Cloud Service Composition using Firefly Optimization Algorithm and Fuzzy Logic

Wenzhi Wang, Zhanqiao Liu
School of Distance Education, Jiaozuo University, Jiaozuo, 454000, China

***Abstract*—Cloud computing involves the dynamic provision of virtualized and scalable resources over the Internet as services. Different types of services with the same functionality but different non-functionality features may be delivered in a cloud environment in response to customer requests, which may need to be combined to satisfy the customer's complex requirements. Recent research has focused on combining unique and loosely-coupled services into a preferred system. An optimized composite service consists of formerly existing single and simple services combined to provide an optimal composite service, thereby improving the quality of service (QoS). In recent years, cloud computing has driven the rapid proliferation of multi-provision cloud service compositions, in which cloud service providers can provide multiple services simultaneously. Service composition fulfils a variety of user needs in a variety of scenarios. The composite request (service request) in a multi-cloud environment requires atomic services (service candidates) located in multiple clouds. Service composition combines atomic services from multiple clouds into a single service. Since cloud services are rapidly growing and their Quality of Service (QoS) is widely varying, finding the necessary services and composing them with quality assurances is an increasingly challenging technical task. This paper presents a method that uses the firefly optimization algorithm (FOA) and fuzzy logic to balance multiple QoS factors and satisfy service composition constraints. Experimental results prove that the proposed method outperforms previous ones in terms of response time, availability, and energy consumption.**

Keywords*—Cloud computing; service composition; QoS; firefly algorithm; fuzzy logic

I. Introduction

Recent rapid growth in artificial intelligence [1, 2], machine learning [3], optical networks [4, 5], smart grids [6], cloud computing [7], 5G connectivity [8], Blockchain [9, 10], and Internet of Things (IoT) [11, 12] have resulted in an explosion of data in almost all fields of engineering and commerce. With cloud computing, large-scale applications can be deployed quickly and efficiently, affordably on a per-use basis [13]. Cloud computing relies on virtualization to share resources among customers [14]. Virtualization technology enables cloud data centres to dynamically share physical resources, allowing multiple applications to run on different platforms known as Virtual Machines (VMs) [15]. As a result of virtualization, cloud service providers can ensure the quality of service (QoS) distributed among different users while achieving maximum resource utilization and minimal power consumption [16]. Cloud environments deliver services tailored to users' requirements. Existing services are combined into composite services to provide users with value-added services. As cloud computing has proliferated, more providers are providing similar functional cloud services but with diverse nonfunctional characteristics [17]. Consequently, cloud service composition must consider QoS awareness in choosing appropriate services and combining them to meet users' expectations, known as QoS-aware cloud service composition [18]. Cloud services can potentially encapsulate many resources as new technologies develop, and combinatorial optimization problems can be transformed into QoS-aware cloud service composition problems [19].

Cloud services currently available to consumers offer similar functionality at varying QoS levels. Cloud services offer different levels of QoS for a given task, so ranking these services based on QoS makes it easier for users to choose cloud services [20]. There might be a service rated best under one QoS parameter yet rated worst under another. Various QoS parameters can be used to rank the performance of a service. Cloud service composition contexts prioritize QoS parameters differently [21]. In choosing a cloud service, all QoS parameters must be taken into account without overlooking the influence of a primary QoS factor. QoS-aware cloud service composition generally considers only one or two QoS factors and ignores balancing QoS parameters or satisfying connectivity constraints [22]. In this paper, we present a novel method based on firefly optimization (FOA) and fuzzy logic to balance multiple QoS parameters and satisfy the connectivity constraints of service composition. The rest of the paper appears as follows. The next section summarizes related works. The proposed strategy is discussed in detail in Section III. Section IV reports the simulation results. Section V concludes the paper with a discussion of future directions. Generally, this work contributes to:

- A model maturity metric is introduced in this paper in order to provide a comprehensive evaluation of the simulation model lifecycle in cloud environments.
- Based on the cooperation relationship between model services, this paper dynamically calculates the maturity score of the combined model.
- A new algorithm based on FOA and fuzzy logic is proposed in this paper for the composition and optimization of cloud model services.

II. Related Work

Kritikos and Plexousakis [23] presented an approach for composing cloud services that optimally satisfy various user requirements while simultaneously composing different cloud services. Cloud application design tools do not simultaneously support quality, deployment, security, placement, or cost

requirements. In addition, the proposed approach considers a type of design choice currently not considered in the literature. Huo, Zhuang [24] propose a novel technique for cloud service composition that formalizes service composition as a nonlinear integer programming problem by incorporating a time attenuation function. Additionally, the discrete gbest-guided artificial bee colony algorithm is presented, representing the exploration of bee hives for food in search of the optimal service composition strategy. Based on experiments, it appears that the time attenuation function can improve the quality of services by making them more consistent with their characteristics at present. In comparison with other algorithms, especially for large-scale data, this algorithm provides the best possible solution in a short amount of time.

Liu, Chu [25] use social learning optimization algorithm (SLOA) to resolve the QoS-aware cloud service composition problem. Improvements to differential evolutionary algorithms and SLOA are incorporated into micro-space and learning spaces. Performance comparison and experimental results demonstrate the efficacy of the SLOA. This work will improve search capabilities and convergence rates by extending the theory of the swarm intelligence optimization algorithm and exploring a novel swarm intelligence optimization model. The challenge of correlation-aware QoS in networks and cloud services is addressed by Huang, Li [26]. This problem is formulated as a multi-constraint optimal path problem, and a novel approach is proposed for solving it. The proposed algorithm is evaluated using extensive simulation. The proposed algorithm yields superior service composition solutions with improved QoS guarantees by taking into account the QoS correlations between the different service types.

Karimi, Isazadeh [27] employed the genetic algorithm to optimize service level agreements globally. Service clustering was applied to reduce the search space, and association rules were applied to a composite service based on their histories to improve service composition efficiency. As compared with similar related works, the proposed method demonstrated higher efficiency. Low-cost access to a simplified, centralized platform or resource is provided by cloud computing. This type of computing allocates resources based on individual needs. However, resources need to be allocated efficiently to meet the expanding needs of cloud users. Service providers are responsible for distributing and sharing resources effectively, preventing resource waste. Furthermore, the user receives the appropriate service based on their request, with the cost of the resource being optimized. Singh, Juneja [28] present an algorithm for automated service composition based on agents, which addresses both service requests and automatic service composition. The algorithm searches for the best services and reduces the cost of on-demand virtual machines.

Wang, Zhou [29] analyzed the relationship between energy consumption, network resource consumption, and QoS performance in a service composition process. An approach to green service composition is then presented. The system prioritizes composite services that run on the same physical server, virtual machine, or edge switch. Green service composition optimization minimizes energy and network resource consumption on physical servers and switches in cloud data centers. Based on experiments, the proposed approach reduces energy consumption by 20-50 percent and network resource consumption by 10-50 percent compared to other approaches. Jian, Li [30] presents an algorithm incorporating the two-order oscillating equation and the historical positions of birds. It enhances bird diversity, strengthens global search algorithms, and improves bird feeding and migration dynamics. Based on simulation results with and without local QoS restrictions, the algorithm minimizes overall QoS restriction execution times. The new eagle search procedure was developed by Jin, Lv [31] by integrating regular mutations with an improved whale optimization algorithm. In order to verify the performance of the new approach, a variety of benchmark functions and problems of cloud service composition are used. The proposed method outperforms the other methods, according to experiments.

Studies presented above suggest a variety of solutions to the problem of service composition. The majority of researchers, however, use a simple additive weighting proposal in order to combine multiple aspects of QoS into a single-objective function. Service composition is primarily a multi-objective problem, in which multiple, often conflicting attributes of QoS must be optimized simultaneously. A number of proposals have been considered to address the problem of service composition from a multi-objective perspective, including a tri-objective service composition [32], a bi-objective service composition [33], and a bi-objective service composition from an energy and global quality of service utility's perspective [14]. However, these studies only identify up to three representative objectives by assigning priorities to QoS factors or reducing a large proportion of QoS factors within two or three objectives. The literature rarely examines service composition scenarios with four or more objectives. The QoS criteria for many applications have expanded to include security, maintainability, reputation, energy consumption, carbon emissions, and ecological impact. With a range of trade-off options available, the decision-making process becomes more flexible. Service composition is strictly a multi-objective optimization problem if we treat all QoS attributes equally.

III. PROPOSED METHOD

This section will provide a new IoT-enabled cloud service composition method based on FOA and fuzzy logic systems. We aim to improve the QoS parameters, reduce energy consumption, extend the network's life, improve packet delivery rates, and decrease delays. Due to two advantages of automatic segmentation of the network and diversity in solutions, the FOA can perform better than other algorithms in finding the optimal solution to optimization problems. In the following, we have discussed the network model and problem assumptions, optimization models, objective function, and finally, the proposed method.

A. The Problem Expression and Formulation

Under one set of QoS parameters, a cloud service may be rated as the best, whereas under another set of QoS parameters, it may be rated as the worst. Various QoS parameters can be used to rank cloud service performance. Cloud services that rank highest under one QoS parameter, such as reliability, may

rank lowest under another QoS parameter, such as response time. Therefore, QoS parameters are prioritized differently depending on the composition context. A cloud service that is optimal under one QoS parameter may fail if that cloud service is not optimal under another QoS parameter. Therefore, including any of these cloud services as candidate services in the composition will have an impact on the overall performance, since these cloud services perform poorly under parameters other than the primary QoS parameter.

QoS-aware services composition problem is finding a set of candidate services with different functional features that comply with the determined limitations by the user and optimize an objective function. This problem is explained in this section.

- A service composition request as a workflow that is modeled using a Directed Acyclic Graph (DAG), G= (V, E).
- $V = \{T_1 . T_2 . T_3 . \dots . T_n\}$ That n is the number of tasks in the workflow.
- E is the set of edges showing the priority of executing the works.
- Each T_i for $0 \leq i \leq n$ is a set of candidate services in the workflow.
- $CS_i = \{CS_i^1 . CS_i^2 \dots . CS_i^m\}$: That CS_i^J for $1 \leq j \leq m$ is a candidate service.
- M_i is the number of existing candidate services for T_i.
- Each candidate service CS_i^J has a set of different QoS information.
- $QOS = \{Q_1 . Q_2 . Q_3 . \dots . Q_n\}$ That Q_i for $1 \leq i \leq k$ shows a QoS feature of cloud services.

Considering the above cases, the goal of the QoS-aware service composition problem is near optimal so that:

$$\forall j = 1 \dots k \begin{cases} \sum_{i=0}^{n} S_i . Q_i < C_j & if\ Q_j\ is\ additive \\ \prod_{i=1}^{n} S_i . Q_i > C_j & if\ \ Q_j\ is\ multiplicative \end{cases} \quad (1)$$

Composition and service selection problem in all computational platforms like cloud environment and IoT is appropriate to meet a user or system requirements that should provide QoS requirements in addition to the user's needs. A fitting function in a service composition problem, like any optimization problem, is used to determine whether the selected service is valuable. This function's output determines the efficiency of the selected services for composition.

Generally, the most important factor for the development and scalability of IoT is considering energy consumption limitations. Hence, energy saving is one of the important challenges in these networks because energy sources in IoT nodes are limited, and changing the energy sources of the nodes on large scales is practically impossible. Based on the mentioned notes, energy consumption is critical for the candidate services' host nodes. Thus, each candidate service must propose its required energy to the service provider module to select one of the most energy-efficient nodes for service composition. Two variables are used in the proposed energy model for the model definition. The first one is the amount of remaining energy and the second one is the amount of consumed energy of the node, which is the amount of consumed energy when executing the user's request. Based on the model, $RE(cs_{ij})$ is the remaining energy of the candidate service host (cs_{ij}), and $CE(cs_{ij})$ is the consumed energy when running the candidate's request (cs_{ij}). Hence, the amount of remaining energy of the host is calculated using 3:

$$RE(cs_{ij}) = CDE(cs_{ij}) - E_{th}(cs_{ij}) \quad (2)$$

where $CDE(cs_{ij})$ is the level of energy in the IoT node equipped with the battery, which is the host of cs_{ij} service, and $E_{th}(cs_{ij})$ is the minimum threshold energy of IoT so that if a node's energy is less than the threshold, the node cannot work in the network and dies.

Since the energy model in this thesis is based on service-oriented calculations, the consumed energy by the service candidate when running cs_{ij} is estimated using equation (3). We assume that the energy consumption of the candidate service cs_{ij} is constant because the services are run on the same IoT platform and use the same resources to answer the service. Moreover, they consume the same amount of energy when data transmission and reception.

$$CE(cs_{ij}) = ECR(cs_{ij}) \times T(cs_{ij}) \quad (3)$$

where $CE(cs_{ij})$ is the consumed energy, and $T(cs_{ij})$ is the run time of the candidate service (cs_{ij}). Finally, the consumed energy of the cs_{ij} candidate service is defined as an EP and calculated by Eq. (4). The profile is estimated based on the ratio of the cs_{ij} candidate service consumed energy to its remaining energy.

$$EP(cs_{ij}) = \frac{CE(cs_{ij})}{RE(cs_{ij})} \quad (4)$$

Based on Eq. (3) to (4), the lower amount of $EP(cs_{ij})$ in a node means that the node is optimal for cs_{ij} service at time T_i. Finally, the EP of service composition is computed using Eq. (5). Hence, the consumed energy of the service composition path x^i is as follows:

$$EP(x) = \sum_{i=1}^{n} EP(x^i | x^i \in s^h) \quad (5)$$

where $s^h$is the set of IoT nodes with a battery. The nodes with lower energy than the threshold energy is removed from the set.

- Response Time: the required time duration for a request transmission and its response reception.

- Availability: the number of successful calls to the total number of calls ratio.
- Successability: the number of responses to the number of request messages ratio.
- Reliability: the number of faulty messages to the total number of messages ratio.
- Latency: the duration of a request process by a server.

In the selection and composition of IoT services, the composing services can be composed and meet the users' needs. Hence, calculating the composed QoS features summation is important, and each serial, parallel, switch, and loop pattern uses its formulas. The requested works and the proposed services are considered a graph for service composition. Thus, the following equations are used for the QoS parameter values calculation in the composing services. Eq. (6) calculated the response time parameter, and Eq. (7) to (11) are used to calculate delay, energy, reliability, availability, and successability parameters in the composing services, respectively.

$$Q_{Responce\ Time}(s) = \sum_{i=1}^{n} Q_{RT}(s) \tag{6}$$

$$Q_{Latency}(s) = \sum_{i=1}^{n} Q_{L}(s) \tag{7}$$

$$Q_{Energy\ Consumption}(s) = \sum_{i=1}^{n} Q_{EC}(s) \tag{8}$$

$$Q_{Reliability}(s) = \prod_{i=1}^{n} Q_{R}(s) \tag{9}$$

$$Q_{Accessability}(s) = \prod_{i=1}^{n} Q_{A}(s) \tag{10}$$

$$Q_{Successability}(s) = \prod_{i=1}^{n} Q_{S}(s) \tag{11}$$

Generally, the fitting function (objective function) is used to determine the value of each solution generated by the optimization problems. This function output determines how much a solution can be useful for meeting the user's or the system's requirements. The fitting function is defined by Eq. (12).

$$Fitness(Sol) = W_1 * Resp + W_2 * Late + W_3 * \frac{1}{Avail} + W_4 * \frac{1}{Succ} + W_5 * \frac{1}{Reli} + W_6 * Energy \tag{12}$$

where W_1, W_2, W_3, W_4, W_5, and W_6 are the positive weights showing the importance of each QoS parameter determined by the user. Eq. (13) and (14) are used for normalizing the parameters' output in the objective function. Eq. (13) is for the minimizer parameters, and (14) is for the maximizer parameters.

$$N_{CS.Q^i}\begin{cases}\dfrac{Q^i_{max} - CS.Q^i}{Q^i_{max} - Q^i_{min}} & Q^i_{max} \neq Q^i_{min} \\ 1 & Q^i_{max} = Q^i_{min}\end{cases} \tag{13}$$

$$N_{CS.Q^i}\begin{cases}\dfrac{CS.Q^i - Q^i_{min}}{Q^i_{max} - Q^i_{min}} & Q^i_{max} \neq Q^i_{min} \\ 1 & Q^i_{max} = Q^i_{min}\end{cases} \tag{14}$$

where $CS \cdot Q^i$ is the ith parameter value of QoS related to the CS selected service, $Ncs \cdot Q^i$ is the normalized value, and Q^i_{max} and Q^i_{min} are the maximum and minimum of the ith parameter among all the services.

B. The Proposed Method's Steps

This section has two subsections. The first subsection includes general information about the firefly algorithm and Fuzzy logic, and the proposed method steps are explained in the second subsection. The FOA algorithm was inspired by how fireflies attract mates by producing flashlights. The FOA uses three idealized features of the firefly's flashing light to produce an optimal solution. Fireflies are unisex and attract other fireflies irrespective of their gender.

Further, when two fireflies are distant from each other, their attractiveness decreases, which is directly proportional to their brightness. Finally, the firefly flashing light is incorporated into the optimization process as an objective function [34, 35]. Fuzzy logic is an approach to variable processing that allows multiple possible truth values to be processed through the same variable. Fuzzy logic attempts to solve problems with an open, imprecise spectrum of data and heuristics that makes it possible to obtain an array of accurate conclusions. The fuzzy logic architecture consists of the following components:

- **Rule base:** This contains the rules and membership functions that regulate or control decision-making in the fuzzy logic system. It also contains the IF-THEN conditions for conditional programming and controlling the system.
- **Fuzzifier:** This component transforms raw inputs into fuzzy sets. The fuzzy sets proceed to the control system, where they undergo further processing.
- **Inference engine:** This tool establishes the ideal rules for a specific input. It then applies these rules to the input data to generate a fuzzy output.
- **Defuzzifier:** This component transforms the fuzzy sets into an explicit output (in the form of crisp inputs). Defuzzification is the final stage of a fuzzy logic system.

The proposed method uses the fuzzy logic system and firefly optimization algorithm for optimal service composition. Since the firefly algorithm has a low speed for local search, Fuzzy logic is used to ease finding the optimal composition in terms of the problem objectives. The firefly algorithm can find and compose more optimal services in less time by applying fuzzy values for each QoS parameter. The basis of the work is as follows: first, a firefly is assigned to each service. Then the fireflies create paths based on their movement pattern and

Luciferin value to find the optimal service composition. Finally, the best services are used for composition by evaluating the route found by the fireflies. The Fuzzy values for each service in the Luciferin function are saved for each firefly. The fireflies find the best composition by applying the Fuzzy values with the neighbour nodes' Luciferin.

1) First step: generating primary population: The primary population is generated in this step using random distribution. The IoT nodes generated based on the geographical positions are distributed in the objective area. After the distribution of nodes, the parameters' initial values, like energy, Luciferin value, and the nodes transmission's radius, are assigned to the parameters. Moreover, data related to each QoS parameter are read from the dataset and put in the parameters.

2) Second step: QoS parameters fuzzification: The proposed method's first step includes fuzzifying each IoT service QoS information. This information includes response time, availability, successability, reliability, energy, and delay. These values are the Fuzzy system inputs in this step, in which a Fuzzy value is generated based on the Fuzzy rules for all inputs to evaluate and select the services based on the value. This Fuzzy value is defined as Low, High, Medium, Very Low, and Very High. Then the system analyzes the information based on the inputs and the written rules in the knowledge base of the fuzzy system and generates the final value after defuzzification. These values then transfer to the firefly algorithm as inputs.

The fuzzy rules are written considering different states of the input parameters and finally are saved in the database. Then using a Fuzzy Inference System (FIS), an output value is generated based on different input states. The Fuzzy rules are written using nested If-Then. The proposed Fuzzy system used 25 rules shown in Table I. Fig. 1 to 5 illustrate the membership functions of the FIS input and output parameters. In the Fuzzy rules table, RT is the response time, D is the delay, En is energy, Av is availability, and Fitness is the output parameter.

After generating different fuzzy values based on the fuzzy rules and input parameters, defuzzification is performed on each fuzzy value. Membership functions of the fuzzy system output variables are presented in Fig. 3 to 11.

3) Third step: using the firefly algorithm to select the best services for the composition: Each firefly in this algorithm is an answer to the problem. All answers have fitting values calculated by the fuzzy system that should be optimized. The first step is creating a list of the proposed services to do the existing tasks in the workflow. Each workflow proposed by the user has n tasks. There is a determined number of services to perform each task, and m is the maximum number of the proposed selected services for each task. At first, an n*1 array of random numbers with size m are generated to initialize the algorithm. The array shows the number of existing services in the accumulator that can perform the workflow tasks. In the next step, the matrix is generated to save the list of the services that can perform the tasks in the workflow. The number of proposed services listed to do the workflow's tasks is generated in the next step. Each presented workflow by the user has m tasks, and there is a determined number of services to do each of the services. The optimization process started with some random solutions. The primary solution is selected among the proposed services for each task. In the firefly algorithm, the primary solution is a service composition, an n*1 array with n tasks in the workflow. The stored number in index i of the array shows the ID of the candidate service that executes the Ti task. The following steps are performed to select and compose the most proper services based on the firefly algorithm steps:

- In the first step, some primary population is generated and distributed randomly. Initialization of Luciferin is applied to each firefly, and the primary solutions are generated. These solutions are valued based on the fitting function value.
- In the second step, the considered proposed QoS parameters as inputs of the problem are fuzzified based on the proposed Fuzzy system. Their values in each round of the algorithm are inserted and updated in the Luciferin matrix as the next step fitting value (the neighbour nodes). Since the parameter values change in each round of the algorithm, the proposed fuzzy system is executed based on the Fuzzy rules in each round, and new values are generated.
- In the third step, the firefly algorithm generates new solutions based on the movement steps. A population of fireflies creates new solutions in the proposed algorithm. The generated solutions of each iteration go to the next iteration for more optimal solutions. A firefly position shows a solution for the composing service problem that the primary position of the fireflies is random. Each firefly generated new solutions in each algorithm iteration based on the fitting value using Eq. (15). The solutions are updated based on their quality. If the new solutions are better than the existing ones, they are replaced.

$$x_{\mathrm{j}}=x_{\mathrm{i}}+\beta_0.e^{-\gamma r i j^2} \quad (15)$$

After that, all fireflies generate their solutions. A probability is used for selecting the most appropriate solution for each firefly, given by Eq. (16). After calculating the selection probability of each node as the next step, one of the nodes is selected randomly.

$$p_{ij}(\mathrm{t})=\frac{L_j(t)-L_i(t)}{\sum_{k\epsilon N_i(t)} L_k(t)-L_i(t)} \quad (16)$$

In the above equation, $L_i(t)$ is the amount of luciferin of each firefly that is calculated using Eq. (17).

$$L_i(t)=(1-\rho)L_i(t-1)+\gamma J(x_i(t)) \quad (17)$$

In the Eq. (17), $L_i(t)$, $L_i(t-1)$ and $x_i(t)$ are respectively the new value of luciferin, the previous value of luciferin, and the fitness of the location of worm i in iteration t of the algorithm, respectively, and ρ and γ are fixed numbers for modeling Gradual decline and fitness effect on luciferin.

The step-by-step movement of the fireflies continues until selecting the most suitable service for composition. After selection, the generated solution evaluation is performed so that the average distance of the selected solution is calculated rather than other solutions. Then the fitting function value is computed for new solutions. If the new solution is better than the existing solution in the memory, the firefly memory is updated using the new solution. Then the parameters are updated. The proposed method flowchart is illustrated in Fig. 6.

TABLE I. PROPOSED FUZZY RULES

The fuzzy rules
If (RT is VL) and (D is VL) and (En is VL) and (Av is VH) then (Fitness is VH)
If (RT is VL) and (D is VL) and (En is L) and (Av is VH) then (Fitness is H)
If (RT is VL) and (D is VL) and (En is L) and (Av is H) then (Fitness is H)
If (RT is VL) and (D is VL) and (En is VL) and (Av is H) then (Fitness is VH)
If (RT is VL) and (D is VL) and (En is VL) and (Av is M) then (Fitness is H)
If (RT is VL) and (D is VL) and (En is VL) and (Av is L) then (Fitness is L)
If (RT is VL) and (D is VL) and (En is VL) and (Av is VL) then (Fitness is VL)
If (RT is L) and (D is L) and (En is L) and (Av is VH) then (Fitness is VH)
If (RT is L) and (D is L) and (En is L) and (Av is H) then (Fitness is H)
If (RT is L) and (D is L) and (En is L) and (Av is M) then (Fitness is H)
If (RT is L) and (D is L) and (En is L) and (Av is L) then (Fitness is L)
If (RT is L) and (D is L) and (En is L) and (Av is VL) then (Fitness is VL)
If (RT is M) and (D is M) and (En is M) and (Av is VH) then (Fitness is M)
If (RT is M) and (D is M) and (En is M) and (Av is H) then (Fitness is M)
If (RT is M) and (D is M) and (En is M) and (Av is M) then (Fitness is M)
If (RT is M) and (D is M) and (En is M) and (Av is L) then (Fitness is L)
If (RT is M) and (D is M) and (En is M) and (Av is VL) then (Fitness is VL)
If (RT is H) and (D is H) and (En is H) and (Av is VH) then (Fitness is M)
If (RT is H) and (D is H) and (En is H) and (Av is H) then (Fitness is H)
If (RT is H) and (D is H) and (En is H) and (Av is M) then (Fitness is L)
If (RT is H) and (D is H) and (En is H) and (Av is L) then (Fitness is VL)
If (RT is H) and (D is H) and (En is H) and (Av is VL) then (Fitness is VL)
If (RT is VH) and (D is VH) and (En is VH) and (Av is VH) then (Fitness is VL)
If (RT is VH) and (D is VH) and (En is VH) and (Av is H) then (Fitness is VL)
If (RT is VH) and (D is VH) and (En is VH) and (Av is M) then (Fitness is L)
If (RT is VH) and (D is VH) and (En is VH) and (Av is L) then (Fitness is M)
If (RT is VH) and (D is VH) and (En is VH) and (Av is VL) then (Fitness is VL)

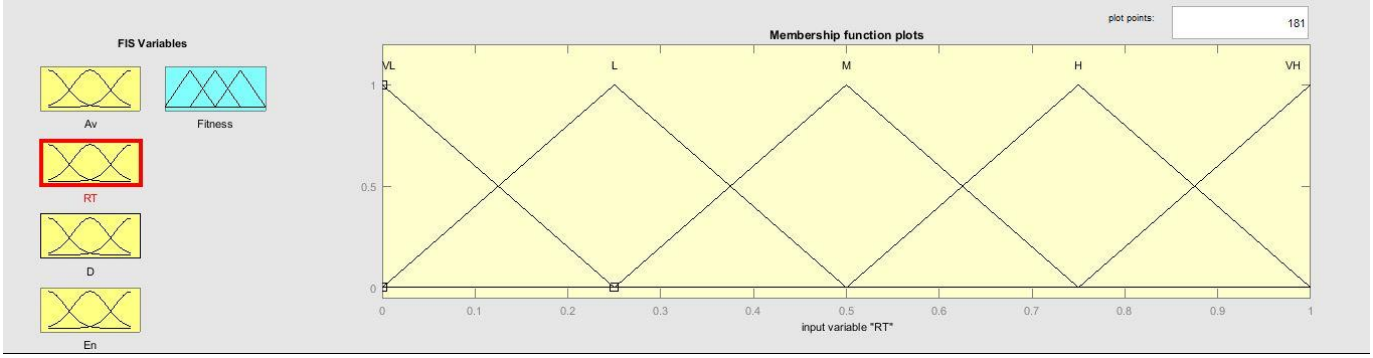

Fig. 1. Membership function for response time parameter.

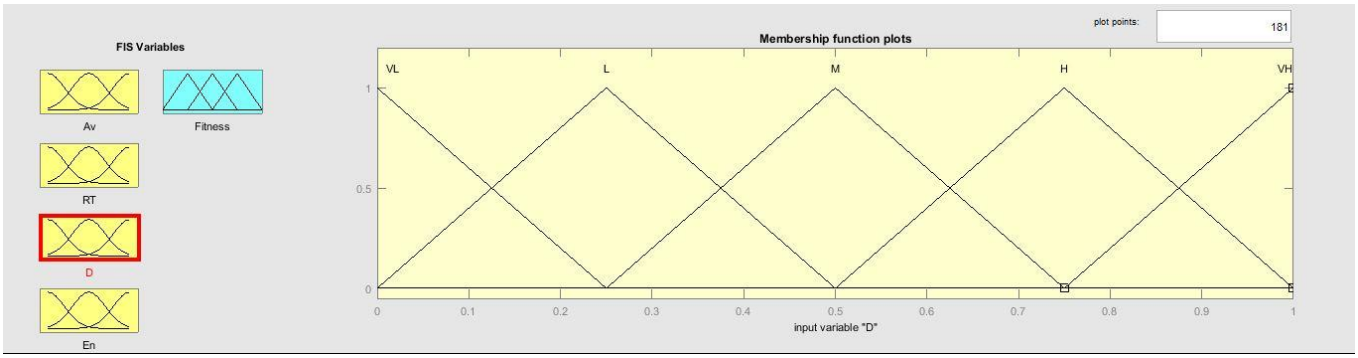

Fig. 2. Membership function for delay parameter.

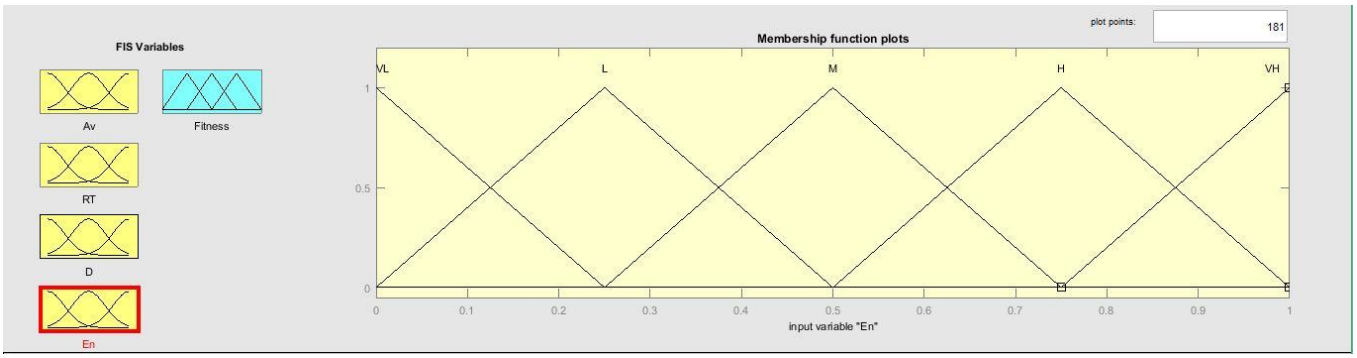

Fig. 3. Membership function for energy parameter.

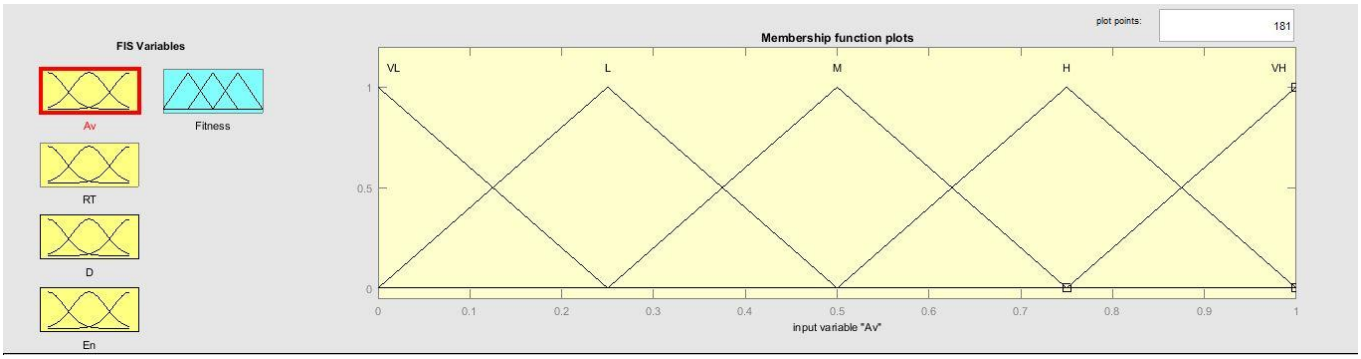

Fig. 4. Membership function for accessibility rate parameter.

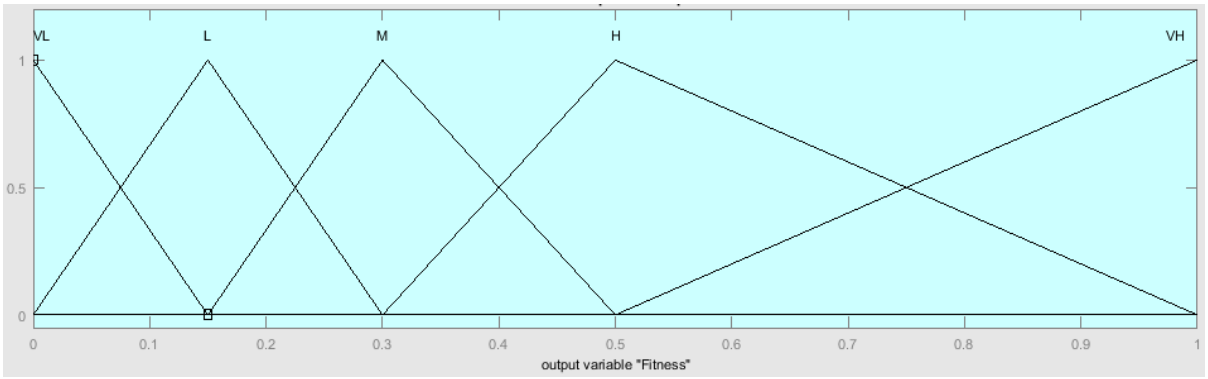

Fig. 5. Membership function for fuzzy system output.

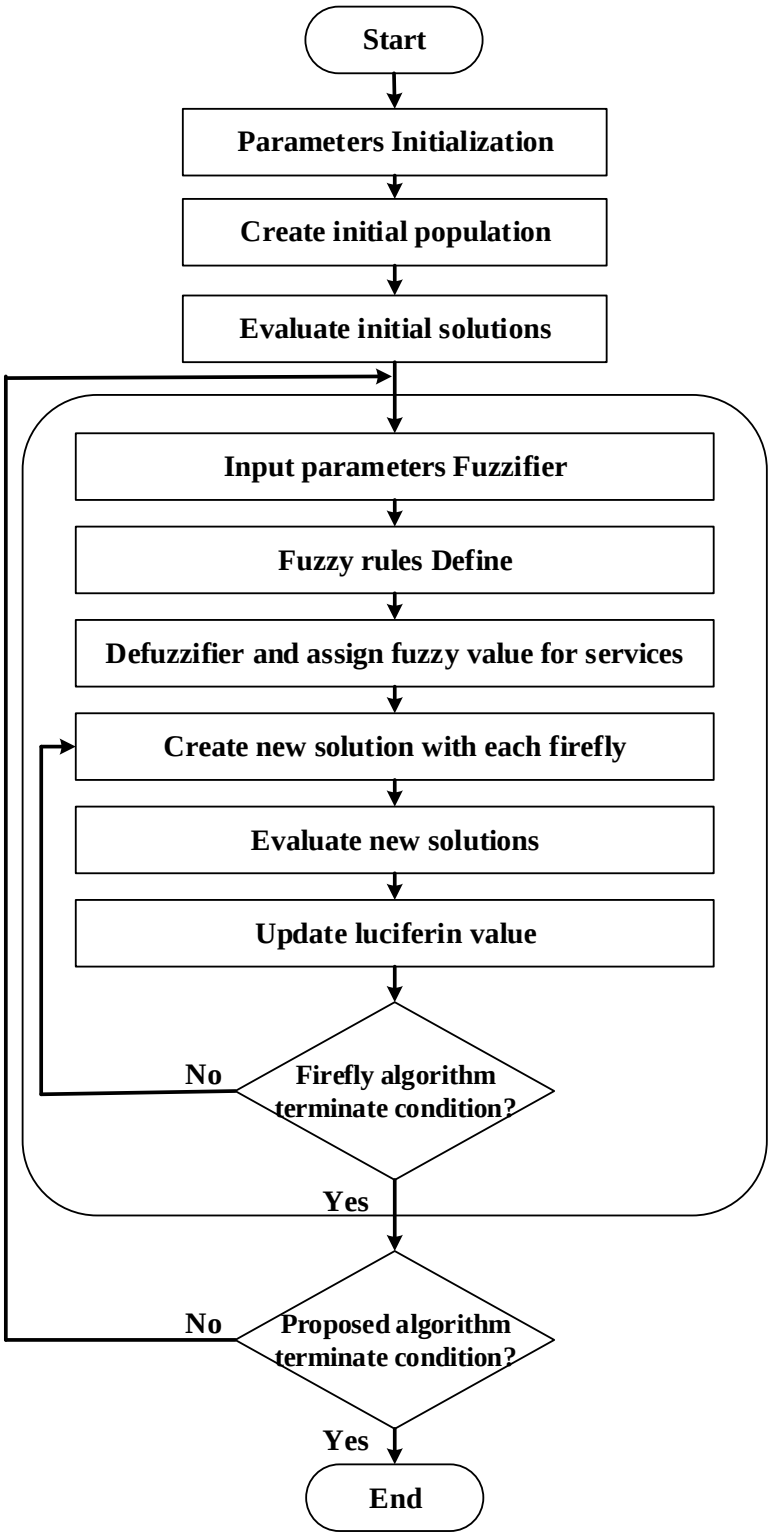

Fig. 6. Proposed method flowchart.

IV. Evaluation and Simulation

Matlab is used for the simulation of the proposed method. All these experiments are performed on an HP computer with a Core i5 2.0 GHz CPU and 4 GB main memory. The utilized dataset in the proposed method is a dataset provided by Al-Masri et al., called Quality of Web Service (QWS), and provides a base for the services researchers. The QWS dataset includes a set of 2507 web services and measures their QoS. These data are presented in Table II. The parameters are considered objective parameters. Firefly algorithm parameters also are presented in Table III.

A. The Simulation Results' Study

The proposed method's results are compared with [36], [37], and [38] in this section and presented in different tables and figures. Moreover, two experiments are explained in the following to study the proposed method's convergence to optimal solutions and stability test.

In the first experiment, the resulting data of [36] are used to evaluate the proposed method in this thesis. The proposed method in [36] is based on a multi-objective genetic algorithm to investigate the service composition of IoT that is evaluated with the different number of tasks and candidate services for the tasks. Hence, our proposed method is evaluated in this experiment with ten tasks and 10, 30, and 50 candidate services. Fig. 7 shows the proposed method's execution time and comparison with the multi-objective genetic algorithm [36]. Based on the results, the proposed method's execution time is less than the genetic algorithm to find appropriate services for composition. Because the proposed method can find the most suitable network node for the services selection and composition that provides QoS requirements by tracing the most optimal IoT nodes. The routing method and selecting the most suitable resource (IoT node) with proper services for the user's requests will be explained in detail with mathematical equations in the third chapter.

In the second experiment, the resulting data of [37] are used to evaluate the proposed method in this thesis. The proposed method in [37] is based on the Markov chain and ant colony optimization algorithm for IoT services composition. The authors of [37] evaluate their method with the different number of requests. Hence, our proposed method (firefly-fuzzy logic) is simulated and evaluated with the different number of requests in this experiment.

Fig. 8 to 11 show the availability rate, response time, reliability rate, and consumed energy of the proposed method, respectively, which are compared with the method in [37]. Based on the results in Fig. 8, the proposed method (firefly-fuzzy logic) has a higher accessibility rate than the Markov-ant colony because of tracing the network nodes based on energy and distance. Moreover, selecting the most suitable route between the network's nodes (network's resources) is based on the movement pattern of the fuzzy firefly. According to Fig. 9, the response time of the proposed method is less than the method in [37]. Fig. 10 to 11 also show better performance of the proposed method than the method in [37] in terms of reliability.

In the third experiment, the resulting data of [39] are used to evaluate the proposed method in this thesis. The proposed method in [39] is based on the Concurrent Requests Integration Optimization (CRIO) mechanism and Gray Wolf (GWO) for IoT services composition. Fig. 12 shows the consumed energy by the proposed method. The results in Fig. 12 show that the proposed method has a lower energy consumption than the CRIO-GWO.

TABLE II. QWS PARAMETERS DEFINITION

Parameters	Definition
Response Time	The required time duration for a request transmission and its response reception.
Availability	The number of successful calls to the total number of calls ratio.
Energy	The amount of the consumed energy of the host nodes of the candidate services.
Successability	The number of responses to the number of request messages ratio.
Reliability	The number of faulty messages to the total number of messages ratio.
Latency	The duration of a request process by a server

TABLE III. SIMULATION VARIABLES

Parameters	Definition	Value	measurement Unit
N	Number of nodes	100-600	-
MaxIT	Maximum algorithm iteration	100-1000	-
L	Data packet length	1024	bit
$E_{initial}$	Initial energy	0.5 – 10	joule
α_0	Firefly algorithm constant	[0 – 1]	-
β_0	Firefly algorithm constant	1	-
γ	Firefly algorithm constant	0.1 – 10	-

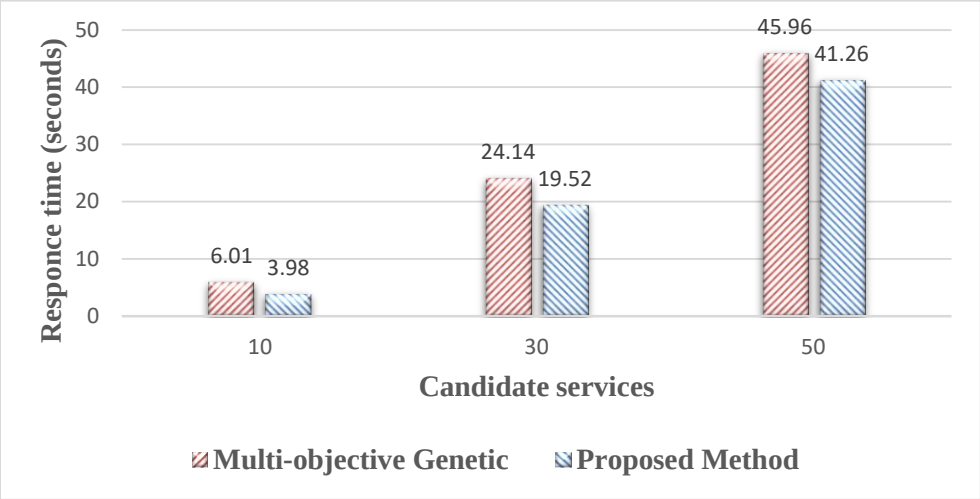

Fig. 7. Response time vs. the number of candidate services.

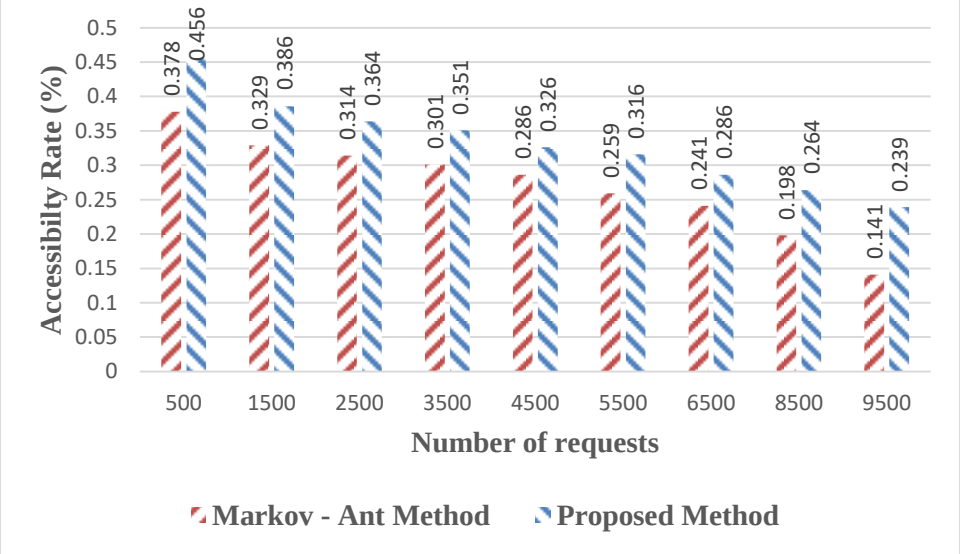

Fig. 8. Accessibility rate vs. the number of requests.

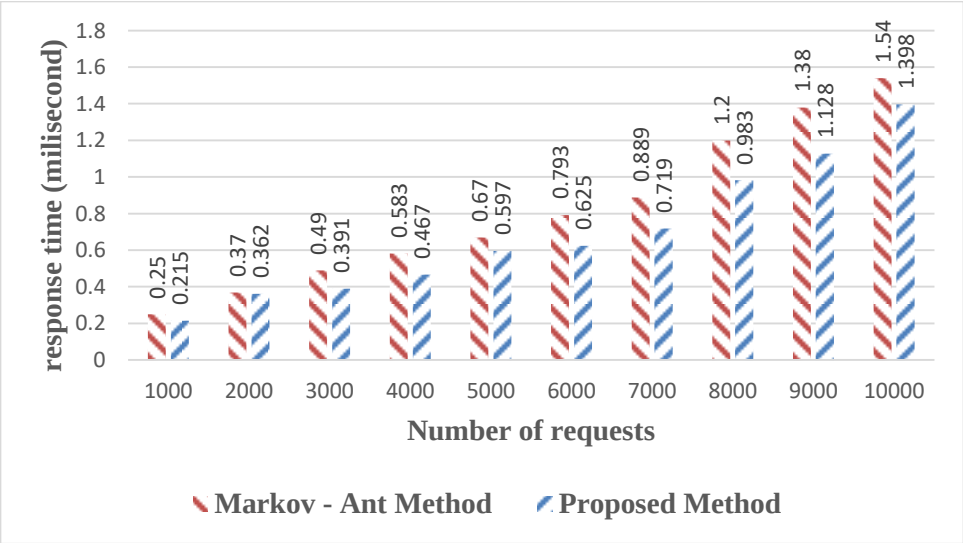

Fig. 9. Response time vs. the number of requests.

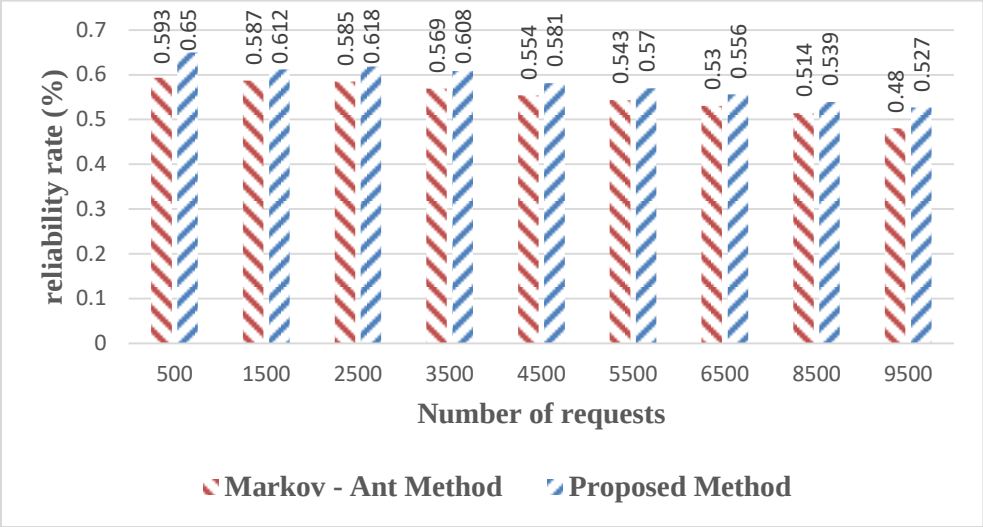

Fig. 10. Reliability rate vs. the number of requests.

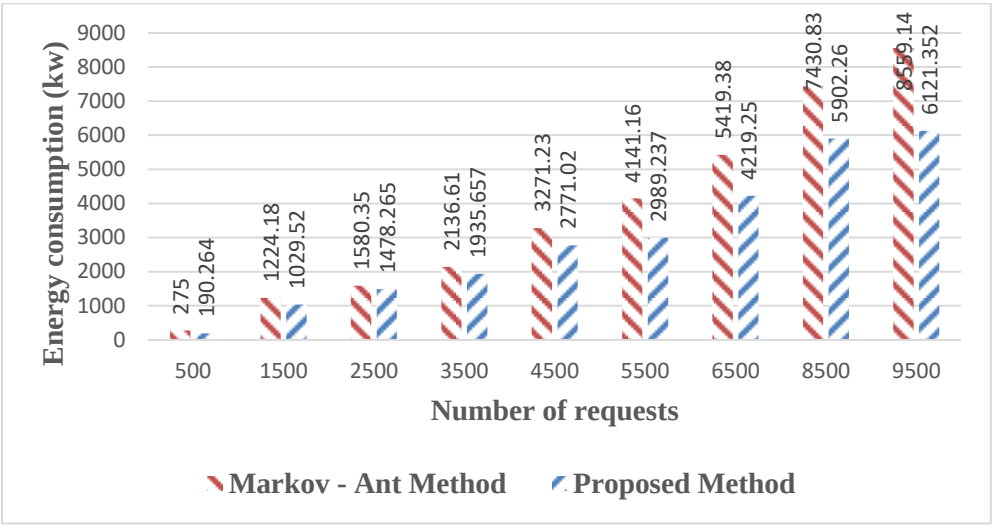

Fig. 11. Energy consumption vs. the number of requests.

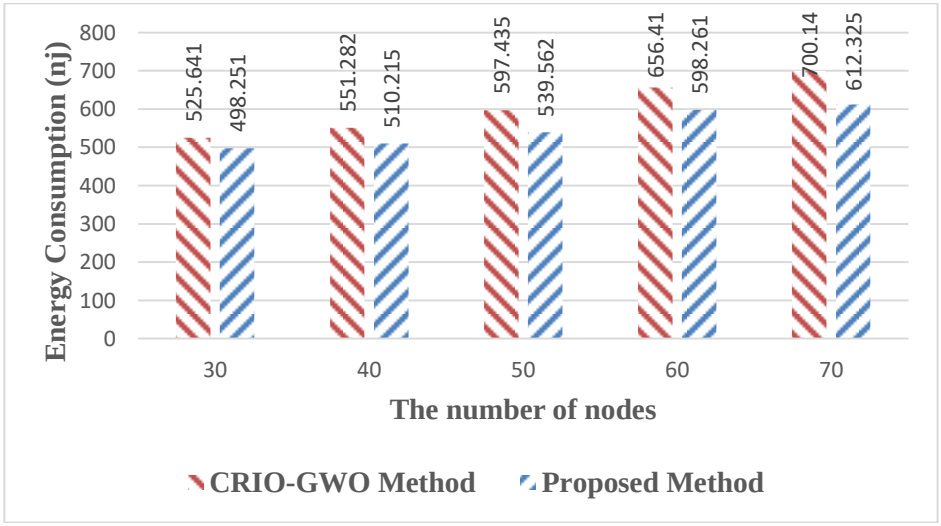

Fig. 12. Energy consumption vs. the number of nodes.

V. CONCLUSION

In general, cloud computing refers to the provision of dynamically scalable and virtualized resources over the Internet as services. Depending on the user's needs, a variety of cloud-based services may be delivered, which are often composited to meet those requirements. To combine and consolidate cloud services, service composition is emerging as a universal technology. By combining previously existing services, this idea aims to reduce costs and improve efficiency through a new cloud service. An IoT-enabled cloud service composition method using the fuzzy logic system and FOA is proposed in this paper. Energy, reliability, delay, and availability are the objective parameters in the proposed method. There are three main steps involved in this method in order to provide the proper nodes for the composition of cloud services. First, the requirements to start the algorithm are provided, like initialization of parameters and the dataset values normalization. Second, fuzzyification is performed on the parameters after determining the input parameters. Third, the firefly algorithm selects the most suitable nodes (resources) for service composition. Finally, in the fourth section of this paper, the proposed method is simulated using Matlab. A comparison of the proposed method with previous ones showed that the proposed method performed better in terms of response time, availability, and energy consumption. The dynamic nature of the cloud may be considered in our future work. A cloud's processing capabilities are limited, and the number of atomic services may change over time. Future research should focus on how to compose atomic services. In addition to addressing the energy consumption issue, we hope to maintain other metrics in the scheduling, including reliability, stability, trust degree, etc.

REFERENCES

[1] Vahedifard, F., et al., Artificial intelligence for radiomics; diagnostic biomarkers for neuro-oncology. World Journal of Advanced Research and Reviews, 2022. 14(3): p. 304-310.

[2] Saeidi, S.A., et al. A novel neuromorphic processors realization of spiking deep reinforcement learning for portfolio management. in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2022. IEEE.

[3] Akhavan, J. and S. Manoochehri. Sensory data fusion using machine learning methods for in-situ defect registration in additive manufacturing: a review. in 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). 2022. IEEE.

[4] Khosravi, F., et al. Implementation of an Elastic Reconfigurable Optical Add/Drop Multiplexer based on Subcarriers for Application in Optical Multichannel Networks. in 2022 International Conference on Electronics, Information, and Communication (ICEIC). 2022. IEEE.

[5] Khosravi, F., et al., Improving the performance of three level code division multiplexing using the optimization of signal level spacing. Optik, 2014. 125(18): p. 5037-5040.

[6] Haghshenas, S.H., M.A. Hasnat, and M. Naeini, A Temporal Graph Neural Network for Cyber Attack Detection and Localization in Smart Grids. arXiv preprint arXiv:2212.03390, 2022.

[7] Taami, T., S. Krug, and M. O'Nils. Experimental characterization of latency in distributed iot systems with cloud fog offloading. in 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS). 2019. IEEE.

[8] He, P., et al., Towards green smart cities using Internet of Things and optimization algorithms: A systematic and bibliometric review. Sustainable Computing: Informatics and Systems, 2022. 36: p. 100822.

[9] Meisami, S., M. Beheshti-Atashgah, and M.R. Aref, Using Blockchain to Achieve Decentralized Privacy In IoT Healthcare. arXiv preprint arXiv:2109.14812, 2021.

[10] Mehbodniya, A., et al., Modified Lamport Merkle Digital Signature blockchain framework for authentication of internet of things healthcare data. Expert Systems, 2022. 39(10): p. e12978.

[11] Pourghebleh, B., et al., A roadmap towards energy-efficient data fusion methods in the Internet of Things. Concurrency and Computation: Practice and Experience, 2022: p. e6959.

[12] Kumar, A., et al., Smart power consumption management and alert system using IoT on big data. Sustainable Energy Technologies and Assessments, 2022: p. 102555.

[13] Pourghebleh, B., et al., The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments. Cluster Computing, 2021: p. 1-24.

[14] Ataie, I., et al. D 2 FO: Distributed Dynamic Offloading Mechanism for Time-Sensitive Tasks in Fog-Cloud IoT-based Systems. in 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC). 2022. IEEE.

[15] Bermejo, B., C. Juiz, and C. Guerrero, Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance. The Journal of Supercomputing, 2019. 75(2): p. 808-836.

[16] Sefati, S., M. Mousavinasab, and R. Zareh Farkhady, Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation. The Journal of Supercomputing, 2022. 78(1): p. 18-42.

[17] Najafizadeh, A., et al., Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Cluster Computing, 2022. 25(1): p. 141-165.

[18] Hayyolalam, V., et al., Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends. Concurrency and Computation: Practice and Experience, 2022. 34(5): p. e6698.

[19] Hayyolalam, V., et al., Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. The International Journal of Advanced Manufacturing Technology, 2019. 105(1-4): p. 471-498.

[20] Praveenchandar, J. and A. Tamilarasi, Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 2021. 12(3): p. 4147-4159.

[21] Dorsala, M.R., V. Sastry, and S. Chapram, Blockchain-based solutions for cloud computing: A survey. Journal of Network and Computer Applications, 2021. 196: p. 103246.

[22] Gabi, D., et al., Cloud customers service selection scheme based on improved conventional cat swarm optimization. Neural Computing and Applications, 2020: p. 1-22.

[23] Kritikos, K. and D. Plexousakis. Multi-cloud application design through cloud service composition. in 2015 IEEE 8th international conference on cloud computing. 2015. IEEE.

[24] Huo, Y., et al., Discrete gbest-guided artificial bee colony algorithm for cloud service composition. Applied Intelligence, 2015. 42(4): p. 661-678.

[25] Liu, Z.-Z., et al., Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition. Information Sciences, 2016. 326: p. 315-333.

[26] Huang, J., et al. QoS correlation-aware service composition for unified network-cloud service provisioning. in 2016 IEEE Global Communications Conference (GLOBECOM). 2016. IEEE.

[27] Karimi, M.B., A. Isazadeh, and A.M. Rahmani, QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm. The Journal of Supercomputing, 2017. 73(4): p. 1387-1415.

[28] Singh, A., D. Juneja, and M. Malhotra, A novel agent based autonomous and service composition framework for cost optimization of resource provisioning in cloud computing. Journal of King Saud University-Computer and Information Sciences, 2017. 29(1): p. 19-28.

[29] Wang, S., et al., Towards green service composition approach in the cloud. IEEE Transactions on Services Computing, 2018. 14(4): p. 1238-1250.

[30] Jian, C., M. Li, and X. Kuang, Edge cloud computing service composition based on modified bird swarm optimization in the internet of things. Cluster Computing, 2019. 22(4): p. 8079-8087.

[31] Jin, H., et al., Eagle strategy using uniform mutation and modified whale optimization algorithm for QoS-aware cloud service composition. Applied Soft Computing, 2022. 114: p. 108053.

[32] Tao, F., et al., Correlation-aware resource service composition and optimal-selection in manufacturing grid. European Journal of Operational Research, 2010. 201(1): p. 129-143.

[33] Zhang, Y., et al., Long/short-term utility aware optimal selection of manufacturing service composition toward industrial internet platforms. IEEE Transactions on Industrial Informatics, 2019. 15(6): p. 3712-3722.

[34] Perumal, B. and A. Murugaiyan, A firefly colony and its fuzzy approach for server consolidation and virtual machine placement in cloud datacenters. Advances in Fuzzy Systems, 2016. 2016.

[35] Singh, U. and R. Salgotra, Synthesis of linear antenna arrays using enhanced firefly algorithm. Arabian Journal for Science and Engineering, 2019. 44(3): p. 1961-1976.

[36] Kashyap, N., A.C. Kumari, and R. Chhikara, Multi-objective Optimization using NSGA II for service composition in IoT. Procedia Computer Science, 2020. 167: p. 1928-1933.

[37] Sefati, S. and N.J. Navimipour, A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm. IEEE Internet of Things Journal, 2021. 8(20): p. 15620-15627.

[38] Alsaryrah, O., I. Mashal, and T.-Y. Chung, Bi-objective optimization for energy aware Internet of Things service composition. IEEE Access, 2018. 6: p. 26809-26819.

[39] Sun, M., et al., Energy-efficient IoT service composition for concurrent timed applications. Future Generation Computer Systems, 2019. 100: p. 1017-1030.