Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
DOI: 10.22034/jscit.2025.220413
Volume 14, Number 1, Spring 2025, pp. 57-67
Received: 01/30/2025, Revised: 04/10/2025, Accepted: 05/04/2025

A Chaotic RIME Optimization Algorithm based on Generalized Quadratic Interpolation with Levy Flight for Task Scheduling in Cloud Computing

Mohammad Reza Sabipour[1], Monireh H. Sayadnavard[2*]
1- Department of Computer Engineering, Bandar Anzali Branch, Islamic Azad University, Bandar Anzali, Iran.
2- Department of Computer Engineering, Bandar Anzali Branch, Islamic Azad University, Bandar Anzali, Iran.
[1]mohammadreza.sabipour@iau.ac.ir, [2*]mo.sayadnavard@iau.ac.ir

Corresponding author's address: Monireh H. Sayadnavard, Faculty of Computer Engineering, Islamic Azad University of Bandar Anzali, Bandar anzali, Iran.

Abstract - Task scheduling in cloud computing still remains a crucial issue regarding system performance and user satisfaction. As organizations increasingly rely on cloud infrastructure to manage and execute their computational tasks, the need for effective scheduling becomes paramount. The main purpose of task scheduling is to assign submitted tasks to available appropriate resources while maintaining the quality of service and service level agreement. An effective task scheduling algorithm must be able to reduce the makespan as a crucial performance metric in cloud computing systems. In this paper, we have introduced a new efficient task scheduling algorithm called GQIRIME based on the RIME optimization algorithm to reduce the makespan, cost, and total execution time of the cloud system. An enhanced exploitation strategy is proposed based on generalized quadratic interpolation and levy flight to increase convergence rate while maintaining robust search. Furthermore, the proposed algorithm is integrated with chaos mapping to get a more diversified initial population. We have evaluated the effectiveness of our proposed approach using the Cloudsim toolkit. The results analysis demonstrates that our proposed algorithm outperforms other methods and significantly improves key performance metrics in task scheduling. According to the experiment results, the proposed task scheduling algorithm achieved a lower cost, makespan, and total execution time. The results show a 29.3% improvement in terms of makespan, 61.8% improvement in cost, and 29.4% improvement in total execution time on average compared to counterparts.

Keywords- Task scheduling, Rime optimization algorithm, Cloud computing, Quadratic interpolation, Meta-heuristic.

I. INTRODUCTION

Cloud computing can be defined as a paradigm that provides convenient and on-demand access to a diverse shared pool of computing resources, including storage, servers, networks, applications, and services. These resources are provisioned over the Internet according to user requirements, offering flexibility and efficiency. The cloud model is characterized by several key features such as on-demand self-service, rapid elasticity, cost-effectiveness, and efficiency. In the cloud computing paradigm, ensuring the desired Quality of Service (QoS) between the provider and the customer is crucial. QoS requirements are established through Service Level Agreements (SLAs), which are conventional agreements outlining the expected service quality from the service provider [1-4]. Cloud computing provides three types of services that are delivered and consumed in real-time, named: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [5, 6].

Cloud servers include physical machines (servers) and an extra layer (the virtual environment of computing resources) that executes user applications [7]. This layer is created by constructing an abstraction layer over computer hardware, hardware components are separated into multiple virtual computers, also called virtual machines (VMs). Each VM is allocated to a single physical machine and consumes the resources of the host machine [6, 8]. Virtual resources are assigned to users based on their requirements. Since these

resources can be heterogeneous, a scheduling mechanism is needed to map user tasks to available resources. In the cloud computing environment, the process of effectively mapping user tasks across multiple VMs is called task scheduling. Task scheduling algorithms directly affect the proficiency of users' tasks and the efficient usage of resources [9]. Task scheduling is known as a major challenge in cloud computing [3]. Therefore, it is important to propose an efficient scheduling algorithm for dispatching user tasks to appropriate virtual machines while meeting QoS requirements [10].

Task scheduling is considered an NP-complete problem [9]. When dealing with such problems finding the optimal solution is not possible in a reasonable time frame. Therefore, approximation techniques are used to find a good solution without finding the global optimum. Non-heuristic techniques rely on mathematical optimization methods that use derivatives (gradients) to find optimal solution. These techniques may fail when dealing with complex optimization problems where the search space is non-differentiable. Meta-heuristic algorithms have proven to be good candidates to solve these problems [11]. They are widely used for solving optimization problems across various fields of engineering and science [12]. Due to their stochastic nature, these algorithms do not guarantee optimality, but rather an approximation of the optimal solution [13]. Meta-heuristics can be applied to a wide range of optimization problems, but it is not guaranteed to get the same performance for all problems. Hence, some modifications need to be made to boost their performance for a specific problem domain. While these techniques may seem promising, they may suffer from limitations such as poor convergence rate and getting stuck in local optimums when applied to a specific optimization problem. Since time is limited, we must find a quality solution in a reasonable amount of time. This gets even more challenging as the problem size increases [12, 13].

The rime optimization algorithm (RIME), which is discussed in this study, has previously been adapted for applications such as feature selection, image segmentation, and other engineering tasks [14-17]. The rime optimization algorithm is a great choice for high-dimensional complex search spaces such as task scheduling where the number of tasks and resources can grow rapidly. As mentioned before, even if the RIME algorithm works well, the algorithm may perform differently, depending on the characteristics of the optimization problem. Therefore, it is necessary to develop and improve it to solve the scheduling problem.

In this paper, the RIME algorithm has been strengthened by integrating different algorithms. The proposed algorithm is called GQIRIME. We have adopted several techniques such as Generalized Quadratic Interpolation and the concept of Random Walks to enhance the exploitation capability of the rime optimization algorithm. Enhancing the exploitation ability can help the algorithm find quality solutions in less time with a better convergence rate. By adopting these concepts, we have developed a new exploitation operator for the rime optimization algorithm. The main objective of our work is to reduce the makespan, cost, and total execution time of the cloud system by enhancing the rime optimizer. The paper makes the following contributions.

- Introducing the GQIRIME algorithm to optimize the scheduling problem in cloud computing and to reduce the makespan and cost in a limited time frame.
- Proposing an enhanced exploitation strategy based on generalized quadratic interpolation and levy flight for a faster and more robust exploitation phase.
- Integrating chaos mapping to get a more diversified initial population, preventing premature convergence.
- Experimental evaluation to validate the effectiveness of our proposed algorithm

The rest of the paper is organized as follows: In Section 2 we provide a review of prior studies and literature. In Section 3, the system architecture is presented. The proposed GQIRIME is discussed in Section 4, Section 5, presents the evaluation of the proposed algorithm and comparative study, and finally, Section 6 summarizes the conclusions.

II. RELATED WORKS

Task scheduling in cloud computing is known as an NP-complete problem [18]. Therefore, many existing studies have been using meta-heuristics for optimizing task scheduling in cloud computing [3, 11, 19-25]. The authors in [3] have proposed a novel algorithm by combining two optimization algorithms namely the Chimp Optimization Algorithm (COA) and the Whale Optimization Algorithm (WOA). The proposed is referred to as the Modified Chimp-Whale Optimization (MCWO) algorithm. The MCWO aims to minimize makespan, energy consumption, and latency. To achieve the desired performance, the COA is enhanced by integrating the Bubble-net attacking method. This technique has improved the ability of the algorithm to escape the local optimal. A simulation was conducted using Cloudsim to evaluate the efficiency of the proposed algorithm. The simulation environment includes several different scenarios with a varying range of tasks and virtual machine quantities. The proposed algorithm has been compared with numerous popular algorithms, such as the Whale Optimization Algorithm (WOA), Cuckoo Algorithm (CA), Grey Wolf Optimization (GWO), Ant Lion Optimizer (ALO), and Genetic Algorithm (GA). Simulation results confirm that the proposed algorithm surpasses other methods, decreasing makespan and energy consumption. Emami in [11], has proposed the enhanced sunflower algorithm to improve the performance of task scheduling in terms of makespan and energy consumption. The proposed algorithm introduces a novel pollination strategy that enhances both exploitation and exploration simultaneously. The simulation environment contains randomly generated tasks to simulate real-world configuration. The simulation environment contains randomly generated tasks to simulate real-world configurations. Results show a 0.73% and 2.24% improvement in makespan and energy consumption metrics compared to the best results achieved by other algorithms. Latreche et al., [19] propose a multi-objective optimization method called the Non-dominated Sorting Genetic Algorithm (NSGA-III) to optimize the scheduling problem in cloud computing environments. The NSGA-III algorithm was introduced to overcome the limitations of the NSGA-II

algorithm. The first improvement is based on the selection mechanism. The original selection mechanism in NSGA-II does not perform well for multi-objective problems. Therefore, NSGA-III utilizes a modified selection mechanism based on predefined reference points. The authors found that the original NSGA-III algorithm may run into problems during the normalization process. Hence, a solution has been proposed to make the normalization function work properly. The algorithm has been evaluated in terms of total runtime, cost, and energy consumption metrics. Results indicate that the proposed NSGA-III algorithm outperforms NSGA-II in all metrics. Natesan et al., [20] have proposed the mean grey wolf optimization algorithm that aims to minimize makespan and energy consumption metrices. To increase the performance of the grey wolf optimizer, the authors modified the encircling and hunting equations. These modifications have improved the search path of each wolf in the search space. Simulation results obtained from the tests prove the efficiency of the proposed mean grey wolf optimizer compared to its counterparts. Kruekaew and Kimpan [21], proposed a multi-objective task scheduling technique based on the Artificial Bee Colony Algorithm with Q-learning called MOABCQ. The main focus of the study is to balance out the load over VMs while improving makespan, cost, and resource utilization. Integrating Q-learning with the ABC algorithm has improved the algorithm's convergence rate. Therefore, load detection is facilitated, and the algorithm can find appropriate resources to run tasks. Experimental results prove the proposed method outperforms other algorithms by minimizing the objective function and reducing the degree of load imbalance. In [22] the authors propose an improved hybrid weighted ant colony optimization algorithm called HWACO. The proposed algorithm aims to achieve convergence in a short period while finding quality solutions. This is achieved by improving two processes, namely the construct-ant method and the update phenomenon phase. The algorithm is evaluated in terms of makespan and cost, where it shows improvements. Mangalampalli et al., [23] argue that the inappropriate assignment of tasks to computational resources can decrease the level of quality of service, which leads to a degradation of trust in the cloud provider. Therefore, the authors have proposed a trust-aware task scheduling method using the Firefly optimization algorithm. The proposed method considers task priorities and virtual machines to efficiently assign tasks to virtual machines. In [24] the authors propose a fault-aware task scheduling technique to decrease the probability of resource failure. The proposed task scheduling technique consists of two phases: clustering and scheduling. In the clustering phase, a clustering algorithm called DBSCAN is used to cluster VMs based on their specifications. Therefore, user tasks will be mapped to highly configured VMs to decrease the probability of failure occurrence. Finally, the min-min algorithm is used to select the best-fit VM from the cluster of VMs that have the same computing power and failure probability. Saravanan et al., [25] propose an Improved Wild Horse Optimization with Levy Flight algorithm (IWHOLF-TSC). The proposed approach utilizes a multi-objective fitness function to minimize makespan and maximize resource utilization. The proposed IWHOLF-TSC incorporates the Wild Horse Optimization algorithm with the concept of random walks. Results indicate that the proposed algorithm outperforms other common meta-heuristic approaches in terms of task scheduling metrics. Kamalinia et al., [26] propose a hybrid meta-heuristic approach for optimizing task scheduling. The proposed algorithm combines the genetic algorithm (GA) with the particle swarm optimizer (PSO). The results of the simulation indicate that the proposed algorithm has minimized the makespan metric. Esfandi et al., [27] state that due to the random and dynamic nature of the cloud environment, proper online job scheduling is a challenging task. Two different learning automata are used for each scheduler. There are two main challenges surrounding the cloud environment: choosing the proper job among the submitted user jobs and choosing the appropriate virtual machines to execute user jobs. To deal with these challenges, the authors have used two distinct learning automata. To prove the effectiveness of the proposed approach, the algorithm is compared with other existing algorithms in terms of makespan, success rate, average waiting time, and load imbalance degree. Reducing the energy consumed by data centers is a challenging task. The authors in [28] propose a novel approach to achieve a reliable VM consolidation method. A Markov chain model is used to determine the reliability of PMs and prioritize them based on the reliability level. The proposed method uses two algorithms to determine source and destination servers. Experimental results show that the proposed VM consolidation approach successfully minimizes energy consumption while decreasing the number of inefficient VM migrations.

Based on the reviewed works, it can be concluded that while progress has been made in task scheduling, the desired outcomes have not yet been fully achieved. Due to the complex nature of the task scheduling, many of the existing strategies remain time consuming, especially when the number of objectives increases. Introducing multiple objectives can easily turn the problem into a challenging one, leading to significantly longer computational times. In this paper, we aim to propose a scheduling algorithm that can find a quality solution in a reasonable time frame with a smaller number of function evaluations while considering makespan, cost, and total execution time.

III. SYSTEM MODEL

This section provides an overview of the cloud system architecture. Cloud computing aims to provide a network of virtual services so that users can access and deploy applications on demand from anywhere in the world [29]. The following describes the high-level architecture in a cloud computing infrastructure.

A. Framework

Fig. 1 illustrates the system architecture of task scheduling in cloud computing. As shown in the figure, users will send a set of tasks to the cloud provider. The key component that is responsible for task scheduling is the broker. A broker is an intermediary entity that uses a scheduling algorithm to assign user tasks to available resources. Cloud servers send

resource data to the broker. Together with the data coming from the users, the scheduling algorithm will provision tasks to available virtual machines [19].

B. Cloud System

In the cloud computing environment, the cloud infrastructure (DS) is constructed of physical machines. All physical machines (PM) contain VMs as follows [25]:

$$DS = [PM_1, PM_2, \cdots, PM_k] \quad (1)$$

Where each physical machine (PM_i) contains M virtual machines.

$$PM = [VM_1, VM_2, \cdots, VM_M] \quad (2)$$

In the cloud computing environment, multiple VMs can be dynamically lunched or stopped on a single physical machine based on client requests, hence multiple VMs can concurrently run user applications on a single physical machine [29].

The scheduling algorithm generates a scheduling plan that assigns user tasks to virtual machines. Each task should be in the scheduling plan and can only be executed on a single virtual machine. The process starts with a list of size N, where N is the number of user tasks sent to the cloud provider. Each element of the list holds an integer corresponding to a VM identifier. Given the scheduling plan, the broker will assign user tasks to the specified virtual machines.

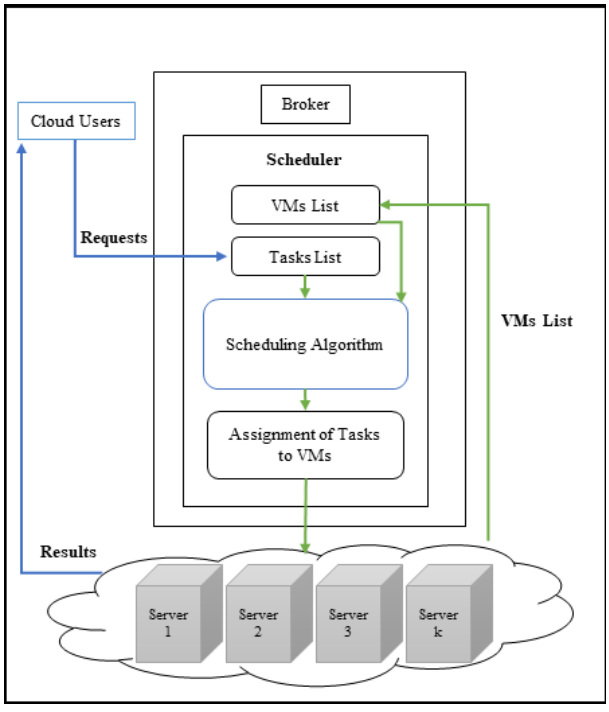


Fig. 1. System architecture [19]

C. Problem Formulation

To solve the scheduling problem, first, a list of size $N-1$ is created that will hold the VM identifiers. This list is initially randomized and evolves over time using evolutionary operators, such as mutation, to converge toward an optimal solution.

Constraints: Since tasks are non-divisible, each task can only be assigned to one VM and cannot be reassigned once allocated. Additionally, not all VMs need to be powered on for the scheduling process. Also, Tasks may require certain resources (e.g., memory or CPU), and the VM must have those available.

Decision variables: In the context of task scheduling, decision variables are the elements of the list (VM indices). For example, {2, 1, 3, 1} is a scheduling plan where the first task will be executed by VM with id #2, the second task is assigned to VM #1, the third one is assigned to the third VM, and the last one will run on the first VM.

Fitness value: For the purposes of this work, we have considered the makespan value of each scheduling plan as the fitness value.

IV. PROPOSED ALGORITHM

The rime optimization algorithm (RIME) [30] is a new meta-heuristic algorithm based on the mechanism of rime-ice formation. The RIME optimizer is composed of two phases: soft-rime and hard-rime. The soft-rime search strategy enables the ability of searching the entire search space, while the hard-rime phase mainly focuses on the exploitation ability. The RIME mechanism ensures a balance between exploration and exploitation by mimicking the rime-ice behaviors [30]. Even though the original RIME optimizer performs well, it has some shortcomings when dealing with high dimensional optimization problems, such as getting stuck in the point of local optima and poor convergence performance. To address these issues, we propose an enhanced version of the RIME optimizer, namely GQIRIME, to further improve its efficiency. We introduce a new hard-rime operator based on generalized quadratic interpolation for faster convergence when dealing with high dimensional problems. The concept of random processes was also used to facilitate the process of preserving the diversity of solutions. Moreover, to improve the diversity of the GQIRIME during the initial phases, we equipped the GQIRIME with a chaotic mapping mechanism to generate a more diversified initial population. The following describes how GQIRIME is applied for task scheduling in cloud computing.

A. Population Initialization

The algorithm starts with initializing a population of solutions called rime-population. The rime-population is composed of S rime agents denoted as R_i where each rime agent contains D rime particles $r_{i,j}$.

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,D} \\ r_{2,1} & & r_{2,D} \\ \vdots & & \vdots \\ r_{S,1} & \cdots & r_{S,D} \end{bmatrix} \quad (3)$$

Each rime agent R_i represents a scheduling plan and each particle refers to a VM identifier that a task runs on.

Population diversity in the early stages is crucial for the algorithm's ability to explore the entire search space and prevent premature convergence. To overcome this issue, an improvement has been made based on chaotic mappings. Chaos theory incorporates randomness and instability, which enhances exploration and helps prevent getting stuck in local optima when dealing with high-dimensional optimization problems [31]. The authors in [32] introduced a Bernoulli chaos mapping with mutual exclusion mechanism to boost the diversity of the initial population. We integrated these techniques with GQIRIME to improve the performance of the proposed algorithm during the initial phases. The process is defined as follows:

$$z^{B}{}_{i,j}(k+1) = \begin{cases} \dfrac{z^{B}{}_{i,j}(k)}{1-\beta}, & z^{B}{}_{i,j}(k) \in (0, 1-\beta] \\ \dfrac{(z^{B}_{i,j}(k) - 1 + \beta)}{\beta}, & z^{B}{}_{i,j}(k) \in (1-\beta, 1] \end{cases} \quad (4)$$

Where β is considered 0.499 [32], $z(0)$ is a random number in the range [0,1] generated randomly and k is the iteration counter.

Furthermore, the mutual exclusion mechanism is applied to disperse rime agents in two different directions. Considering the mutual exclusion mechanism, two rime particles are generated using the Bernoulli chaotic mapping concept. The two are compared with each other and the one with the best fitness value is chosen. Mutual exclusion is applied using the following equations [32]:

$$r^{1}_{i,j} = r^{min}_{j} + z^{B}{}_{i,j}\left(r^{max}_{j} - r^{min}_{j}\right) \quad (5)$$

$$r^{2}_{i,j} = r^{max}_{j} - z^{B}{}_{i,j}\left(r^{max}_{j} - r^{min}_{j}\right) \quad (6)$$

$$r_{i,j} = min\left(r^{1}_{i,j}, r^{2}_{i,j}\right) \quad (7)$$

Where r^{min}_{j} and r^{max}_{j} are respectively set to 1 and M. The fitness value is the makespan of that solution. Since scheduling tasks is a discrete problem, each particle is rounded to the nearest integer.

B. Soft-rime Phase (Exploration)

As previously highlighted, the soft-rime operator is one of the two key steps of the RIME optimizer. This operator allows the algorithm to thoroughly explore the solution space. As mentioned before, the proposed GQIRIME uses a new modified hard-rime operator. To achieve an optimal balance between exploitation and exploration throughout the algorithm, we changed some parameters of the soft-rime operator to make sure both operators work together correctly. The original soft-rime operator calculates new positions through perturbations of the current best rime agent (r^{best}_{j}). In our modified soft-rime operator, $\left(r^{best}_{j}\right)$ is replaced with the j-th particle of the current selected rime agent ($r_{i,j}$) [30]. The soft-rime operator is defined as follows:

$$r^{new}_{i,j} = r_{i,j} + rand_1 . f . \left(h . \left(r^{max}_{j} - r^{min}_{j}\right) + r^{min}_{j}\right) \quad if\ rand_2 < E \quad (8)$$

Where $r^{new}_{i,j}$ is the new position of the updated particle, $rand_1$ and $rand_2$ are random numbers in range [-1,1] , h is a random number in the range (0,1), f and E are calculated using the following equations [30]:

$$f = \cos\left(\pi . \frac{iter}{iter_{max} . 10}\right) . \left(1 - \frac{\left[\frac{w . iter}{iter_{max}}\right]}{w}\right) \quad (9)$$

$$E = \sqrt{\left(\frac{t}{T}\right)} \quad (10)$$

Where $iter$ is the current iteration counter, $iter_{max}$is the maximum number of iterations, w is 5 and [.] denotes rounding. E is the coefficient that controls the probability of a particle being attached [30].

C. Hard-rime Phase (Exploitation)

The hard-rime operator takes inspiration from a phenomenon called rime puncture [30]. The original hard-rime operator replaces the current particle $r_{i,j}$ with the best j-th particle of the best-found agent so far. When dealing with high dimensional intricate problems, this approach may cause the algorithm to get trapped in local optima. In order to find high-quality solutions in a reasonable time frame and to effectively improve exploration and exploitation simultaneously, we propose a new hard-rime operator based on generalized quadratic interpolation (GQI). In fact, generalized quadratic interpolation is adopted to balance convergence and exploration well. The GQI method uses three random points and chooses the appropriate interpolation function based on the specific cases of the three points [33]. Using the GQI method, a new rime agent r^{q}_{i} is obtained to facilitate the process of exploring the unexplored areas of solution space and exploit the optimal solutions. The proposed GQIRIME utilizes the following hard-rime operator:

$$r^{new}_{i,j} = levy . \left(r^{q}_{i,j} - r^{best}_{j}\right) \quad if\ rand_3 < F^{normr}_{i} \quad (11)$$

Where $r^{new}_{i,j}$ is the new position of the updated particle, $rand_3$ is a random number in the range (-1,1), F^{normr}_{i} is the normalized fitness value of the current rime agent, r^{best}_{j} is the best rime agent in the current iteration and *levy* is a type of non-Gaussian random process called Levy flight. *levy* follows the levy flight distribution which is calculated using (12) and (13) [34]:

$$levy = \frac{u}{|v|^{1/\beta}} \quad (12)$$

Where u and v are random variables following the Gaussian distribution with zero mean. The standard deviation of v is set to 1 and σ_u is calculated using Eq. (13) [34]:

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin\frac{\pi\beta}{2}}{\Gamma\left[\frac{(1+\beta)}{2}\right]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta} \quad (13)$$

Where Γ(.) is the Gamma function, and beta is considered 1.5 respectively. On each iteration, the $levy$ variable will be multiplied by -1 with ½ probability to determine the direction of $levy$ step. Thus, the $levy$ variable will take both negative and positive values. Integrating the concept of levy flight improves the convergence rate of the algorithm, while providing randomness. Additional randomness is crucial for the algorithm when dealing with high dimensional intricate search space. When levy flight and GQI work together, they simultaneously enhance exploration and exploitation by applying robust mutation. Levy flight introduces increased randomness, helping the algorithm to escape local optimums, while GQI refines the solutions by generating high-quality particles, promoting a broader search of the solution space.

D. Positive Greedy Selection Mechanism

In this phase each rime agent in R^{new} will be compared to its parents and the one with the best fitness value will be chosen for the next generation. The idea is to ignore the agents that have worse fitness value; Thus, it ensures that the population consistently maintains good agents. At the end of this phase, the agent with the global optimum will be updated if an agent in R^{new} has better fitness value.

To summarize, in this paper the pseudo-code of the GQIRIME algorithm for solving the scheduling problem in cloud computing, is shown in Algorithm 1. Also, the overall structure of the algorithm is shown in Fig. 2 using a flowchart.

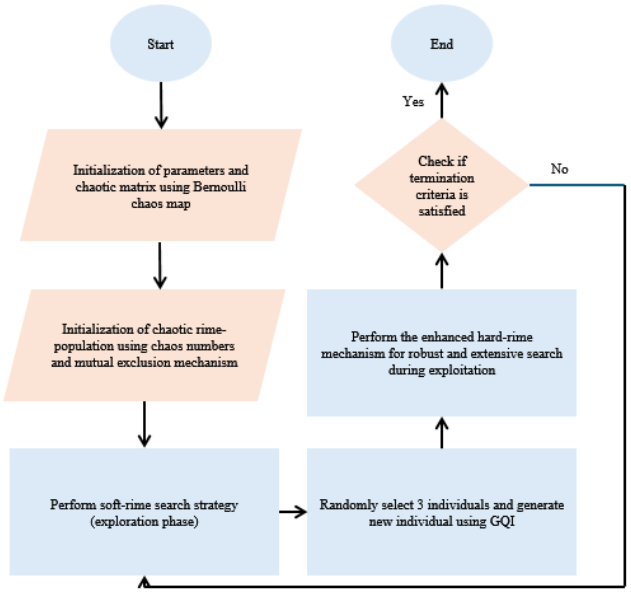


Fig. 2. Flowchart of the proposed algorithm

Algorithm 1: Pseudo-code of GQIRIME

Input:

N *tasks,* $T = \{t_1, t_2, \cdots, t_N\}$;

M *virtual machines,* $V = \{v_1, v_2, \cdots, v_M\}$;

Output:

An optimal task scheduling solution;

Set parameters:

maximum iterations, number of particles (S), dimension (D) = N, Lb = 1, Ub =M;

Generate chaotic random numbers by Eq. (4);

Initialize the rime-population R by Eq. (3), Eq. (5), Eq. (6) and Eq. (7);

Calculate the fitness of each rime agent;

Get the current best rime agent r^{best};

$iter = 0$;

While $iter < iter_{max}$ ***do***

  Calculate E by Eq. (10);

  For $i = 1$ *to S* ***do***

    For $j = 1$ *to D* ***do***

      If $rand_2 < E$:

        Apply soft-rime operator by Eq. (8);

      End If

      If $rand_3 < normalized\ fitness\ of\ R_i$:

        Apply hard-rime operator by Eq. (11);

      End If

    End For

    If $Fitness(R_i^{new}) < Fitness(R_i)$:

      Perform ***Positive greedy selection mechanism;***

      Update $\boldsymbol{r^{best}}$ *if there is a better solution*

  End For

  $iter \leftarrow iter + 1$;

End While

Return r^{best} *as the optimal solution;*

E. Computational Complexity

The computational complexity of the algorithm depends on the population size (n), dimensionality (d), and maximum number of iterations (T).

The complexity of RIME is primarily determined by its evolutionary operators. Both the soft-rime and hard-rime operators take $O(n^2)$ time to execute at worst-case time analysis. The positive greedy selection mechanism has a time complexity of $O(nlogn)$. Therefore the overall complexity of the RIME optimizer in terms of big-oh notation is $O(n + logn) * n)$ [30]. The proposed GQIRIME introduces an additional phase (the GQI) which takes $O(Tmd)$ time where m is the number of individuals used for the minimizer which is 3 and constant [33]. Therefore, the overall complexity of the GQIRIME is the same as the original RIME optimizer.

V. EXPERIMENTAL RESULTS

In this section, to investigate the impact of the proposed GQIRIME, an experiment is conducted to evaluate the performance of our proposed algorithm. The simulation environment was developed using Cloudsim simulator [35]. Cloudsim is a toolkit that enables modelling and simulation of cloud systems [36]. The proposed GQIRIME is compared with five other popular meta-heuristic algorithms: standard RIME optimizer [1], Grey Wolf optimizer [37], Levy flight Jaya algorithm [38], Particle Swarm optimizer [39] and Min-Min algorithm [40]. The final performance was considered as an average of over 20 independent simulation runs.

A. Parameter Configuration

The RIME optimization algorithm consists of several parameters. For instance, parameter E determines whether the soft-rime operator should be applied. The GQIRME algorithm has one extra key parameter called the $levy$ variable. The $Levy$ variable essentially acts as the step size during the optimization process. A larger step size leads to a more diversified solution search, allowing the algorithm to explore the solution space more broadly. This parameter is instantiated directly from the Levy distribution, which governs the scale of the step size. For the purposes of this work, the levy variable is multiplied by either 1 or -1. This helps the algorithm to search in all directions. With this extra step, the algorithm can backtrack and look for better solutions in previously searched areas.
In this manuscript, the key parameters of all algorithms involved in the simulation were adopted as default values. Table 1 shows the key parameters of each algorithm.

TABLE 1. KEY PARAMETERS OF EACH ALGORITHM

Algorithm	Parameters
RIME	$w = 5$
GWO	-
PSO	$c1 = 2;\ c2 = 2$
LJA	$\beta = 1.8$

B. Experimental Metrics

For each algorithm, the following metrics are calculated to evaluate the performance of each algorithm [9]:

- Makespan

Makespan is equivalent to the maximum completion time of the latest task which is calculated using Eq. (14). The objective is to reduce the makespan to the minimum.

$$Makespan = \max_{t_j \in T} \left(FT(t_j) \right) \tag{14}$$

Where T is the set of all tasks, and FT is the finishing time of j-th task.

- Cost

The cost in cloud computing refers to the total payment made by users to cloud service providers in exchange for utilizing computing resources. While cloud providers aim to maximize their profits, users seek to minimize their expenses by efficiently utilizing these resources. The cost associated with processing a task t_j on a virtual machine v_i is determined based on the combined utilization of CPU and memory resources. The cost of processing t_j at v_i is calculated as follows [21]:

$$Cost(t_{j,i}) = \begin{pmatrix} \left(c_1 * ExeTime(t_{j,i})\right) + \\ \left(c_2 * ExeTime(t_{j,i})\right) \end{pmatrix} * b_{j,i} \tag{15}$$

Where c_1 and c_2 are CPU usage cost and memory usage cost per unit. CPU cost and memory cost per unit are considered $0.17/hour$ and $0.05/GB/hour$ [41]. $b_{j,i}$ is a boolean variable that determines if the j-th task is executed by i-th VM, and $ExeTime(t_{j,i})$ is the execution time of j-th task running on i-th VM. The total cost ($TCost$) is calculated as follows:

$$TCost = \sum_{i=1}^{m} \sum_{j=1}^{n} Cost(t_{j,i}) \tag{16}$$

- Total Execution Time

Total execution time is the cumulative running time of all tasks running on all VMs. The objective is to minimize total execution time.

$$TExeTime = \sum_{t_j \in T} ExeTime(t_j) \tag{17}$$

Where $ExeTime$ is the execution time of j-th task.

C. Experimental Setup

The core process of the experiment was simulated on Intel Core i5 2450m processor, 2.5 GHZ machine with 6 GB of

RAM. To simulate real world configurations, we have initialized tasks randomly. The simulation environment was considered heterogeneous, reflecting real-world scenarios where the configurations of virtual machines (VMs) can vary. The population size (S) and maximum number of iterations ($iter_{max}$) were set to 30 and 20 respectively. Table 2 Shows the simulation environment configuration.

TABLE 2. SIMULATION PARAMETERS

Parameters	**Value**
No of Data Centers	1
No of Hosts	2
No of Virtual Machines	30
VM MIPS	100-1000
No of CPUs	1-5
VM Policy	Time Shared
VM RAM Size	512 MB
No of Tasks	50-300
Task Length	1000-2000

D. Experimental Analysis

Fig .3 illustrates the comparative results of the algorithms based on makespan measurements. The performance of GQIRIME is 178, 399, 714, 743, 885, and 1240 s by makespan for 50, 100, 150, 200, 250, and 300 tasks, respectively. The algorithm obtains better results in terms of makespan in all cases, outperforming other algorithms. Results indicate that the proposed algorithm shows improvements over the original RIME optimizer. The GQIRIME algorithm's peak improvement is at 300 tasks where it shows an improvement of 68.5% over the original RIME optimizer. In terms of the makespan metrics, the LJA optimizer seems the closest competitor to the GQIRIME optimizer.

Fig .4 shows the total cost values reported by algorithms during simulation runs. Efficient usage of resources leads to lower cost values. At the same time, we want to map user tasks to less-cost heavy resources. Results of the tests in terms of cost show significant improvements of the proposed approach over other algorithms. The GQIRIME has obtained 0.3, 1.1, 1.5, 2.8, 3.0, and 4.6 ($) in terms of cost metrics. The GQIRIME optimizer outperforms other algorithms in all cases, respectively. In terms of cost metrics,, the GQIRIME optimizer has performed 68% better on average compared to the original RIME optimizer. Finally, the GQIRIME algorithm is evaluated in terms of total execution time metrics. The proposed algorithm has obtained better performance in all benchmarks compared to its counterparts. The total running time obtained by the GQIRIME optimizer is 7942, 35545, 132295, 196551, and 330948 s, respectively. Results indicate that the proposed algorithm performs better, exceeding the best results reported by its counterparts. Fig.5 shows the results obtained from the algorithms in terms of total execution time metrics. To summarize, the results obtained from the experiment show significant improvements of the GQIRIME optimizer over the original RIME optimizer in terms of task scheduling metrics.

To better investigate the results, Table 3 presents the amount of improvement gained by the GQIRIME optimizer compared to its counterparts in all cases.

TABLE 3. SIMULATION RESULTS

		Number of tasks					
Metric	GQIRIME vs.	50	100	150	200	250	300
Makespan	**RIME**	39.1%	24.4%	29.2%	35.4%	11.3%	**68.5%**
	GWO	10.5%	17.3%	39.6%	73.4%	38.6%	12.9%
	LJA	54.5%	28.4%	7.6%	4.9%	5.7%	18.7%
	PSO	31.7%	19.7%	54.2%	39.8%	53.7%	28.5%
	Min-Min	22.2%	13.8%	25.1%	25.2%	17%	28.8%
Cost	**RIME**	97.3%	75%	80.1%	43.8%	34.2%	83.8%
	GWO	44.7%	52.6%	89.7%	79.1%	65.4%	17.2%
	LJA	**128.9%**	73.2%	51.9%	13%	31.9%	25.9%
	PSO	89.4%	60.7%	107%	49.4%	76.8%	39.9%
	Min-Min	84.2%	56.2%	77.5%	37.8%	45.6%	44.4%
Total execution time	**RIME**	39.1%	24.4%	29.4%	35.2%	11.3%	**68.4%**
	GWO	10.8%	17.5%	39.8%	73.2%	38.8%	12.9%
	LJA	54.8%	28.4%	7.6%	4.8%	5.8%	18.6%
	PSO	32.1%	19.9%	54%	39.2%	53.9%	28.4%
	Min-Min	22.3%	14%	25.4%	25.1%	17.1%	28.5%

E. Comparison between GQIRIME and RIME

As we previously mentioned, enhancing the exploration and exploitation abilities of metaheuristics plays a pivotal role in the performance of them. In this work, we have enhanced the exploration and exploration ability of the RIME optimizer to make it suitable for the problem of task scheduling. Based on simulation results, compared with RIME, the GQIRIME has the following advantages.

a) A balance between exploration and exploitation is achieved. According to the results obtained, the RIME algorithm experiences a significant degradation in performance as the problem size increases. It struggles to maintain optimal efficiency, particularly when the problem's dimensionality becomes larger, leading to slower convergence and a reduced ability to explore the solution space effectively. In contrast, the GQIRIME has demonstrated the ability to maintain consistent performance across various problem sizes, showing no significant degradation even as the problem dimension grows. This highlights the stability and robustness of the GQIRIME in handling larger-scale problems.

b) Due to the incorporation of the GQI method, the proposed algorithm demonstrates enhanced exploration capabilities. This method enables the algorithm to explore regions of the solution space that may have been otherwise inaccessible. By improving the exploration phase, the algorithm can effectively search through more diverse solution spaces, resulting in better performance, particularly in larger and more complex problems.

c) The proposed algorithm utilizes the same control parameters as the original algorithm, with no additional control parameters introduced.

d) Based on the results, the GQIRIME demonstrates a better convergence rate. The proposed GQIRIME was able to find superior solutions compared to the original algorithm in less time in all cases.

e) The time complexity of the proposed GQIRIME has not increased and emains the same, while achieving better performance.

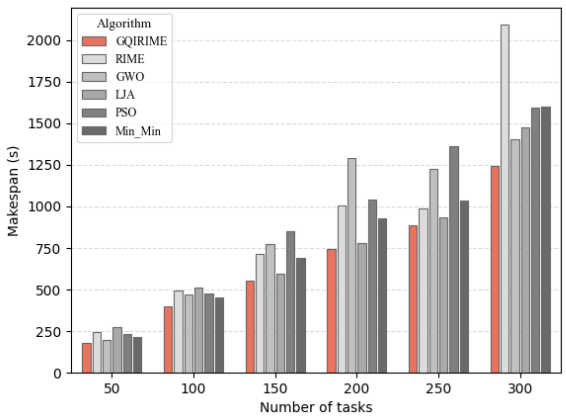


Fig. 3. The comparative results in terms of makespan

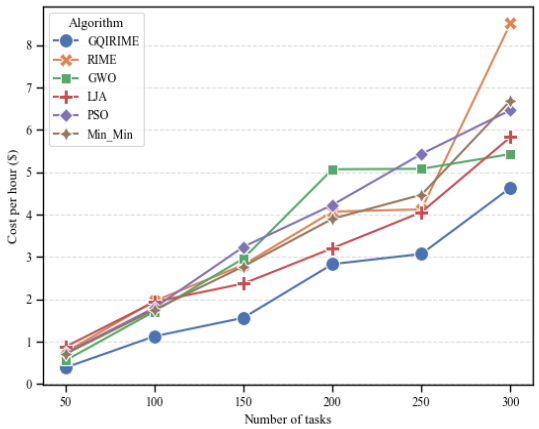


Fig. 4. Comparison of algorithms based on total cost

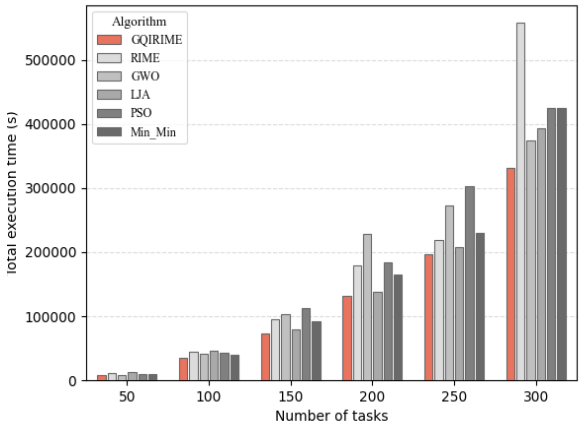


Fig. 5. Obtained results in terms of total execution time

VI. CONCLUSION

Task scheduling is considered a major challenge in the cloud computing paradigm, because requests come from heterogeneous sources and inefficient task scheduling in the cloud environment leads to the degradation of the quality of service and violates the SLA. Therefore, an effective schedule should be used in the cloud computing model. In this paper, we proposed GQIRIME task scheduling algorithm that is an enhanced version of the RIME optimizer. We implemented an extensive simulation in the Cloudsim toolbox. The results analysis demonstrates that our proposed algorithm outperforms other methods and significantly improves key performance metrics in task scheduling such as makespan, cost, and total execution time. As future work directions, scheduling tasks along with task dependency modelling and considering their priority in order to reduce the average delay and complete tasks within a certain time limit is one of the research topics that can be studied and investigated.

REFERENCES

[1] P. Mell, "The NIST definition of cloud computing," *NIST Special Publ.*, Special Publ. 800-145, 2011, doi: 10.6028/NIST.SP.800-145.

[2] M. H. Sayadnavard, A. T. Haghighat, and A. M. Rahmani, "A multi-objective approach for energy-efficient and reliable dynamic VM consolidation in cloud data centers," *Eng. Sci. Technol. Int. J.*, vol. 26, Art. no. 100995, 2022, doi: 10.1016/j.jestch.2021.06.004.

[3] C. Chandrashekar, P. Krishnadoss, V. K. Poornachary, and B. Ananthakrishnan, "MCWOA scheduler: Modified chimp-whale optimization algorithm for task scheduling in cloud computing," *Comput. Mater. Continua*, vol. 78, no. 2, pp. 2593–2616, 2024, doi: 10.32604/cmc.2024.046304.

[4] M. S. Bali, K. Gupta, D. Gupta, G. Srivastava, S. Juneja, and A. Nauman, "An effective technique to schedule priority aware tasks to offload data on edge and cloud servers," *Meas. Sensors*, vol. 26, Art. no. 100670, 2023, doi: 10.1016/j.measen.2022.100670.

[5] J. Singh and N. K. Walia, "A comprehensive review of cloud computing virtual machine consolidation," *IEEE Access*, vol. 11, pp. 132102–132117, 2023, doi: 10.1109/ACCESS.2023.3335289.

[6] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy, survey, and issues of cloud computing ecosystems," in *Cloud Computing: Principles*, Systems and Applications, London, UK: Springer, 2010, pp. 21–46, doi: 10.1007/978-1-84996-241-4_2.

[7] M. Adil, S. Nabi, M. Aleem, V. G. Diaz, and J. C. W. Lin, "CA-MLBS: Content-aware machine learning based load balancing scheduler in the cloud environment," *Expert Syst.*, vol. 40, no. 4, Art. no. e13150, 2023, doi: 10.1111/exsy.13150.

[8] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: A survey on software technologies," *Cluster Comput.*, vol. 26, no. 3, pp. 1845–1875, 2023, doi: 10.1007/s10586-022-03713-9.

[9] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. I. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS ONE*, vol. 12, no. 5, Art. no. e0176321, 2017, doi: 10.1371/journal.pone.0176321.

[10] M.-L. Chiang, H.-C. Hsieh, W.-C. Tsai, and M.-C. Ke, "An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network," in *Proc. 8th Int. Conf. Awareness Sci. Technol. (iCAST)*, Taichung, Taiwan, 2017, pp. 290–295, doi: 10.1109/ICAwST.2017.8256478.

[11] H. Emami, "Cloud task scheduling using enhanced sunflower optimization algorithm," *ICT Express*, vol. 8, no. 1, pp. 97–100, 2022, doi: 10.1016/j.icte.2021.08.010.

[12] V. Tomar, M. Bansal, and P. Singh, "Metaheuristic algorithms for optimization: A brief review," *Eng. Proc.*, vol. 59, no. 1, Art. no. 238, 2024, doi: 10.3390/engproc2024059238.

[13] K. Rajwar, K. Deep, and S. Das, "An exhaustive review of the metaheuristic algorithms for search and optimization: Taxonomy, applications, and open challenges," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13187–13257, 2023, doi: 10.1007/s10462-023-10470-y.

[14] L. Guo, L. Liu, Z. Zhao, and X. Xia, "An improved RIME optimization algorithm for lung cancer image segmentation," *Comput. Biol. Med.*, vol. 174, Art. no. 108219, 2024, doi: 10.1016/j.compbiomed.2024.108219.

[15] M. Abdel-Salam, G. Hu, E. Çelik, F. S. Gharehchopogh, and I. M. El-Hasnony, "Chaotic RIME optimization algorithm with adaptive mutualism for feature selection problems," *Comput. Biol. Med.*, vol. 179, Art. no. 108803, 2024, doi: 10.1016/j.compbiomed.2024.108803.

[16] Y. Li, D. Zhao, C. Ma, J. Escorcia-Gutierrez, N. O. Aljehane, and X. Ye, "CDRIME-MTIS: An enhanced rime optimization-driven multi-threshold segmentation for COVID-19 X-ray images," *Comput. Biol. Med.*, vol. 169, Art. no. 107838, 2024, doi: 10.1016/j.compbiomed.2023.107838.

[17] D. Yousri, A. Fathy, H. E. Farag, and E. F. El-Saadany, "Optimal dynamic reconfiguration of thermoelectric generator array using RIME optimizer," *Appl. Therm. Eng.*, vol. 238, Art. no. 122174, 2024, doi: 10.1016/j.applthermaleng.2023.122174.

[18] S. S. Murad, A. A. Al-Azzawi, and M. A. Bakar, "Optimized Min-Min task scheduling algorithm for scientific workflows in a cloud environment," *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 2, pp. 480–506, 2022.

[19] I. L. Imene, S. Sihem, O. K. Okba, and M. Benmohammed, "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, 2022, doi: 10.1016/j.jksuci.2021.08.010.

[20] G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," *ICT Express*, vol. 5, no. 2, pp. 110–114, 2019, doi: 10.1016/j.icte.2018.02.002.

[21] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022, doi: 10.1109/ACCESS.2022.3149959.

[22] C. Chandrashekar, P. Krishnadoss, V. K. Poornachary, B. Ananthakrishnan, and K. Rangasamy, "HWACOA scheduler: Hybrid weighted ant colony optimization algorithm for task scheduling in cloud computing," *Appl. Sci.*, vol. 13, no. 6, Art. no. 3433, 2023, doi: 10.3390/app13063433.

[23] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "An efficient trust-aware task scheduling algorithm in cloud computing using firefly optimization," *Sensors*, vol. 23, no. 3, Art. no. 1384, 2023, doi: 10.3390/s23031384.

[24] S. D. S. Mustapha and P. Gupta, "Fault aware task scheduling in cloud using min-min and DBSCAN," *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 68–76, 2024, doi: 10.1016/j.iotcps.2023.12.003.

[25] G. Saravanan, S. Neelakandan, P. Ezhumalai, and S. Maurya, "Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing," *J. Cloud Comput.*, vol. 12, no. 1, Art. no. 24, 2023, doi: 10.1186/s13677-023-00401-1.

[26] A. Kamalinia and A. Ghaffari, "Hybrid task scheduling method for cloud computing by genetic and PSO algorithms," *J. Inf. Syst. Telecommun.*, vol. 4, no. 4, pp. 271–281, 2016.

[27] A. Esfandi, J. Akbari Torkestani, A. Karimi, and F. Zarafshan, "TPALA: Two phase adaptive algorithm based on learning automata for job scheduling in cloud environment," *J. Inf. Syst. Telecommun.*, vol. 1, no. 1, pp. 29–36, 2024, doi: 10.30495/jist.2024.703768.

[28] M. Hosseini Sayadnavard and A. Toroghi Haghighat, "Toward an enhanced dynamic VM consolidation approach for cloud datacenters using continuous time Markov chain," *J. Inf. Syst. Telecommun.*, vol. 4, no. 4, pp. 236–244, 2019.

[29] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012, doi: 10.1016/j.future.2011.04.017.

[30] H. Su, D. Zhao, L. He, and Q. Zhang, "RIME: A physics-based optimization," *Neurocomputing*, vol. 532, pp. 183–214, 2023, doi: 10.1016/j.neucom.2023.02.010.

[31] M. Zhang, H. Lyu, H. Bian, and N. Ghadimi, "Improved chaos grasshopper optimizer and its application to HRES techno-economic evaluation," *Heliyon*, vol. 10, no. 2, Art. no. e24321, 2024, doi: 10.1016/j.heliyon.2024.e24321.

[32] W. S. Xiao, G. X. Li, C. Liu, and L. P. Tan, "A novel chaotic and neighborhood search-based artificial bee colony algorithm for

solving optimization problems," *Sci. Rep.*, vol. 13, no. 1, Art. no. 20496, 2023, doi: 10.1038/s41598-023-47701-x.

[33] W. Zhao, L. Wang, Z. Zhang, S. Mirjalili, N. Khodadadi, and Q. Ge, "Quadratic interpolation optimization (QIO): A new optimization algorithm based on generalized quadratic interpolation and its applications to real-world engineering problems," *Comput. Methods Appl. Mech. Eng.*, vol. 417, Art. no. 116446, 2023, doi: 10.1016/j.cma.2023.116446.

[34] M. Chawla and M. Duhan, "Levy flights in metaheuristics optimization algorithms–A review," *Appl. Artif. Intell.*, vol. 32, no. 9–10, pp. 802–821, 2018, doi: 10.1080/08839514.2018.1525856.

[35] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, 2011, doi: 10.1002/spe.995.

[36] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Leipzig, Germany, 2009, pp. 1–11, doi: 10.1109/HPCSIM.2009.5256777.

[37] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[38] G. Iacca, V. C. dos Santos Junior, and V. V. de Melo, "An improved Jaya optimization algorithm with Lévy flight," *Expert Syst. Appl.*, vol. 165, Art. no. 113902, 2021, doi: 10.1016/j.eswa.2020.113902.

[39] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, Perth, Australia, 1995, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[40] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810–837, 2001, doi: 10.1006/jpdc.2001.1714.

[41] 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, doi: 10.1109/ACCESS.2015.2508940.

مجله علمی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
شناسه دیجیتال: 10.22034/jscit.2025.220413
جلد ۱۴، شماره ۱، بهار ۱۴۰۴ ، صفحه ۵۷-۶۷
دریافت: ۱۴۰۳/۱۱/۱۱، بازنگری: ۱۴۰۴/۰۱/۲۱، پذیرش: ۱۴۰۴/۰۲/۱۴

الگوریتم بهینه‌سازی RIME آشوب‌ناک مبتنی بر درون‌یابی درجه دوم تعمیم‌یافته با پرواز لوی برای زمان‌بندی وظایف در رایانش ابری

محمدرضا صبی‌پور[۱]، منیره حسینی صیادنورد*[۲]
۱- گروه مهندسی کامپیوتر، واحد بندرانزلی، دانشگاه آزاد اسلامی، بندرانزلی، ایران.
*۲- گروه مهندسی کامپیوتر، واحد بندرانزلی، دانشگاه آزاد اسلامی، بندرانزلی، ایران.
[1]mohammadreza.sabipour@iau.ac.ir, [2*]mo.sayadnavard@iau.ac.ir

* نشانی نویسنده مسئول: منیره حسینی صیادنورد، بندرانزلی، دانشگاه آزاد اسلامی واحد بندرانزلی، گروه مهندسی کامپیوتر.

چکیده- زمان‌بندی وظایف در رایانش ابری یک مسئله اساسی مرتبط با عملکرد سیستم و رضایت کاربر است. با توجه به اینکه سازمان‌ها به طور فزاینده‌ای به زیرساخت‌های ابری برای مدیریت و اجرای وظایف محاسباتی خود تکیه می‌نمایند، زمان‌بندی مؤثر وظایف بسیار اهمیت پیدا می‌کند. هدف اصلی زمان‌بندی وظایف، تخصیص وظایف ارسال شده به منابع مناسب موجود با حفظ کیفیت خدمات است. یک الگوریتم زمان‌بندی کارا باید بتواند مدت زمان اجرای وظایف را به عنوان یک معیار اساسی در خصوص عملکرد سیستم‌های رایانش ابری کاهش دهد. در این مقاله، یک الگوریتم زمان‌بندی جدید و کارآمد به نام GQIRIME مبتنی بر الگوریتم بهینه‌سازی RIME به هدف کاهش مدت زمان اجرا، هزینه و زمان اجرای کل سیستم ارائه شده است. همچنین یک استراتژی بهره‌برداری پیشرفته بر اساس درون‌یابی درجه دوم تعمیم‌یافته و پرواز لوی جهت افزایش نرخ همگرایی و اجرای جستجوی قوی پیشنهاد شده است. علاوه بر این، الگوریتم پیشنهادی با نگاشت آشوب ترکیب شده است تا جمعیت اولیه متنوع‌تری به دست آید. کارایی روش پیشنهادی با استفاده از ابزار Cloudsim ارزیابی شده است. تحلیل نتایج نشان می‌دهد که الگوریتم پیشنهادی از سایر روش‌ها بهتر عمل کرده و به طور قابل‌توجهی معیارهای کلیدی در زمان‌بندی وظایف را بهبود می‌بخشد. بر اساس نتایج آزمایش‌ها، الگوریتم زمان‌بندی پیشنهادی هزینه، مدت زمان اجرا و زمان اجرای کل وظایف را کاهش می‌دهد. نتایج نشان‌دهنده بهبود ۲۹,۳ درصدی در زمان اتمام کارها، بهبود ۶۱,۸ درصدی هزینه و بهبود ۲۹,۴ درصدی زمان کلی اجرا در مقایسه با روش‌های مشابه است.

واژه‌های کلیدی: زمان‌بندی وظایف، الگوریتم بهینه‌سازی RIME، رایانش ابری، درون‌یابی درجه دوم، فرا اکتشافی.