cases} +1 \ if \ f_{new} > f_{best} \\ 0 \ if \ f_{new} = f_{best} \\ -1 \ if \ f_{new} < f_{best} \end{cases}$. (15)

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
Here, $f_{new}\ and\ f_{best}()$ are the new reward and the best reward (objective function e.q. (13) values) ever observed.

(iii) The $greedy$ and the $\in -greedy$ polics are implemented as follows:

(1) $Greedy$ policy chooses the best reward ($+1\ value\ for\ the\ reward\ function$) , i.e. $\arg\max_k \mathrm{Q}(\mathrm{s}, \mathrm{a_k})$ where $k = 1,2,3.$ (16)

(2) $\in -greedy$ consist of two mechanism:

$\in$% probability for selecting actions where the index of the selected actions computed as $k = \lceil 3 * rand(\,) \rceil$, $rand(\,)$ returns a random number between 0 and 1. (17)

If $\rho\%$ (1-$\in$= ρ) probability exists then the second mechanism is invoked which chooses the k^{th} action based on roulette-wheel selection while the probability to select the k^{th} action is assigned according to the function:

$$probeblity\ (\mathrm{a_k}) = \frac{exp\ (\mathrm{Q(s,a_k)})}{\sum exp\ (\mathrm{Q(s,a_k)})}. \quad (18)$$

Following is the Modified Q-Learning Hyper-heuristic (MQLH) algorithm:

Algorithm-v (Modified Q-Learning Hyper-heuristic (MQLH)

1: Input: S = $\{\mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3}\}$ and A = $\{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\}$, Type equation here.
2: Initialize Q − table Q(s, a)← 0 for all s ∈ S and a ∈ A a, then set initial state randomly.
3: While (s is not terminal)
4: if rand () ≤ ∈.
5: $\mathbf{a_t \leftarrow a_k}$**: 1 ≤ k ≥ 3 is computed using eq. 17.**
6: else if rand () ≤ ∈ + **ρ**
7: $\mathbf{a_t \leftarrow a_k}$**: 1 ≤ k ≥ 3 is computed using eq. 18.**
8: else
9: $\mathbf{a_t \leftarrow a_k}$**: 1 ≤ k ≥ 3 is computed using eq. 16.**
10: End if
11: if $\mathbf{a_t = 1}$
12: HSA ( ) {algorithm-ii}
13: else if $\mathbf{a_t = 2}$
14: CS ( ) {algorithm-iii}
15: else
16: BA ( ) {algorithm-iv}.
17: Obtain reward r.
18: Obtain max Q factor.
19: update Q($\mathbf{s_t}$, $\mathbf{a_t}$).

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
20: $s_t \leftarrow s_{t+1}$.
21: End while.
22: Output Q-table.

The flow of algorithm V can be understood on a top level by looking at the figure 2 (compare it with figure 1).

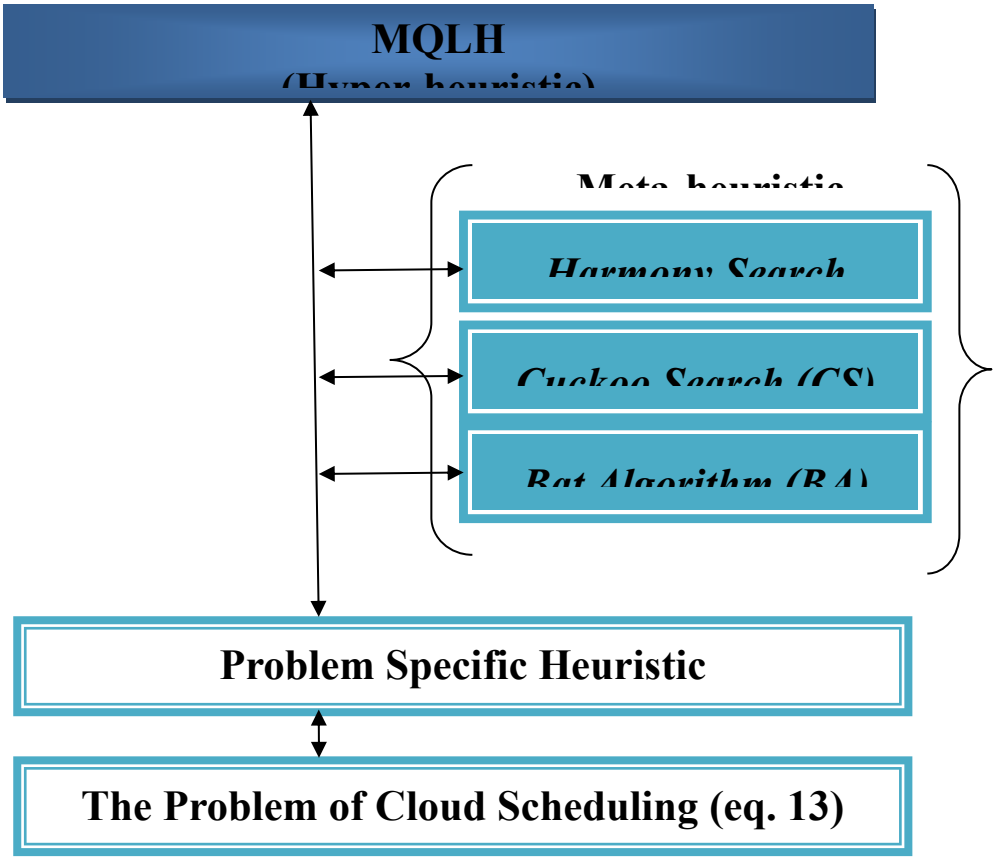

Figure 2: Pictorial representation of our proposed (Hyper-heuristic) methodology.

Figure 3 illustrates the solution structure for meta-heuristic HSA/CS/BA. Although, it is understood from subsection 3.1 we can first define an arbitrary numbering of tasks as 1, 2…, n which are then to be scheduled such that the flowtime is minimized (see problem definition).

Figure 3: Solution structure for HSA/CS/BA (Meta-heuristic).

Our proposed methodology generates cloud scheduling specific new heuristics such that it combines the advantages of the HSA, CS and BA, which make it different from others.
This becomes feasible because, after each time step, the MQLH framework gains complete control over the optimization process. As a result, it can dynamically switch from one meta-heuristic (e.g.,

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
BA) to another (e.g., CS) even during the ongoing optimization, without interrupting the workflow. This newly generated policy is highly problem specific and is able to handle the problem of cloud scheduling which is NP-complete in unique and quite effective way.

4. Implementation and Results

To create a real cloud environment, we configure a Hadoop cluster using MATLAB [52]. Here we'll go over a few benefits of Apache Hadoop, a platform for distributed computing that allows for the storing and processing of large datasets over a cluster of machines. Ranging from a single computer to thousands of computer Hadoop in this scalable way provides storage and processing of petabytes of data, where each computer in the cluster provides a local storage. The basic idea of Hadoop is simple, a user submits a job to the Hadoop cluster which is then divided in tasks of a predefined size and then run in parallel on local computers. Obviously, then computers in the cluster are either identical or nonidentical (which is the most general setup) with respect to each other.

Equipped with all these advantages Hadoop is the framework of choice [53] for:

(1) Machine learning
(2) Big data
(3) Vertical industries
(4) Cloud computing

We are interested in the $(4)^{th}$ advantage of Hadoop which is used by all major giants Google, AOL, IBM, Ebay, Amazon, Yahoo etc. In essence, Hadoop framework provides an excellent opportunity to create cloud which is used by all major IT giants.

More microscopically, in a Hadoop cluster (figure 4) a input job is divided into independent Map-tasks which have predefined (Map-)size of slots available and also a number of (reduce–task-)slots are provided as shown in figure 4. The most important point is: the size of each map-task is the only thing known to algorithms while the size of each reduce-task is not known. Therefore, only map task scheduling is applicable which is the problem of interest which we are going to solve here. The mapping between Hadoop and Cloud can be simply considered as: in Hadoop the slots can be considered as the virtual machines in the cloud environment. There we are faced with the scheduling of tasks actually with our settings *map-tasks*.

The data we are inserting as jobs to our system is generated as follows.

1) We use Matlab script to generate string of size 100 bytes by creating Tall arrays. (there are really many essentially same ways to do it in MATLAB). We intend to create tall (MATLAB) arrays.
2) The size and input period is generated randomly (there are many ways to do it we follow [5]).
3) All the technicalities of Hadoop are handled by MATLAB itself as described in [52].

Matlab on a single system and on a system of 10 connected computers is easy to setup and even a single setup provides multiple slots. All of the algorithms (proposed one and compared ones) are implemented on MATLAB (2020), networking and Hadoop functionalities are created and handled by MATLAB as described in [52].

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

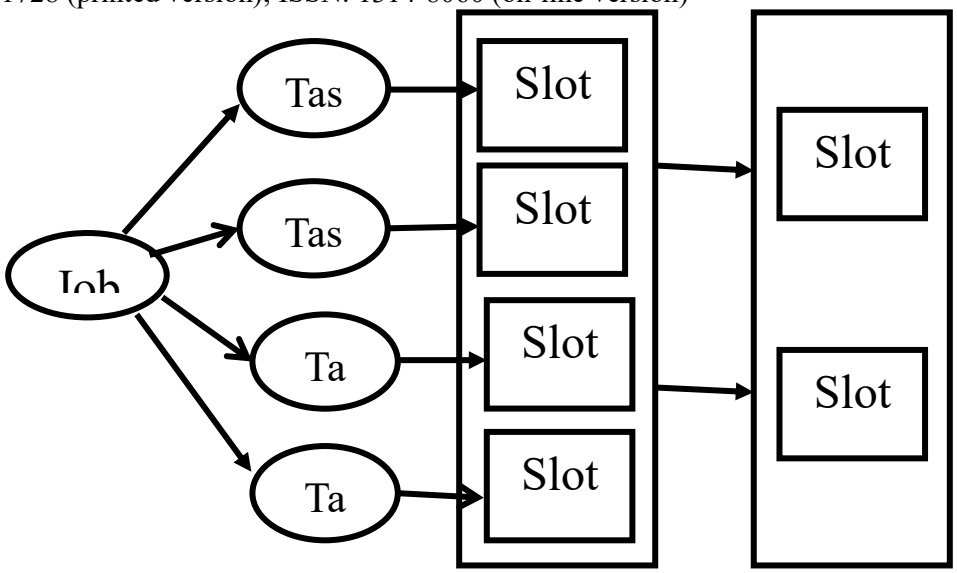

Figure 4: Demonstration of a job execution on Hadoop.

For the comparisons purpose we have (with Hadoop):

(i) FIFO,

(ii) Fair scheduler.

Moreover, we have implemented HSSA (hyper-heuristic scheduling algorithm) proposed in [5]. The hyper-parameters of HSSA are exactly as defined in [5, section V-B].

The (hyper-) parameters details of various algorithms that are used as meta-heuristics in our proposed MQLH are as follows:

Table 1: Concrete hyper-parameter settings for different meta-heuristics.

(1) For HSA

Properties	Values/Descriptions
Population Size	20
Fitness (F) criteria	eq. (13)
r_{accept}	0.85
r_{pa}	0.4
P_{random}	$1 - r_{accept}$

(2) For CS

Properties	Values/Descriptions
Population Size	20
Objective function	$eq.(13)$
p_a	0.25

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

(3) For BA

Properties	Values/Descriptions
Population Size	20
Objective function	$eq.(13)$
A_i^0	1.0
r_i^0	1.0
f_{max}	2
f_{min}	0
α	0.1.

We use the following programs to evaluate the performance of MQLH**:**

1) Grep program available with map/reduce (with standard meaning). Dataset consist of (as described previously) string of size 100 bytes.
2) Map/reduce web log analyzer, which is used to calculate the reference count of URLs. For this purpose, following [5] the data used is from [54]. Complete description of the dataset is also available there.
3) TeraSort also available with Hadoop (generate and sort a large dataset).

The results of experiments performed on the three benchmark programs are tabulated in table 2.

Table 2: Results of comparing MQLH with state of art traditional and hyper-heuristic algorithm.

Task		**Algorithm (*performance in sec*)**			
		FIFO	Fair Scheduler	HSSA	MQLH
Grep	Average	837	845	850	812
	Best	821	825	805	800
	Worst	856	865	898	825
Log Analyzer	Average	1173	1141	1148	1112
	Best	1101	1127	1134	1090
	Worst	1245	1155	1165	1137
TeraSort	Average	1067	948	860	850
	Best	980	910	800	790
	Worst	1055	988	920	910

Table 2 essentially says that, for example, for the job Grep the average execution time (in second) on Hadoop taken by the schedule generated by FIFO algorithm is 837, by Fair Scheduler is 845,

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
by HSSA is 850 and by our MQLH is 812. The best and worst case can be understood similarly for each job while Table 3 can be read similarly.

Moreover, we go a step ahead to compare proposed MQLH with the state of the art meta-heuristics used to schedule (VM in) clouds. We choose

1. BFO [50, 61] (Bacterial Forge Algorithm),
2. PSO [57, 58] (Particle Swarm Optimization),
3. GA [49] (Genetic Algorithm),
4. CS,
5. BA and HSA.

It also includes our selected meta-heuristics CS, BA and HSA. (MQLH performance is evaluated for its second time application). Before we perform the experiments it will be quite instructive to see the common methodology of meta-heuristic cloud scheduling. It will not only make clear how different meta-heuristics used to schedule cloud but also helps us to demonstrate MQLH advantage. Consider figure 1 again and compare it with figure 5 below.

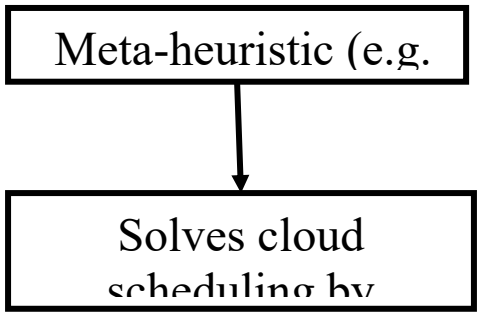

Figure 5: working of a meta-heuristic on cloud scheduling problem.

In figure 5 it is shown pictorially how any meta-heuristic (we choose BFO, PSO, GA, CS, BA and HSA) algorithm solves the problem of scheduling in cloud. Meta-heuristic cloud scheduling can be understood essentially as the optimization of the function eq. 13. All the details are just the same as described for meta-heuristic pool. To distinguish the proposed hyper-heuristic approach from conventional meta-heuristic methods, consider the following points:

(1) A wide range of meta-heuristic algorithms exist in the literature, each offering its own strengths and advantages [55].

(2) However, every meta-heuristic also suffers from specific situational limitations that restrict its applicability [55].

(3) Determining which algorithm is universally optimal for a given problem remains an open and unresolved question.

(4) Even when attempts are made to answer this question, no single meta-heuristic consistently performs best across all scenarios, as indicated by studies in [56][57].

From the above it is clear that there are many advantages and disadvantages associated with meta-heuristic algorithms. In traditional meta-heuristic scheduling, a single algorithm is typically employed, which is not problem-specific and may perform poorly under varying conditions. In contrast, the MQLH framework integrates the strengths of multiple meta-heuristics within its pool,

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
enabling it to adapt dynamically and pursue a more effective solution to the optimization objective represented in Equation (13).

Therefore, it is instructive to compare proposed MQLH with a few powerful meta-heuristics. The computing environment and testing environment will be the same as described before for MQLH and HSSA (and FIFO, Fair Scheduler) then the results of the comparisons are described in table 4. The hyper-parameters for compared meta-heuristics CS, BA and HSA are as described in table 1 while for GA, BFO and PSO are listed table 3.

Table 3: Concrete hyper-parameter settings for different meta-heuristics.

(1) For GA

Properties	Values/Descriptions
Population Size	50
Fitness (F) criteria	$eq.(8)$
Probabilities of crossover p_c	0.8
Probabilities of mutation p_m	0.2
Generations	100
Crossover type	Single-point
Mutation type	Single-point

(2) For BFO

Properties	Values/Descriptions
Population Size	50
Objective function	$eq.(13)$
N_c	As described in [50].
N_s	As described in [50].
N_{re}	As described in [50].
N_{ed}	As described in [50].
P_{ed}	As described in [50].
$C(i)$	As described in [50].
Generations	100

(3) For PSO

Properties	Values/Descriptions
Inertia weight	0.9
Objective function	$eq.(8)$
Acceleration coefficient	1.0
Iterations	100

Table 4: Results of comparing MQLH with state of art meta-heuristics.

Task	**Algorithm (*performance in sec*)**

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

		BFO	PSO	GA	CS	BA	HSA	MQLH
Grep	Average	886	900	850	847	935	880	789
	Best	808	820	800	805	890	860	780
	Worst	965	980	901	890	980	900	835
Log Analyzer	Average	1367	1455	1275	1265	1265	1270	1161
	Best	1305	1401	1250	1220	1210	1225	1101
	Worst	1430	1510	1300	1310	1320	1315	1221
TeraSort	Average	860	960	872	940	910	937	835
	Best	800	910	820	830	865	850	770
	Worst	920	1010	925	1050	955	1025	900

5. Result Analysis

First, we tested the cutting-edge algorithms HSSA, FIFO, and FIFO (table 2).
. Here we observed that:

1) For Grep job, proposed MQLH outperforms the other, but this lead is marginal that can be justified by knowing the fact that the Hadoop system is not at its full capacity as new jobs arrived after previous jobs are finished (figure 6).
2) In log analyzer jobs, where workload size has a major impact, MQLH performs notably better than the alternative methods (Figure 7).
(3) For TeraSort, the performance difference is more significant, largely due to job bursts and increased job sizes (Figure 8).

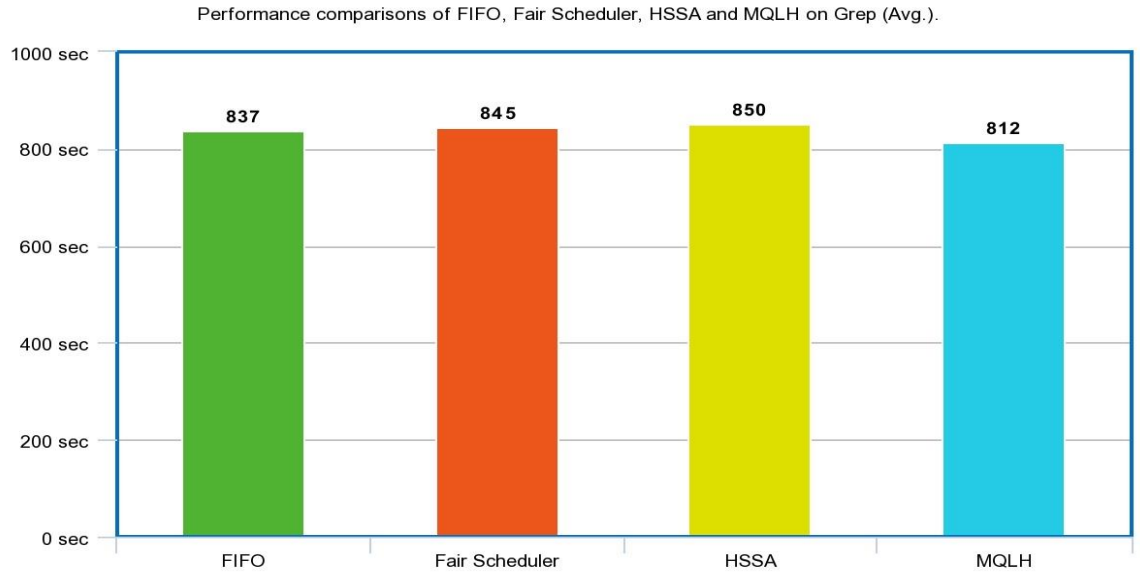

Figure 6: Performance comparisons on Grep.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

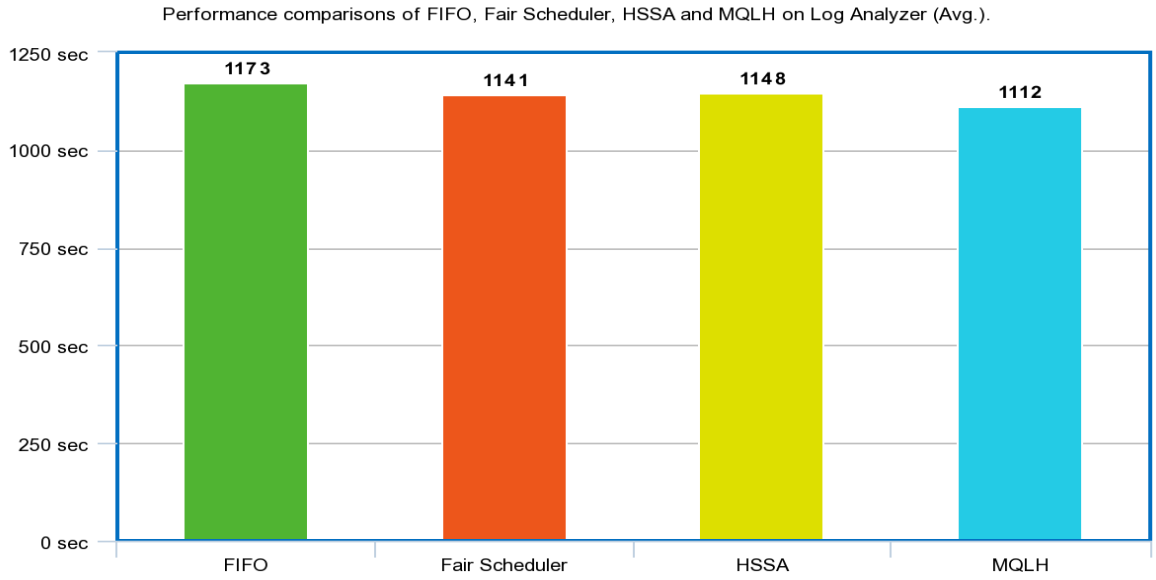

Figure 7: Performance comparisons on Log Analyzer.

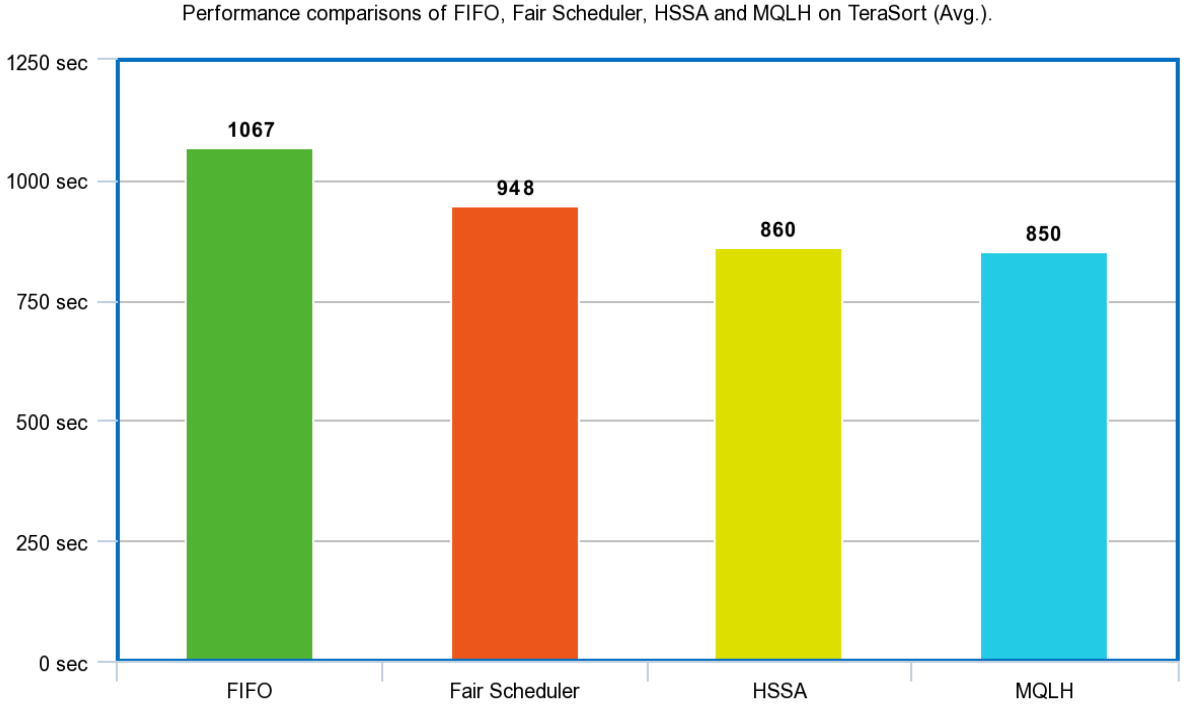

Figure 8: Performance comparisons on TeraSort.

In our second experiment MQLH is compared with state of art meta-heuristics (BFO, PSO, GA, CS, BA and HSA) where CS, BA and HSA are also used in meta-heuristic pool (table 2) (here it is used individually on e.q. (13)). Here it can be observed that:

1) Again for Grep, MQLH outperforms the other, but this performance is marginal for the same previous reasons (figure 9).

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

2) In the Log Analyzer case, MQLH outperforms the others, mainly due to the impact of larger data sizes, which drives the observed performance gains (Figure 10).
3) In TeraSort a moderate performance difference can be seen while MQLH outperforms the other meta-heuristics. This again can be attributed to the burst of jobs and also the size of jobs (figure 11).

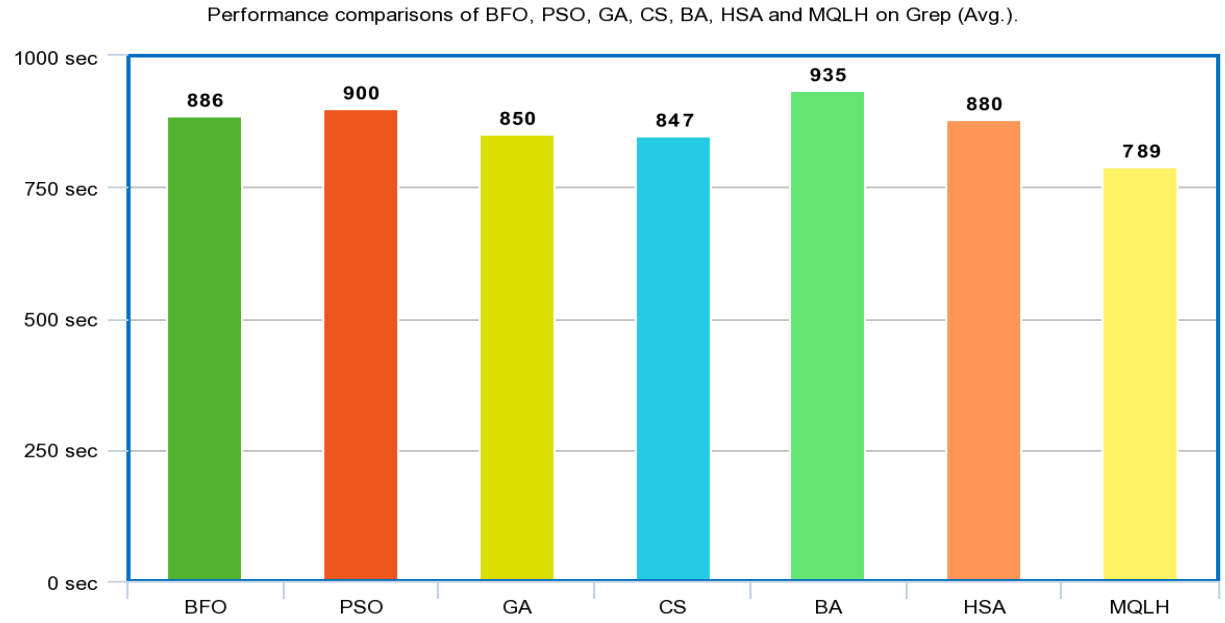

Figure 9: Performance comparisons on Grep.

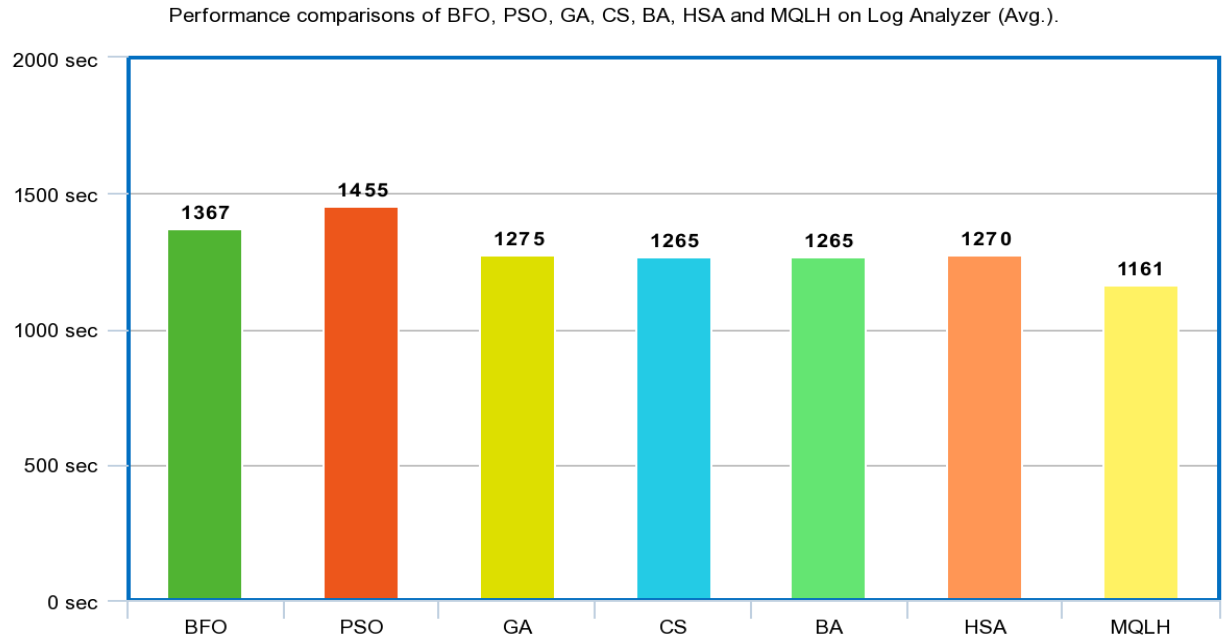

Figure 10: Performance comparisons on Log Analyzer.

Received: August 09, 2025

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

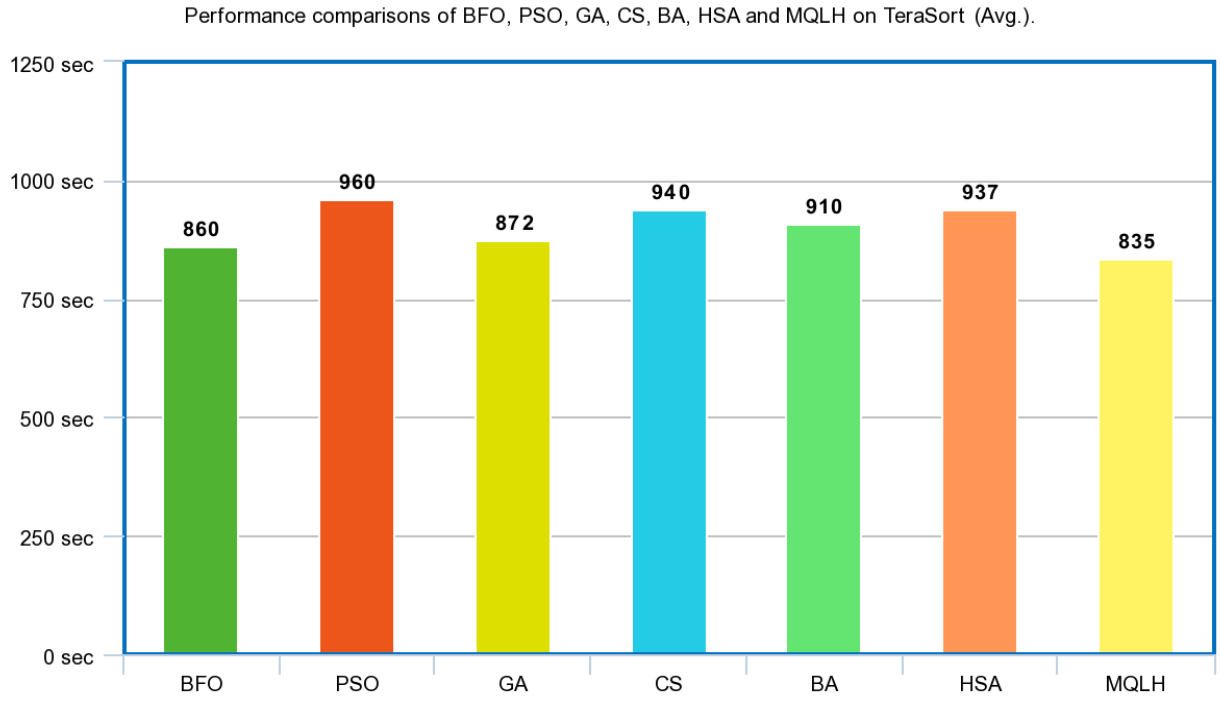

Figure 11: Performance comparisons on TeraSort.

6. Conclusions

This paper presents a modified Q learning based hyper heuristic framework for cloud scheduling problem. In particular, we want to use the hidden knowledge in the dataset itself to automate our task. In the proposed model the modified Q-learning generates a heuristic derived from the knowledge inherent in the data set itself and by combining advantages of different meta-heuristics. Meta-heuristics, in our model, on the other hand works directly on the basic set of tasks which are given to the system in reality. We have chosen HSA, CS and BA as our meta-heuristics pool. With this strategy we had proposed the Modified Q-Learning Hyper-heuristic (MQLH) algorithm that solves the NP-complete cloud scheduling problem in a way that outperforms or comparable to other compared algorithms.

The hypothesis that MQLH will be more robust and problem centric which then yield better performance is then validated. We find this opportunity to work with real system Hadoop directly, which is the system of choice for cloud computing and therefore we caste all of our results on Hadoop. For comparisons with MQLH we have implemented other state of the art cloud scheduling methods which includes a hyper-heuristic, a few deterministic and a few meta-heuristic methods. In comparisons it can be clearly seen that our model performs better than others on most of the time while in other cases it marginally outperforms others. This validates the stated hypothesis.

As the future work many other issue can be addressed e.g. selection of the hyper-parameters of meta-heuristics and hyper-heuristics for getting more optimal performance, complexity analysis,

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
data specific meta-heuristic selection and performance of other reinforcement learning algorithm (e.g. SARSA) with respect to proposed MQLH. Specifically, it will be very interesting to compare SARSA with Q-learning.

Declarations:

Ethical Approval: "not applicable".

Competing interests: "The authors declare that they have no conflict of interest."

Authors' contributions:

1) **Mr. Arvind Upadhyay:** All aspect of the article.
2) **Ramesh Thakur:** Review & editing, Validation, Supervision
3) **Archana Thakur:** Review & editing, Validation, Supervision
4) **Kavita Upadhyay:** Review & editing, Validation

Funding: "This research work receives no funding"

Availability of data and materials: "All of the result analysis and Matlab code is available upon request to the first author."

References:

[1] S. Singhal and A. Sharma, "A job scheduling algorithm based on rock hyrax optimization in cloud computing," *Computing*, vol. 103, no. 9, pp. 2115–2142, 2021.

[2] B. Hayes, "Cloud computing," *Communications of the ACM*, pp. 9–11, 2008.

[3] D. C. Marinescu, *Cloud Computing: Theory and Practice*. Morgan Kaufmann, 2022.

[4] E. Gorelik, "Cloud computing models," Ph.D. dissertation, Massachusetts Institute of Technology (MIT), 2013.

[5] 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, 2014.

[6] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35–44, 2022.

[7] K. V. Kumar and A. Rajesh, "Multi-objective load balancing in cloud computing: A meta-heuristic approach," *Cybernetics and Systems*, vol. 54, no. 8, pp. 1466–1493, 2023.

[8] R. Valarmathi and T. Sheela, "Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. Suppl. 5, pp. 11975–11988, 2019.

[9] T.-K. Dao, T.-S. Pan, T.-T. Nguyen, and J.-S. Pan, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," *J. Intell. Manuf.*, vol. 29, pp. 451–462, 2018.

[10] Q.-V. Pham, S. Mirjalili, N. Kumar, M. Alazab and W.-J. Hwang, "Whale optimization algorithm with applications to resource allocation in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4285–4297, 2020.

[11] F. S. Gharehchopogh and H. Gholizadeh, "A comprehensive survey: Whale optimization algorithm and its applications," *Swarm Evol. Comput.*, vol. 48, pp. 1–24, 2019.

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
[12] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A systematic and meta-analysis survey of whale optimization algorithm," *Comput. Intell. Neurosci.*, vol. 2019, no. 1, Article ID 8718571, 2019.

[13] M. Petrović, Z. Miljković, and A. Jokić, "Optimal single mobile robot scheduling using whale optimization algorithm," *Appl. Soft Comput.*, vol. 81, p. 105520, 2019.

[14] L. Han, S. Zhu, H. Zhao, and Y. He, "An enhanced whale optimization algorithm for task scheduling in edge computing environments," *Front. Big Data*, vol. 7, p. 1422546, 2024.

[15] W. Yang, J. Su, Y. Yao, Z. Yang, and Y. Yuan, "A novel hybrid whale optimization algorithm for flexible job-shop scheduling problem," *Machines*, vol. 10, no. 8, p. 618, 2022.

[16] H. Jin, S. Lv, Z. Yang, and Y. Liu, "Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition," *Appl. Soft Comput.*, vol. 114, p. 108053, 2022.

[17] Y. Liu *et al.*, "HPCP-QCWOA: High performance clustering protocol based on quantum clone whale optimization," *Future Gener. Comput. Syst.*, vol. 135, pp. 315–332, 2022.

[18] S. Sarkar, S. Vimala, M. Vignesh, and C. Sivakumaran, "A hybrid framework for job scheduling through firefly and cuckoo search," *Int. J. Comput. Sci. Math.*, vol. 20, no. 3, pp. 197–207, 2024.

[19] H. R. Boveiri, "Enhanced cuckoo optimization algorithm for task graph scheduling in cluster computing," *Soft Comput.*, vol. 24, no. 13, pp. 10075–10093, 2020.

[20] J. Gong, "QoS improvement in IoT networks using cuckoo search algorithm," *Wireless Pers. Commun.*, vol. 126, no. 3, pp. 2321–2346, 2022.

[21] D. S. Johnson and M. R. Garey, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[22] J. D. Ullman, "NP-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, 1975.

[23] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, pp. 1–16, 2018.

[24] N. Li, R. Dhiman, and J. Shanmugapriyan, "Modified cat swarm optimization for intelligent physical education," *J. Interconnect. Networks*, vol. 22, no. S1, p. 2141019, 2022.

[25] Y. Li and G. Wang, "Sand cat swarm optimization with elite collaboration," *IEEE Access*, vol. 10, pp. 89989–90003, 2022.

[26] Z. He, M. Zhao, T. Luo, and Y. Yang, "Compact cat swarm optimization based on probability model," *Appl. Sci.*, vol. 12, no. 16, p. 8209, 2022.

[27] T. Vaiyapuri *et al.*, "Cat swarm optimization-based lung cancer classification," *Appl. Sci.*, vol. 12, no. 11, pp. 5491, 2022.

[28] X. Xiao and M. Zhao, "Routing optimization in IoT using improved cat swarm," *Neural Comput. Appl.*, 2022.

[29] P. Hu *et al.*, "Binary PSO for feature selection," *Appl. Soft Comput.*, vol. 121, p. 108736, 2022.

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
[30] P.-W. Tsai, J.-S. Pan, S.-M. Chen, and B.-Y. Liao, "Enhanced parallel cat swarm optimization based on Taguchi method," *Expert Syst. Appl.*, vol. 39, no. 7, pp. 6309–6319, 2012.

[31] T. P. Talafuse and E. A. Pohl, "Bat algorithm for redundancy allocation," *Eng. Optim.*, vol. 48, no. 5, pp. 900–910, 2016.

[32] Y. Zhou, Q. Luo, J. Xie, and H. Zheng, "Hybrid bat algorithm for vehicle routing problem," in *Metaheuristics and Optimization in Civil Engineering*, 2016, pp. 255–276.

[33] J. Xie, Y. Zhou, and H. Zheng, "Hybrid bat algorithm for aircraft landing problem," *J. Appl. Math.*, vol. 2013, p. 742653.

[34] Y. Lu and T. Jiang, "Discrete bat algorithm for job shop scheduling," *IEEE Access*, vol. 7, pp. 14513–14522, 2019.

[35] Y. Xu and D. Pi, "Hybrid enhanced bat algorithm for redundancy allocation," *Swarm Evol. Comput.*, vol. 50, p. 100562, 2019.

[36] S. Supreeth and K. Patil, "GA-MPSO for VM scheduling," *Int. J. Emerg. Technol. Learn.*, vol. 17, no. 7, pp. 208–225, 2022.

[37] S. Nabi and M. Ahmed, "PSO-RDAL: Resource-aware dynamic load balancer," *J. Supercomput.*, vol. 78, no. 4, pp. 4624–4654, 2022.

[38] J. Balusamy and M. Karunakaran, "Immune + PSO scheduling," *Distrib. Parallel Databases*, vol. 40, no. 1, pp. 85–107, 2022.

[39] W. A. N. G. Gang *et al.*, "Cloud task scheduling using PSO and Capuchin search," *IJACSA*, vol. 14, no. 7, 2023.

[40] G. Wei, "Quadratic particle swarm optimisation for task scheduling," *J. Inf. Knowl. Manag.*, vol. 22, no. 01, p. 2250067, 2023.

[41] R. Pellerin, N. Perrier, and F. Berthaut, "Survey of hybrid metaheuristics," *Eur. J. Oper. Res.*, vol. 280, no. 2, pp. 395–416, 2020.

[42] L.-Y. Tseng and S.-C. Liang, "Hybrid metaheuristic for QAP," *Comput. Optim. Appl.*, vol. 34, no. 1, pp. 85–113, 2006.

[43] Z. Zhang *et al.*, "Two-stage hybrid meta-heuristic for corridor allocation," *Adv. Eng. Inform.*, vol. 53, p. 101700, 2022.

[44] M. N. Aktan and H. Bulut, "Metaheuristic task scheduling for cloud environments," *Concurrency Comput.: Pract. Exper.*, vol. 34, no. 9, p. e6513, 2022.

[45] P. Kumari and S. K. Sahana, "Hybrid ACO-PSO for QoS multicast routing," *Wireless Pers. Commun.*, pp. 1–23, 2022.

[46] E. N. Alkhanak and S. P. Lee, "Hyper-heuristic cost optimisation for workflow scheduling," *Future Gener. Comput. Syst.*, vol. 86, pp. 480–506, 2018.

[47] E. K. Burke *et al.*, "Hyper-heuristics: A survey," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.

[48] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[49] X.-S. Yang, *Nature-Inspired Optimization Algorithms*. Academic Press, 2020.

Received: August 09, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
[50] S. Srichandan, T. A. Kumar, and S. Bibhudatta, "Hybrid bacteria foraging for cloud scheduling," *Future Comput. Inform. J.*, vol. 3, no. 2, pp. 210–230, 2018.

[51] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[52] MATLAB Parallel Server, "Configure a Hadoop cluster," MathWorks, Available: https://www.mathworks.com/help/matlab-parallel-server/configure-a-hadoop-cluster.html.

[53] Google Cloud, "What is Hadoop?," Available: https://cloud.google.com/learn/what-is-hadoop.

[54] Lawrence Berkeley Laboratory, "WorldCup98 Dataset," Available: https://ita.ee.lbl.gov/html/contrib/WorldCup.html.

[55] A. Sharma, "Stochastic nonparallel hyperplane SVM and no-free-lunch theorems," *Evol. Intell.*, vol. 15, no. 1, pp. 215–234, 2022.

[56] A. Sharma, "Nature inspired algorithms with hypercomputational perspective," *Inf. Sci.*, vol. 608, pp. 670–695, 2022.

[57] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, pp. 33–57, 2007.

[58] Y. Shi and R. C. Eberhart, "Empirical study of PSO," in *Proc. CEC*, IEEE, 1999, pp. 1945–1950.

[59] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "Exposing misleading metaphor-based algorithms," *Int. Trans. Oper. Res.*, vol. 30, no. 6, pp. 2945–2971, 2023.

[60] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "Why cuckoo search does not bring novel ideas," *Comput. Oper. Res.*, vol. 142, p. 105747, 2022.

[61] K. M. Passino, "Bacterial foraging optimization," in *Innovations & Developments of Swarm Intelligence*, IGI Global, 2012, pp. 219–234.

[62] A. Upadhyay, R. Thakur, and A. Thakur, "A Nature Inspired Crow Search Algorithm (CSA) Approach for Efficient Task Scheduling in Cloud Computing.," *Journal of Computational Analysis and Applications (JoCAAA)*, vol. 33, no. 4,pp 945–966, 2024.

Received: August 09, 2025