Reliability in Cloud Computing Applications with Chaotic Particle Swarm Optimization Algorithm

Wenli WANG*, Yanlin BAI
School of Artificial Intelligence, Zhengzhou Railway Vocational & Technical College, Zhengzhou 450000, China

***Abstract*—In recent years, IT managers of large enterprises and stakeholders have turned to cloud computing due to the benefits of reduced maintenance costs and security concerns, as well as access to high-performance hardware and software resources. The two main challenges that need to be considered in terms of importance are ensuring that everyone has access to services and finding efficient allocation options. First, especially with software services, it is very difficult to predict every service that may be needed. The second challenge is to select the best independent service among different providers with features related to application reliability. This paper presents a framework that uses the particle swarm optimization technique to optimize reliability parameters in distributed systems applications. The proposed strategy seeks a program with the best service and a high degree of competence. Although this method does not provide an exact solution, the particle swarm optimization algorithm reaches a result close to the best solution and reduces the time required to adjust the parameters of distributed systems applications. The results of the work have been compared with the genetic algorithm and it has been shown that the PSO algorithm has a shorter response time than both the genetic algorithm and the PSO. Also, the PSO algorithm shows strong stability and ensures that the solution obtained from the proposed approach will be close to the optimal solution.**

Keywords*—Reliability; cloud computing; chaotic particle swarm optimization algorithm; distributed systems

I. INTRODUCTION

The augmentation of available services leads to a corresponding increase in the proliferation of services that possess comparable functionalities across several servers [1]. These comparable services are situated in distinct geographical locations and exhibit varying degrees of reliability based on different characteristics [2]. Due to this rationale, service composition employs suitable methodologies to choose an atomic service from a pool of identical services hosted on distinct servers, with the aim of attaining the utmost level of dependability based on specific requirements and priorities [3]. The user has been obtained. Because end user requirements and accessible services are always changing, service composition architecture in the cloud environment must be flexible and capable of running independently [4]. Hence, the selection of appropriate and efficient elementary services for integration into complex composite services constitutes a significant concern within this domain [5]. The service composition challenge in cloud computing refers to the determination of appropriate simple atomic services that, when combined, satisfy the functional and reliability requirements of complex services, as dictated by the end user's needs [6]. The complexity of service composition in cloud computing is attributed to the multitude of influential factors and the extensive range of basic services offered by numerous providers in the cloud pool. As a result, this problem is classified as NP-hard [7].

The issue of software reliability in distributed systems is a significant concern for both software providers and consumers that rely on such software [8]. Numerous models have undergone scrutiny and assessment with regards to their trustworthiness within the context of large-scale commercial projects [9]. Notably, these models have been subject to meticulous examination in the domains of e-government, e-commerce, multimedia services, and other relevant scenarios [10]. However, the presence of dependability issues persists in software and systems [11]. Cloud environments have a dynamic nature characterized by both sporadic and deliberate modifications. The aforementioned modifications provide cloud computing with a range of issues within the context of distributed systems [12]. Several issues associated with applications in cloud computing have been identified [13]. a) The dynamic contracting of cloud service providers: an analysis of pricing policies employed by various service providers. The determination of costs for services is contingent upon the interplay between supply and demand factors. Hence, it is imperative to establish a method that facilitates the updating of the specification table pertaining to the range of resources that are now accessible. b) Resolving Insufficient Cloud Resources: The intermediary's decision on the ideal cloud service is contingent upon the presence of comprehensive and up-to-date information regarding the available services [14]. The occurrence of several alterations in service features has the potential to result in the inadvertent deletion or loss of certain data [15].

A significant portion of the research conducted on cloud services often results in suboptimal outcomes, necessitating the completion of the service within a limited timeframe [16]. Hence, it is imperative to put forth certain methodologies aimed at resolving the issue of partial optimization and enhancing the rate of convergence of the algorithm [17]. The chaotic evolutionary algorithm is founded upon the principles of optimization and the utilization of chaos operators, thereby synergistically integrating their respective benefits [18]. The randomness technique incorporates the concepts of unpredictability, initial sensitivity, and chaos operator to establish a mapping between the chaotic variable and a domain of linear optimization variables [19]. By employing this approach, the issue of stagnant search is mitigated and the absence of an optimization mechanism is addressed. Consequently, it enhances the algorithm's diversity and overall optimization [20].

In choosing the particle swarm optimization (PSO) method based on chaos theory for the current research, there are several reasons that indicate the importance and necessity of using this method in checking the reliability of cloud computing programs. First, PSO is an optimization algorithm based on collective intelligence, which is known for its simple structure and high efficiency in solving complex optimization problems. By imitating the social behavior of birds or fish, this algorithm quickly converges towards optimization and has the ability to search more widely in the search space. The use of chaos theory in PSO is very effective to prevent the algorithm from falling into local optima and improve its convergence rate. By adding controlled uncertainty to the search process, chaos theory increases the variety of responses and helps the algorithm achieve more optimal results.

In this study, we aim to examine the dependability of cloud computing applications by utilizing the particle swarm optimization algorithm grounded in chaos theory. This approach is anticipated to offer a thorough exploration of the problem-solving domain and enhance the accuracy of predicting the reliability of cloud services. In order to fulfill the requirements of academic discourse, it is necessary to revise the user's text to conform to the standards of formal the proposed method will be implemented using the MATLAB software environment, with the aim of achieving an optimal solution in terms of both convergence and stability. In summary, the writers of this research have made the following contributions:

- The application of chaos theory in predicting the reliability of cloud computing applications allows for a more extensive exploration of the problem area.
- Identifying a cloud service that offers near-optimal performance while considering the reliability of the service.

The subsequent sections of the article are structured in the following manner. The second half of the document provides a comprehensive review of prior research and scholarly contributions. Section III presents the formulation of the problem. In Section IV, the proposed method is expounded upon. Section V of the paper presents an evaluation and simulation of the proposed solution. Subsequently, Section VI provides the conclusion and outlines potential avenues for future research.

II. Related Works

A proposed strategy has emerged in response to the increasing significance of networks in the amalgamation of cloud services, which takes into account the distinct reliability of applications and network services [21]. In order to achieve this objective, the actual network delay between the desired services and their users is represented using a low time complexity model, enabling the selection of the service with the lowest delay time. The introduction of a reliability equation by researchers enables the calculation of application dependability, delay, and transmission rate. In the final stage of the methodology, the selection algorithm was devised to implement the proposed models using the genetic algorithm. The outcomes of this algorithm were then compared to those of Dijkstra's algorithm and random selection. The findings of this intriguing study can be enhanced through the utilization of real-world datasets.

The authors of this study have presented an enhanced genetic algorithm [22] for the service provider system, taking into account self-adaptation. In this algorithm, the traditional competitive selection method for choosing individuals for intersection and mutation operators has been replaced with a clonal selection algorithm [23]. A well-established methodology has been employed. The primary article lacks a comprehensive discussion of the researchers' efforts in self-adaptation, as it fails to include specific details about the suggested algorithm and the experimental outcomes.

The utilization of game theory by researchers has led to the development of a service combination algorithm that is founded on service level agreement [24]. This study encompasses four distinct components inside the agreement, namely: the primary details of the agreement, information pertaining to service providers and users, specifications about the type and dimensions of the service, and a comprehensive set of obligations for applications. The process of establishing an agreement involves the consideration of service composition as a dynamic multi-player game, referred to as the proposal game. In this game, the sellers and consumers of the service act as players with the objective of attaining their respective aims. Within the context of this competitive framework, it is imperative for every consumer to declare a price for each desired service, taking into account the relevant parameters and the suggested price set by other consumers. Subsequently, sellers have the autonomy to select their service based on the level of quality requested, which is duly influenced by the suggested price. Contained inside the mutually agreed upon and formally executed agreement. The method's reliability is constrained by its narrow scope, since it lacks comparative analysis with alternative approaches and fails to incorporate real-world data sets for comparison.

The authors have introduced a variant of the chaotic optimization algorithm that operates in parallel, with the aim of addressing the issue of application services [25]. The length of the sequence was dynamically altered by the researchers, taking into consideration the evolutionary position of the answer. The researchers also employed the roulette wheel selection process as a preliminary step, followed by the application of the chaos operator, in order to mitigate the presence of randomly generated unsuitable solutions and avoid their detrimental effects. Given that a primary objective of this study is to minimize the duration of execution, the parallelization of the suggested algorithm is also taken into account. In order to accomplish this objective, a comprehensive connection architecture is selected based on its superior searchability and message transmission interface [26]. A novel migration technique, known as reactive path migration, has been recently devised and implemented to mitigate the communication overhead associated with fully connected topologies. In comparison to the genetic algorithm, chaos genetic algorithm, and chaos optimization, the method given in this study has demonstrated superior outcomes in terms of both the best fit achieved and the execution time required.

In [27] present a novel paradigm for adaptive service selection in the context of mobile cloud computing. This

framework facilitates the prompt extraction of consumer preferences upon receipt of a request. Subsequently, utilizing the Euclidean distance metric, the customer priority services that exhibit the shortest distances are identified and subsequently recommended to the service adapter. Ultimately, the service adapter determines the optimal service from the available options for the consumer based on the compatibility of the underlying device and the efficacy of the service alternative. The service adapter module incorporates a fuzzy known map model to facilitate the achievement of context matching service based on input information. One limitation of this approach is that the offered framework is applicable solely for the purpose of selecting a singular service. Furthermore, it is worth noting that this particular strategy has not been subjected to comparative analysis with alternative methodologies.

On a pay-as-you-go basis, cloud computing provides worldwide access to utility-based information technology services, with many uses in the commercial, academic, and consumer spheres. But data centers that host cloud applications use a lot of energy, which means they cost a lot to run and pollute the environment with their carbon emissions. Powerful servers that use a lot of energy and related peripherals are necessary for these centers to manage the daily influx of requests from various users. In order to lower energy consumption in data centers, resource efficiency is key. Focusing on energy reduction and load prediction, this research adopts a novel hybrid approach for dynamic resource allocation in the cloud. Specifically, they have migrated virtual machines using an ant colony optimization technique and utilized neural fuzzy networks for load prediction [39].

When it comes to cloud computing, the problem of work scheduling directly affects the quality of services provided. Allocating tasks to available resources according to demand is known as task scheduling. Finding the optimal allocation plan to get more done in less time is the objective of this NP-hard problem. The job scheduling problem has been addressed by several approaches. To fix this, the authors of [40] suggest an IPSO algorithm, which stands for enhanced particle swarm optimization. The original Particle Swarm Optimization (PSO) technique for cloud work scheduling is optimized using a multi-adaptive learning strategy to reduce execution time. The proposed MALPSO method establishes two particle types—normal particles and local best particles—during the first population stage. The population's variety is decreasing and the likelihood of reaching the local optimum is increasing at this stage. Distance, load balance, stability, and efficiency are the four metrics used to evaluate alternative algorithms in this study. In addition, the CEC 2017 benchmark is used to assess the suggested method. We can solve the problem faster and achieve the best answer for most of the criteria using the provided strategy compared to what is currently known.

Our proposed work has significant differences from the works in the "Related Works" section. Unlike previous methods that mainly used genetic algorithms, game theory, and classical optimization algorithms, we use the combination of particle swarm optimization (PSO) algorithm with chaos theory. This combination not only has the ability to improve convergence and stability, but also effectively solves the local optimization problem. While previous methods such as genetic algorithms and game theory compare and select the best services based on complex models and with real data, our method uses a collective approach that brings a significant improvement in performance and prediction accuracy.

In addition, our proposed approach using chaos theory and dynamic population size adjustment has been able to overcome the problems in classical PSO, such as being stuck in local optima and slow convergence. While some existing methods have only focused on optimizing the execution time of the algorithm or improving the quality of the services provided, our approach is a more comprehensive model by focusing on the stability and reliability of cloud services and it provides more efficiency that can be more widely used in real scenarios.

III. Problem Formulation

The issue pertaining to reliability-aware cloud services in applications involves identifying a collection of potential cloud services that possess varying performance attributes. These services must fulfill two criteria: firstly, they must adhere to the limitations established by the user, and secondly, they must satisfy an objective function. To optimize refers to the process of maximizing efficiency or effectiveness in a given context. In this section, the aforementioned issue is explicitly articulated. One instance of the issue pertaining to the integration of cloud services while considering service reliability can be officially articulated as follows:

A service composition request is represented as a workflow modeled using a directed acyclic graph G= (V, E).

- $V = \{T_1, T_2, \ldots, T_n\}$, where n denotes the workflow's job count.
- E: The group of edges indicating the order in which tasks are being completed.
- The process for every T_i $(1 \le i \le n)$ job includes a set of nomination services called $CS_i = \{CS_i^1, CS_i^2, \ldots, CS_i^{m_i}\}$, where $CS_i^j (1 \le j \le m_i)$a cloud nomination service is.
- *Mi*: the entire number of potential workers that are willing to take up T_i jobs.
- Every potential service A property of cloud services' service dependability is represented by Q_l $(1 \le l \le K)$, one of the various sets of service reliability information $QoS_i^j = \{Q_1, Q_2, \ldots, Q_K\}$that CS_i^jhas.
- The service reliability warehouse houses service reliability data pertaining to cloud services.
- K: the quantity of cloud service-related service reliability features included in the service reliability model.

Given the aforementioned context, the primary aim of the reliability-aware service composition problem is to identify a cloud composite service that is near-optimal [28]. This objective is achieved by ensuring that the selected service exhibits a high level of reliability.

$$\forall j = 1 \ldots K \begin{cases} \Sigma_{i=1}^{n} S_i.Q_j < C_j if Q_j is additive \\ \Pi_{i=1}^{n} S_i.Q_j > C_j if Q_j is multiplicative \end{cases} \quad (1)$$

A. Service Reliability Model

The dataset utilized in the suggested methodology for determining service dependability parameter values in applications is sourced from Al-Masri and colleagues. The provided dataset serves as a fundamental resource for academics in the field of service. The dataset comprises a collection of 2507 cloud services together with their corresponding measurements of service dependability in various applications. The authors of the study have utilized their proposed service broker framework to measure the values of service dependability parameters [29]. The QWS dataset comprises individual records that encompass the values of ten distinct parameters associated with each cloud service. The initial eight elements inside each record pertain to service dependability metrics that were assessed over a duration of six days using the cloud service broker framework. The service reliability numbers within the dataset represent the mean measurements obtained during the specified time interval. Table I presents a concise overview of the eight service dependability metrics, including a straightforward description for each.

Upon meticulous examination of the reliability parameters presented in Table I, it becomes evident that the response, best practice, and documentation parameters exhibit a consistent value across multiple service calls during execution. Consequently, in light of this observation, these two parameters are disregarded, and the values of the remaining six service reliability parameters are utilized.

B. Service Reliability Parameter Values are Normalized

Various service dependability characteristics associated with a cloud service are assessed using distinct units. In order to compute the objective function, it is necessary to ensure that all of these parameters are measured using a consistent scale [30]. In light of this matter, it is imperative to standardize the values of all service dependability parameters on a consistent scale. The normalizing of service dependability metrics enables the establishment of a standardized metric for evaluating their values. To achieve this objective, a commonly employed method involves normalizing the values of all parameters within the range of zero to one. The criteria pertaining to service reliability can be classified into two distinct categories: those aimed at maximizing reliability and those aimed at minimizing it. Maximization parameters refer to parameters that are intended to be maximized, while minimization parameters refer to parameters that are intended to be minimized. Relations (2) and (3) illustrate the normalization principles for maximizing and minimizing parameters, correspondingly.

TABLE I. AN EXPLANATION OF THE PARAMETERS FOR SERVICE RELIABILITY FOUND IN THE QWS DATASET

Parameters	description	unit
Ability to succeed	The number of responses to the number of request messages	%
the answer	The extent to which the WSDL document conforms to the WSDL specification	%
best way	The degree to which a service conforms to the base WS-I profile	%
Delay	The amount of time it takes for the server to process a request	Millisecond
Documentation	Measuring documentation (descriptive tags) in WSDL	%
response time	The time it takes to send a request and receive a response	Millisecond
accessibility	The number of successful calls over the total number of calls	%
Throughput	The total number of calls for a given time period	Calls per second

$$N_{CS.Q^i} = \begin{cases} \dfrac{Q^i_{\max} - CS.Q^i}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 & Q^i_{\max} = Q^i_{\min} \end{cases} \quad (2)$$

$$N_{CS.Q^i} = \begin{cases} \dfrac{CS.Q^i - Q^i_{\min}}{Q^i_{\max} - Q^i_{\min}} & Q^i_{\max} \neq Q^i_{\min} \\ 1 & Q^i_{\max} = Q^i_{\min} \end{cases} \quad (3)$$

In the aforementioned relationships, $CS.Q^i$ represents the value assigned to the i-th parameter of service reliability pertaining to the candidate service [31]. The normalized value of CS and N_{CS} is denoted as CS and $N_{CS.Q^i}$, respectively. Additionally, Q^i_{max} and Q^i_{min} represent the upper and lower bounds of the *i-th* parameter across all services.

IV. PROPOSED METHOD

Given the fact that the issue of identifying cloud services that are cognizant of service reliability falls under the classification of NP-Hard issues, many approaches to discovery can be employed in order to address this challenge. The primary objective of this paper is to employ the chaotic particle optimization technique in order to identify a dependable cloud service. Collective intelligence is a highly potent optimization strategy that relies on the behavior of a group. The particle optimization algorithm is a social search algorithm that is designed based on the collective behavior observed in flocks of birds. Initially, this method was employed to uncover the underlying patterns that regulate the concurrent flying of avian species, as well as the abrupt alterations in their trajectory and the ideal configuration of the flock. The particle optimization algorithm involves the movement of particles inside the search space. The relocation of particles within the search space is influenced by both their own experiences and knowledge, as well as the experiences and knowledge of their neighboring particles. Hence, the alternative configuration of particle mass influences the manner in which a particle is sought. The outcome

of simulating this social behavior manifests as a search process wherein particles exhibit a tendency to converge towards regions of success. Particles acquire knowledge from one another and navigate towards their most optimal neighbors. The particle optimization algorithm operates on the premise that each particle in the search space determines its position based on the best position it has previously occupied. Selects and optimizes the most favorable position within its surrounding vicinity. Through the analysis and refinement of this technique within computer systems, as well as its adaptation to expert and intelligent systems, it becomes feasible to apply it in addressing a wide range of optimization problems. The use of the particle swarm optimization algorithm is anticipated to enhance the dependability of the cloud service generated. The application of chaos theory has also been employed in addressing the issue of local optima and enhancing the rate of convergence.

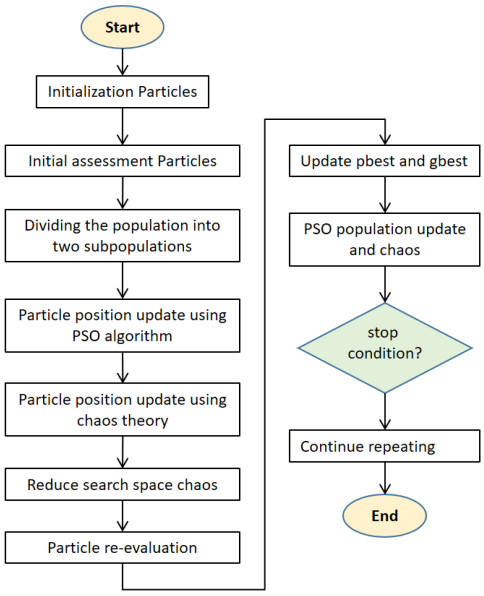

Fig. 1. Process flowchart of the proposed method.

The flow/block diagram in Fig. 1 shows the proposed research process to improve the reliability of cloud services using particle swarm optimization (PSO) algorithm and chaos theory. At first, the particles are initialized and an initial evaluation is done on them. Then the particle population is divided into two subpopulations: one using the PSO algorithm and the other using the chaos theory. The position of particles in each subpopulation is updated and the chaotic search space is reduced. After that, the particles are re-evaluated and the local best (pbest) and global best (gbest) are updated. This update and evaluation process continues until a stop condition is met. Finally, the algorithm ends by finding a near-optimal solution. This combined method of PSO and chaos theory helps to improve the convergence and stability of the algorithm.

The particle optimization algorithm initiates its operation by generating a set of particles within the search space. Each particle represents the position of a potential solution to a given problem, specifically in the context of a composite service within a cloud environment. The starting positions of the particles within the node are determined randomly [32]. The program will subsequently conduct a search for the optimal place based on the highest merit value. The subsequent section outlines the sequential procedures involved in attaining the most advantageous location, or in other words, elucidates the process by which the algorithm progressively approaches a solution that is close to optimal. Eq. (4) is utilized to represent the position of the *i-th* particle.

$$X_i = (x_{i1}, \ldots, x_{id}, \ldots x_{iD}) \quad (4)$$

Eq. (5) is utilized to retain and present the prior optimal position of the i-th particle.

$$P_i = (p_{i1}, \ldots, p_{id}, \ldots p_{iD} \quad (5)$$

The term used to refer to this concept is known as *pbest* [33]. The optimal solution within a population of particles is sometimes referred to as the global best *(gbest)*. The velocity of the *i-th* particle is also represented by the vector V_i, as depicted in Eq. (6):

$$V_i = (v_{i1}, \ldots, v_{id}, \ldots v_{iD}) \quad (6)$$

The fundamental principle underlying particle swarm optimization involves the manipulation of the location and velocity of individual particles towards their personal best (pbest) and global best (gbest) values, as described by Eq. (7) and (8).

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (p_{gd} - x_{id}) \quad (7)$$

$$x_{id} = x_{id} + v_{id} \quad (8)$$

In this context, the symbol "w" represents the inertial weight, while the numbers c_1 and c_2 denote the acceleration constants [34]. Additionally, rand () refers to a random number generator that produces values uniformly distributed throughout the interval [0, 1]. Algorithm 1 presents the pseudocode for the fundamental particle swarm optimization.

ALGORITHM 1: PSEUDO CODE OF BASIC PSO

```
01: Start
02:  Initialize particle swarm
03:  While (number of iterations, or the stopping criterion is not met)
04:   Evaluate fitness of particle swarm
05:   For n = 1 to number of particles
06:    Find pbest
07:    Find gbest
08:    For d = 1 to number of dimensions of particle
09:     Update the position of particles via equations (3-4) and (3-5)
10:    End For
11:   End For
12:  End While
13:  Stop
```

The fundamental particle swarm optimization approach has demonstrated commendable efficacy in addressing intricate challenges. However, notwithstanding this, it is afflicted by the issue of succumbing to the local optimum trap [35]. There exist various approaches for resolving this issue, with one particularly significant option being the utilization of chaos theory. According to the dictionary, the term "chaos" refers to a condition characterized by a lack of organization and clarity. In the realm of scientific discourse, a universally accepted definition for the concept in question remains elusive. The

concept of chaos is commonly acknowledged as a phenomenon characterized by distinct and discernible patterns within unpredictable circumstances, often referred to as "order in chaos." Chaos theory is a well-established theoretical framework that may be formulated based on a set of deterministic principles and mathematical equations. According to the principles of chaos theory, the future is entirely governed by preceding events. The logical equation, renowned for its association with chaos theory, holds significant prominence and finds application within the proposed methodology. Eq. (9) represents the logical equation:

$$x_{t+1} = \mu x_t(1 - x_t) \quad (9)$$

The control parameter, denoted as μ, and the variable *x* are both present in the given context. The search algorithm employed in the suggested method utilizes chaos theory principles. In this approach, the population is divided into two distinct sub-populations in the following manner:

The population in PSO is updated by utilizing the fundamental algorithm (Algorithm 1) to modify the position and velocity of the particles [36]. The population exhibits a state of disorder, as indicated by the utilization of Eq. (9) to update the positions of particles. The dynamic alteration of the particle quantities in both the PSO and chaos populations is determined by employing Eq. (10) and (11).

$$NewPSOpopulation = \frac{finessofcurrentglobalbest}{finessofpreviousglobalbest} * populationsize \quad (10)$$

$$NewChaoticpopulation = Populationsize - NewPSOpopulation \quad (11)$$

Algorithm 2 presents the pseudo-code of the suggested method, which incorporates the particle swarm optimization algorithm with chaos search.

ALGORITHM 2: PSEUDO-CODE FOR OPTIMIZATION OF PARTICLE SWARM AND CHAOS PROBE

```
Input: Composition request as a workflow (DAG) and Reil constraints
Output: Near optimal composite cloud service
Initialization as Basic PSO
While (number of iterations, or the stopping criterion is not met)
  For each particle in Chaotic Population
    Update Particle Position using equation (6)
    Decrease Chaotic search space
Xmax - Xmin
     where Xmax is the maximum position for PSO
     where Xmin is the minimum position for PSO
  End For
  For each particle In PSO Population
    Update particle velocity equation (4)
    Update Particle Position equation (5)
  END For
  For each particle In Population
    Evaluate fitness of particle
    If (current position < best position) Then
      Xpbest = current   position
    End If
    If (current position< gbest position) Then
      gbest = current particle index
    End If
  End For
  Update PSO and Chaotic Populations using (7) and (8)
End While
```

A. Initialization

During the initialization step, it is necessary to generate the initial population of particles. The particle optimization approach utilizes particles to symbolize solutions to the problem at hand. In this context, a solution refers to the compound cloud service, which is represented by an array of size n, where n corresponds to the number of jobs inside the workflow. In order to fulfill the desired objective or meet the specified criteria [37]. The value contained at index *i* within the array represents the identification number of the candidate service responsible for executing task T_i. Given that the quantity of particles in the first population is denoted as *P*, the initial population of solutions can be represented as a matrix of dimensions P×n.

B. Merit Function

The primary objectives associated with addressing the challenge of integrating cloud services while considering service reliability are adhering to user-defined constraints and maximizing a merit function. The optimization of service dependability characteristics for the composite cloud service should be the primary objective of the fitness function [38]. The proposed reliability model has six parameters: reaction time (Resp), availability (Avail), throughput (Through), success capability (Succ), reliability (Reli), and delay (Late). The merit function for a solution is determined by relation (12).

$$Fitness(Sol) = \frac{w_1 * Sol.Avail + w_2 * Sol.Throu + w_3 * Sol.Succ + w_4 * Sol.Reli}{w_5 * Sol.Resp + w_6 * Sol.Late} \quad (12)$$

The coefficients w_1, w_2, w_3, w_4, w_5, and w_6 represent positive weights assigned by the user to indicate the relative significance of each service dependability metric.

V. DISCUSSION AND EVALUATION

The proposed combination algorithm was simulated and evaluated using the MATLAB software. All tests were conducted using a Dell computer equipped with a 2.0 GHz Core i7 processor and 4 GB of RAM. Furthermore, the QWS dataset has been employed as a source of service information pertaining to applications in distributed systems.

In light of the fact that the method employed a heuristic algorithm, an assessment has been conducted to evaluate the outcomes in relation to convergence and stability. The ensuing findings will be expounded upon in the subsequent sections. Furthermore, the outcomes of the suggested approach have been juxtaposed with those of two genetic algorithms and a rudimentary particle swarm optimization technique. Tables II and III present the parameters pertaining to the genetic algorithm and optimization of both the basic particle swarm and the suggested technique, respectively.

TABLE II. PARAMETERS OF A GENETIC ALGORITHM

Parameter	Amount
Initial population size	200
Number of generations	300
Cut operator	two points
selection operator	Roulette wheel
Cutting rate	0/8
Mutation rate	0/05

TABLE III. PARAMETERS OF THE SUGGESTED TECHNIQUE AND THE FUNDAMENTAL PARTICLE SWARM OPTIMIZATION ALGORITHM

Parameter	Amount
Number of elementary particles	200
The number of repetitions	300
Inertia weight (w)	0/5
C1	0*0 2/5rnd
C2	0*0 2/5rnd
Maximum speed	$v_{max} = \lambda * x_{max}$ $0.1 \le \lambda \le 1.0$
μ	0.6

A. Convergence Test

In order to assess convergence, the proposed method, along with genetic algorithms and basic particle swarm optimization, were applied to three distributed systems applications. These applications consisted of 5, 10, and 20 tasks, respectively. Fig. 2, 3, and 4 depict the convergence process leading to the final solution for each respective application. The aforementioned graphs depict the horizontal axis representing the sequence of algorithm iterations, while the vertical axis represents the highest measure of performance achieved in each iteration.

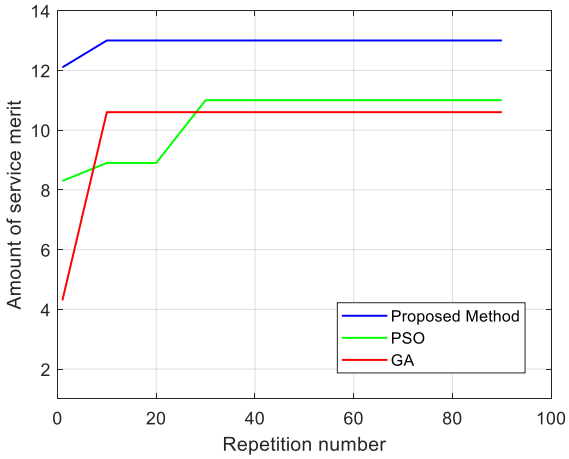

Fig. 2. Convergence of the suggested combination method compared to genetic and elementary particle optimization algorithms (number of tasks: 5).

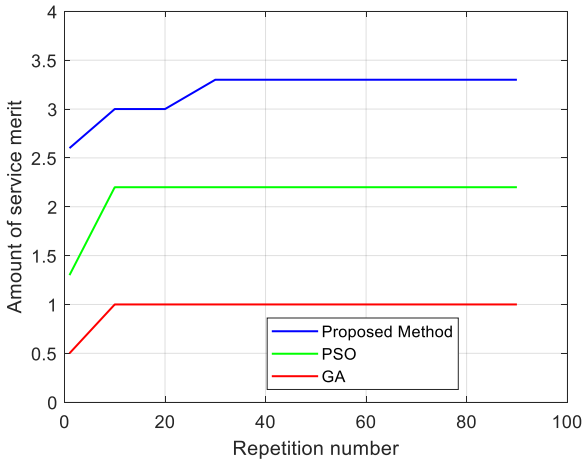

Fig. 3. Convergence of the suggested combination method compared to genetic and elementary particle optimization algorithms (number of tasks: 10).

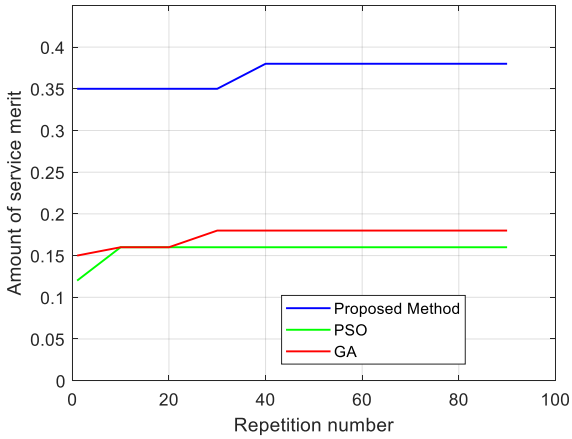

Fig. 4. Convergence of the suggested combination method compared to genetic and elementary particle optimization algorithms (number of tasks: 20).

B. Stability Test

It is imperative to conduct a thorough examination of the stability of the associated algorithm when evaluating discovery algorithms in distributed systems applications. Given the inherent stochastic character of discovery algorithms, such as the particle swarm optimization method, it is imperative to assess their stability. The concept of algorithmic stability pertains to the consistency of an algorithm's output throughout multiple executions, ensuring that the method yields identical or similar results. In order to evaluate the stability of the algorithm under consideration, the suggested methodology was implemented in four distinct distributed systems applications. These applications were subjected to 5, 10, and 20 iterations, respectively. The proposed technique was executed ten times for each application, and the resulting service merit values were recorded. These values are presented in Fig. 5, 6, and 7.

The graphs depict the order of algorithm execution on the horizontal axis, while the vertical axis represents the merit value of the composite cloud service in the applications of distributed systems for each respective order of execution.

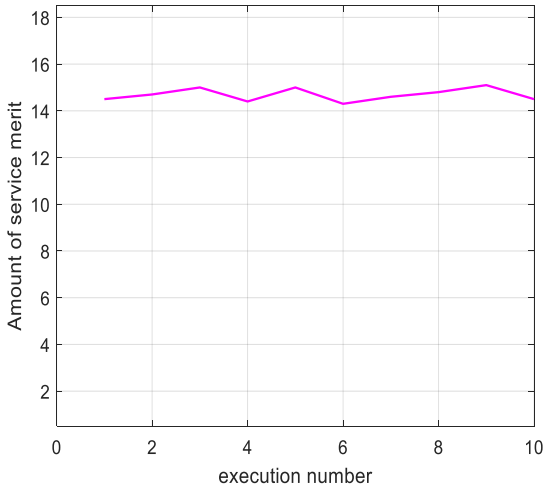

Fig. 5. The suggested combination algorithm's stability (number of tasks: 5).

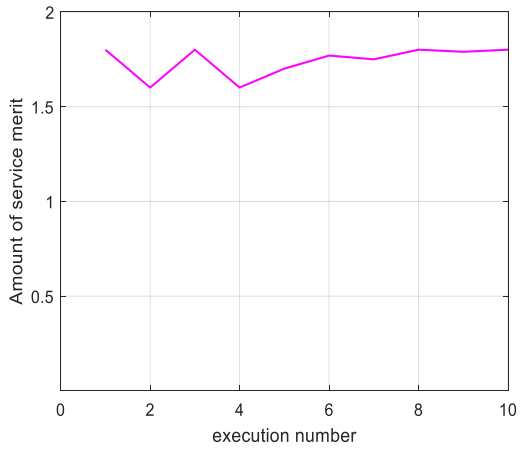

Fig. 6. The suggested combination algorithm's stability (number of tasks: 10).

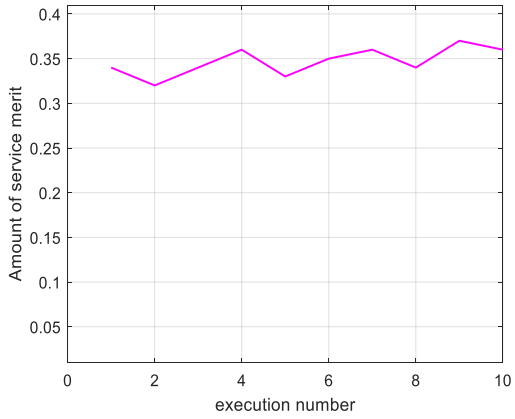

Fig. 7. The suggested combination algorithm's stability (number of tasks: 20).

Upon analysis of the stability graphs, it is evident that the suggested algorithm exhibits a notable degree of stability across various distributed systems applications. The primary factor contributing to the observed high stability is the consistent absence of fluctuations in the near-optimal service merit value over multiple algorithm iterations. The findings indicate that the algorithm's stability is significantly greater for combination requests involving a small number of tasks compared to those involving a large number of tasks.

C. Evaluating the Generated Cloud Composite Services for Quality

The primary objective of this experiment is to evaluate and compare the efficacy of the suggested method with the fundamental and genetic particle swarm optimization methods in developing distributed systems applications. In order to conduct the experiment, the aforementioned methodologies were employed to analyze 20 distinct application requests, each consisting of 5, 10, or 20 jobs. The resulting average merit values derived from the execution of these methodologies are presented in Table IV. Upon careful examination of the findings shown in Table IV, it becomes evident that the service quality of the applications generated using the suggested method surpasses that of the fundamental particles and genetics optimization algorithms, as indicated by the service quality criteria.

TABLE IV. APPLICATIONS' SERVICE QUALITY COMPARISON

Method	n = 20	n = 10	n = 5
GA [16]	0/289	1.675	10.985
PSO [19]	0.378	2.985	13.152
Proposed Method	0.426	4.898	15.685

1) Test of service quality criteria: The purpose of this study was to assess the efficacy of the developed cloud composite service based on the dimensions of accessibility, dependability, and success. To conduct the test, the user's service composition request was distributed across all three techniques, and the resultant cloud composite service was evaluated in terms of accessibility, reliability, and success. Fig. 8, 9, and 10 depict the values of the quality criteria for accessibility, reliability, and success, respectively. The findings obtained indicate that the suggested method yields a cloud composite service of superior quality compared to the particle optimization algorithm [19] and the genetics algorithm [16], as assessed by the aforementioned quality standards.

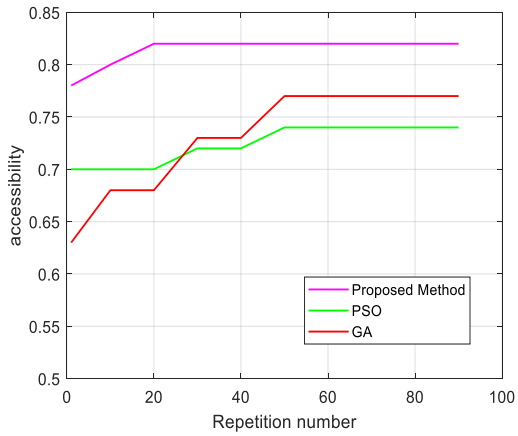

Fig. 8. The application's service accessibility.

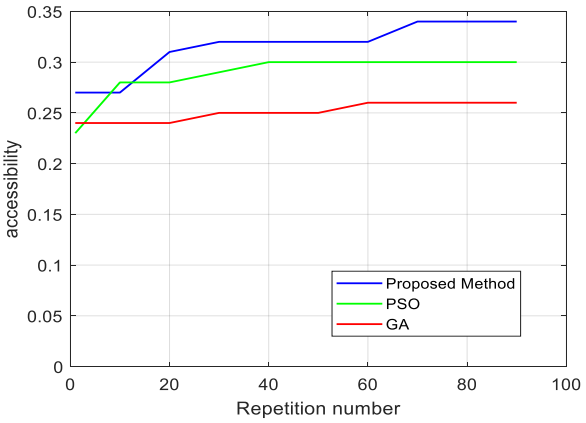

Fig. 9. The application's service reliability.

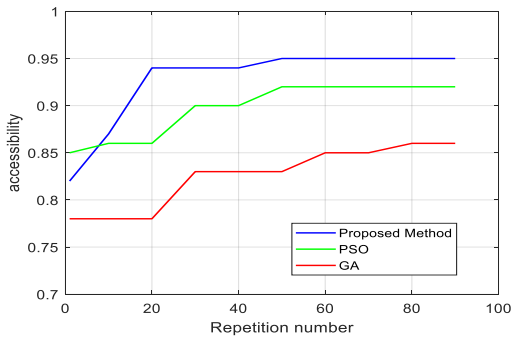

Fig. 10. The application's capacity for service success.

D. Package Delivery Rates

The quantity of packets received by application services serves as one parameter for assessing the effectiveness of algorithms in distributed system applications. A higher number of packets received by the services indicates that the algorithm performed better and supplied more data to the distribution system. We simulate the number of different gateways and then count the number of packets received by the services to get the total number of received packets. The quantity of packets that the distribution system application has received is displayed in Fig. 11.

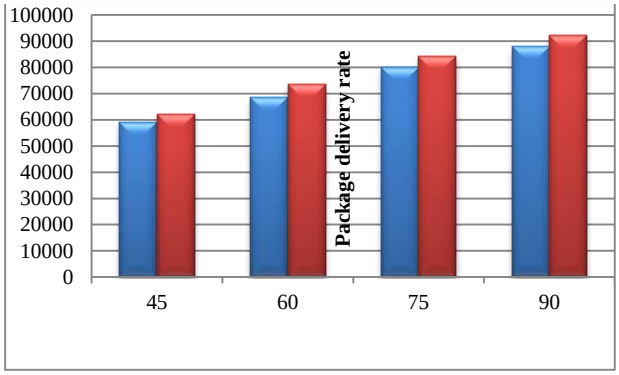

Fig. 11. The number of received packages.

The diagram presented below illustrates the packet delivery rate of the method across various services, as indicated by the observed trend (see Fig. 12).

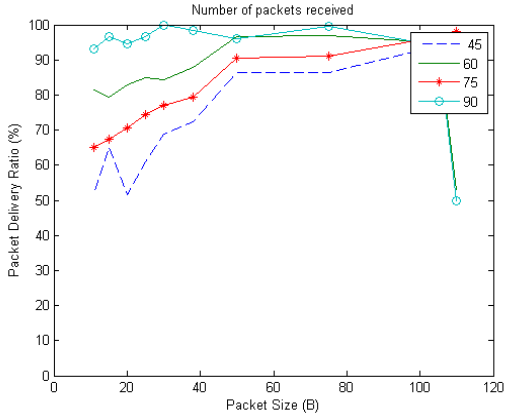

Fig. 12. Packet delivery rate.

E. Energy Consumption

The subsequent significant metric assessed to gauge the efficacy of the algorithm is the level of energy usage. The energy consumption referred to in this section pertains to the aggregate energy consumed by all services and applications throughout distributed systems and gateways. Specifically, there are 60 gateways and an unspecified number of services within these systems. The distribution value is set at 500 and the simulation is executed. The energy consumption of the two algorithms is quantified in Joules, as depicted in Fig. 13.

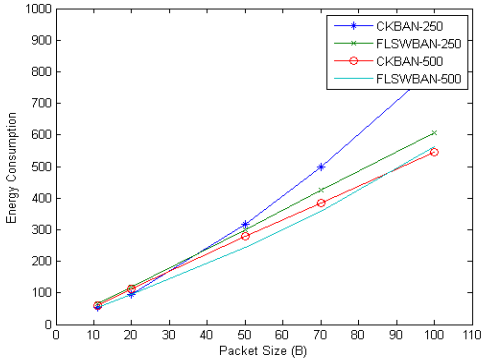

Fig. 13. Energy usage in all applications involving distributed systems.

The energy consumption in the proposed technique is comparatively lower than that of the method described in reference [17]. This can be attributed to the more efficient selection of programs utilizing chaos-based particle swarming for gateway services. When the service selection is optimized, it implies that program services are not required to transmit their data over vast distances in order to achieve their objectives, hence resulting in reduced energy consumption.

F. Reliability

As depicted in Fig. 14, the determination of reliability necessitates the presence of a timer that computes the temporal aspects associated with diverse activities, including transmission and reception. The degree of dependability associated with the initial stage, specifically when the service is introduced into the program, is disregarded. The longevity of the distributed system is of significant concern, spanning several days or even weeks. Consequently, the relatively brief duration of the first phase may be disregarded.

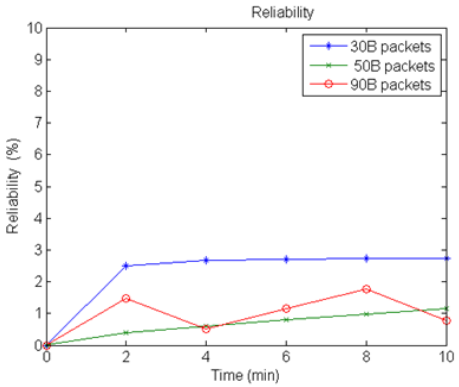

Fig. 14. Comparison of reliability over time.

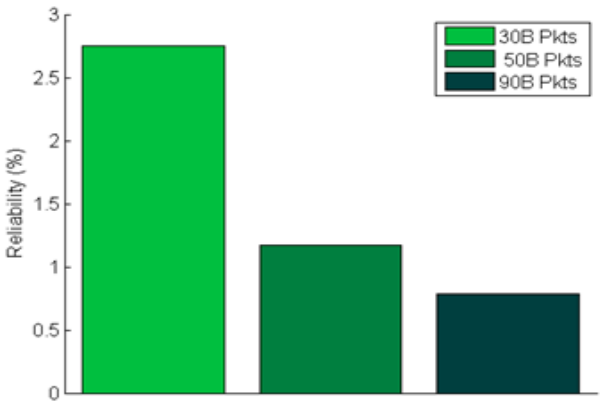

Fig. 15. Comparison of dependability in relation to received packet size.

Fig. 15 presents a crucial comparison pertaining to reliability, which is evaluated by quantifying the number of received packets. It is evident that the proposed method exhibits a high level of reliability across various quantities of received packets.

The findings of the research carried out in this article show that the use of particle swarm optimization (PSO) algorithm along with chaos theory has been able to reduce the time required to adjust the parameters of distributed systems and achieve a near-optimal solution. These findings are consistent with the results of previous researches that have investigated reliability models in large commercial projects. For example, research conducted by Al-Masri et al. has shown that the use of reliable datasets can bring significant improvements in the reliability assessment of cloud services.

However, the method proposed in this paper also has innovations that have not been found in previous research. In particular, combining the PSO algorithm with chaos theory to improve the convergence and stability of the algorithm is a new and innovative approach. This combination helps to reduce the problem of entrapment in local optima and increases the diversity of the particle population, which has not been investigated in this way in previous research. Therefore, it can be said that this research is consistent with previous findings and provides innovations that have not been investigated before.

VI. Conclusion

This study introduces a framework that utilizes the particle swarm optimization technique to optimize dependability parameters in distributed systems applications. The objective of the proposed methodology was to identify a suitable application that offers optimal service and demonstrates a high degree of expertise. Furthermore, the algorithm under consideration was executed using various settings, and the resulting graphs were subsequently analyzed. The utilization of the Particle Swarm Optimization (PSO) algorithm has been seen to decrease the time needed for determining the parameters of distributed systems applications to a certain degree. However, it should be noted that the PSO algorithm does not provide an absolute solution, but rather yields an approximation that is in close proximity to the optimal solution. In order to assess the efficacy of any algorithm, it is important to do a comparative analysis with respect to prior algorithms. Consequently, the outcomes of the study were juxtaposed with the genetic algorithm. After conducting a comparative analysis of the algorithms, it was determined that the PSO algorithm exhibits a shorter response time in comparison to both the genetic algorithm and PSO. Additionally, the PSO algorithm has favorable stability, resulting in the suggested technique yielding a solution that closely approximates the optimal solution.

Based on the analysis of the convergence graphs, it is evident that the Particle Swarm Optimization (PSO) algorithm has a favorable convergence pattern when coupled with the principles of chaos theory. Based on the stability level depicted in the graphs, it is observed that the PSO algorithm employing the chaos theory approach consistently produces a singular solution across multiple tasks. This indicates that the stability of the PSO algorithm with the chaos theory approach is commendable. Consequently, it can be inferred that the solution derived from the proposed algorithm has the potential to be the optimal solution. Based on the findings and materials elucidated in this study, a recommendation for future endeavors is conducting a comparative analysis of these algorithms through the application of chaos theory to alternative evolutionary algorithms, including genetics and colonial competition, as well as ant colony algorithms, among others.

Although this research has addressed the optimization of reliability parameters in distributed systems using the Particle Swarm Optimization (PSO) algorithm, it also has some limitations. One of the most important limitations is that the PSO algorithm only reaches a close approximation to the best solution and not a definite and absolute solution. This can be problematic in precision-sensitive applications. In addition, this research has only been compared with the genetic algorithm and has not used other evolutionary algorithms such as the colonial competition algorithm or the ant colony algorithm. Also, the presented method is implemented in the MATLAB environment, which may have limitations in generalizing the results to other platforms and execution environments.

For further studies, it can be suggested that a more comprehensive comparison be made with other evolutionary algorithms to determine the strengths and weaknesses of each one more precisely. Also, reviewing and implementing the algorithm in different environments and analyzing the results can help to increase the generalizability and applicability of the results. The use of more and more diverse real data can also lead to a more accurate evaluation of the algorithm's efficiency.

References

[1] J.-W. Wang, H.-N. Wu, Y. Yu, and C.-Y. Sun, “Mixed H2/H∞ fuzzy proportional-spatial integral control design for a class of nonlinear distributed parameter systems,” Fuzzy Sets Syst, vol. 306, pp. 26–47, 2017.

[2] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, “Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing,” Evol Intell, vol. 14, pp. 1997–2025, 2021.

[3] B. Huang, C. Li, and F. Tao, “A chaos control optimal algorithm for QoS-based service composition selection in cloud manufacturing system,” Enterp Inf Syst, vol. 8, no. 4, pp. 445–463, 2014.

[4] A. G. Gad, "Particle swarm optimization algorithm and its applications: a systematic review," Archives of computational methods in engineering, vol. 29, no. 5, pp. 2531–2561, 2022.

[5] J. Huang, Y. Liu, and Q. Duan, "Service provisioning in virtualization-based cloud computing: Modeling and optimization," in 2012 IEEE Global Communications Conference (GLOBECOM), IEEE, 2012, pp. 1710–1715.

[6] S. Gharehpasha and M. Masdari, "A discrete chaotic multi-objective SCA-ALO optimization algorithm for an optimal virtual machine placement in cloud data center," J Ambient Intell Humaniz Comput, vol. 12, pp. 9323–9339, 2021.

[7] K. Sellami, P. F. Tiako, L. Sellami, and R. Kassa, "Energy efficient workflow scheduling of cloud services using chaotic particle swarm optimization," in 2020 IEEE Green Technologies Conference (GreenTech), IEEE, 2020, pp. 74–79.

[8] H. Zhang and R. Jia, "Application of Chaotic Cat Swarm Optimization in Cloud Computing Multi objective Task Scheduling," IEEE Access, 2023.

[9] F. Tao, Y. LaiLi, L. Xu, and L. Zhang, "FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system," IEEE Trans Industr Inform, vol. 9, no. 4, pp. 2023–2033, 2012.

[10] Fansen Wei, Liang Zhang, Ben Niu, Guangdegn Zong. Adaptive decentralized fixed-time neural control for constrained strong interconnected nonlinear systems with input quantization. International Journal of Robust and Nonlinear Control, 2024, https://doi.org/10.1002/rnc.7497.

[11] S. Wang, Q. Sun, H. Zou, and F. Yang, "Particle swarm optimization with skyline operator for fast cloud-based web service composition," Mobile Networks and Applications, vol. 18, pp. 116–121, 2013.

[12] H. Cui, Y. Li, X. Liu, N. Ansari, and Y. Liu, "Cloud service reliability modelling and optimal task scheduling," Iet Communications, vol. 11, no. 2, pp. 161–167, 2017.

[13] H. Ben Alla, S. Ben Alla, A. Ezzati, and A. Mouhsen, "A novel architecture with dynamic queues based on fuzzy logic and particle swarm optimization algorithm for task scheduling in cloud computing," in Advances in Ubiquitous Networking 2: Proceedings of the UNet'16 2, Springer, 2017, pp. 205–217.

[14] K. Mishra, R. Pradhan, and S. K. Majhi, "Quantum-inspired binary chaotic salp swarm algorithm (QBCSSA)-based dynamic task scheduling for multiprocessor cloud computing systems," J Supercomput, vol. 77, pp. 10377–10423, 2021.

[15] Nasiri, E., & Wang, L. (2024). Hybrid force motion control with estimated surface normal for manufacturing applications. arXiv preprint arXiv:2404.04419.

[16] Asghari, A., Zoraghchian, A. A., & Trik, M. (2014). Presentation of an algorithm configuration for network-on-chip architecture with reconfiguration ability. International Journal of Electronics Communication and Computer Engineering (IJECCE), 5(5), 124-136.

[17] Hosseini, A., Azar, P. A., & Yang-Seon, K. (2021). An Investigation on the Impact of Retrofitting the Envelope of a Typical Small Office Building with PCM on the Building Energy Consumption in Different Zones of the US. ASHRAE Transactions, 127(1).

[18] Nasiri, E., & Wang, L. (2024). Admittance Control for Adaptive Remote Center of Motion in Robotic Laparoscopic Surgery. arXiv preprint arXiv:2404.04416.

[19] Trik, M., Pour Mozafari, S., & Bidgoli, A. M. (2021). An adaptive routing strategy to reduce energy consumption in network on chip. Journal of Advances in Computer Research, 12(3), 13-26..

[20] M. Trik, A. M. N. G. Molk, F. Ghasemi, and P. Pouryeganeh, "A hybrid selection strategy based on traffic analysis for improving performance in networks on chip," J Sens, vol. 2022, 2022.

[21] Zhang, L., Hu, S., Trik, M., Liang, S., & Li, D. (2024). M2M communication performance for a noisy channel based on latency-aware source-based LTE network measurements. Alexandria Engineering Journal, 99, 47-63.

[22] Liao, Y., Tang, Z., Gao, K., & Trik, M. (2024). Optimization of resources in intelligent electronic health systems based on Internet of Things to predict heart diseases via artificial neural network. Heliyon.

[23] Li, Y., Wang, H., & Trik, M. (2024). Design and simulation of a new current mirror circuit with low power consumption and high performance and output impedance. Analog Integrated Circuits and Signal Processing, 119(1), 29-41.

[24] Mokhlesi Ghanevati, D., Khorami, E., Boukani, B., & Trik, M. (2020). Improve replica placement in content distribution networks with hybrid technique. Journal of Advances in Computer Research, 11(1), 87-99.

[25] Hedayati, S., Maleki, N., Olsson, T., Ahlgren, F., Seyednezhad, M., & Berahmand, K. (2023). MapReduce scheduling algorithms in Hadoop: a systematic study. Journal of Cloud Computing, 12(1), 143.

[26] Z. Wang, Z. Jin, Z. Yang, W. Zhao, and M. Trik, "Increasing efficiency for routing in Internet of Things using Binary Gray Wolf Optimization and fuzzy logic," Journal of King Saud University-Computer and Information Sciences, vol. 35, no. 9, p. 101732, 2023.

[27] M. Samiei, A. Hassani, S. Sarspy, I. E. Komari, M. Trik, and F. Hassanpour, "Classification of skin cancer stages using a AHP fuzzy technique within the context of big data healthcare," J Cancer Res Clin Oncol, pp. 1–15, 2023.

[28] J. Sun, Y. Zhang, and M. Trik, "PBPHS: a profile-based predictive handover strategy for 5G networks," Cybern Syst, pp. 1–22, 2022.

[29] Minggang Liu and Ning Xu. Adaptive Neural Predefined-Time Hierarchical Sliding Mode Control of Switched Under-Actuated Nonlinear Systems Subject to Bouc-Wen Hysteresis, International Journal of Systems Science, https://doi.org/10.1080/00207721.2024.2344059, 2024.

[30] Wang, G., Wu, J., & Trik, M. (2023). A novel approach to reduce video traffic based on understanding user demand and D2D communication in 5G networks. IETE Journal of Research, 1-17.

[31] Xiangjun Wu, Ning Zhao, Shuo Ding, Huanqing Wang, and Xudong Zhao. Distributed Event-Triggered Output-Feedback Time-Varying Formation Fault-Tolerant Control for Nonlinear Multi-Agent Systems. IEEE Transactions on Automation Science and Engineering, 2024, DOI: 10.1109/TASE.2024.3400325.

[32] M. Trik, H. Akhavan, A. M. Bidgoli, A. M. N. G. Molk, H. Vashani, and S. P. Mozaffari, "A new adaptive selection strategy for reducing latency in networks on chip," Integration, vol. 89, pp. 9–24, 2023.

[33] W. Qi, "Optimization of cloud computing task execution time and user QoS utility by improved particle swarm optimization," Microprocess Microsyst, vol. 80, p. 103529, 2021.

[34] Chen Cao, Jianhua Wang, Devin Kwok, Zilong Zhang, Feifei Cui, Da Zhao, Mulin Jun Li, Quan Zou. webTWAS: a resource for disease candidate susceptibility genes identified by transcriptome-wide association study. Nucleic Acids Research.2022, 50(D1): D1123-D1130.

[35] Trik, M., Pour Mozaffari, S., & Bidgoli, A. M. (2021). Providing an Adaptive Routing along with a Hybrid Selection Strategy to Increase Efficiency in NoC-Based Neuromorphic Systems. Computational Intelligence and Neuroscience, 2021(1), 8338903.

[36] Fakhri, P. S., Asghari, O., Sarspy, S., Marand, M. B., Moshaver, P., & Trik, M. (2023). A fuzzy decision-making system for video tracking with multiple objects in non-stationary conditions. Heliyon, 9(11).

[37] Khosravi, M., Trik, M., & Ansari, A. (2024). Diagnosis and classification of disturbances in the power distribution network by phasor measurement unit based on fuzzy intelligent system. The Journal of Engineering, 2024(1), e12322.

[38] K. Guo and Y. Lv, "Optimizing routing path selection method particle swarm optimization," Intern J Pattern Recognit Artif Intell, vol. 34, no. 12, p. 2059042, 2020.

[39] Huang, H., Wang, Y., Cai, Y., & Wang, H. (2024). A novel approach for energy consumption management in cloud centers based on adaptive fuzzy neural systems. Cluster Computing, 1-24.

[40] Pirozmand, P., Jalalinejad, H., Hosseinabadi, A. A. R., Mirkamali, S., & Li, Y. (2023). An improved particle swarm optimization algorithm for task scheduling in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 14(4), 4313-4327.