2528-9705

Journal of Organizational Behavior Research

Vol.: 10, Is.: S1, Year: 2025, ID: 25S0-1034

Available online

https://odad.org/

Combining Quality-Aware Cloud Services in the Internet of Things Application with the Moth-Flame Algorithm

Bahram Nasiri[1]*

[1]Department of Computer Engineering, Faculty of Engineering, Tehran Branch, Islamic Azad University Science and Research of Tehran, Iran.

***Corresponding Author**

E-mail: bahram09211984@yahoo.com

ABSTRACT

With the increase of users in the Internet of Things services and cloud computing, the providers are encouraged to present services with different functional and nonfunctional features (service quality) in the service pool. Based on the demand and supply rules, alongside the considerable growth in the number of provided services, these cloud service providers are facing tough competition for improving their service quality. This competition creates a tough and complex NP-Hard process of selecting and mixing the simple services for providing composite cloud services. The method of selecting the appropriate services from the available pool and overcoming the limitation determine the importance of different service quality parameters; also focusing on dynamic features and the fast changes in service properties are the most important matters that require analysis. This study presents a moth-flame algorithm-based method with service quality-awareness in the Internet of Things after the analysis of different service composition solutions; the results of the study show that the proposed method has a good degree of convergence for combining services with proper optimization. Also, the results show that the quality of the composite services created by the proposed method is better than other similar optimization algorithms.

Keywords: Cloud Computing, Internet of Things, Service Composition, Quality, and Moth-Flame Algorithm.

Introduction

Fast development in the usage of cloud computing leads to the release of more cloud services in the worldwide service pool. Due to the existence of complex and diverse systems, a simple service cannot provide the requirements of many real- world operations; on the other hand, to complete a complex system, a group of such simple services needs to work with others. Therefore, the need arises to create a system composition method in cloud computing. The process of introducing, requesting, and connecting to services, as shown in Figure 1, can be implemented in a way so that the service providers would introduce themselves to the broker for providing the user requests. Still, the users send their service requests to the broker, who must select the best services from an available collection based on the user's needs and orientation. The broker requests the service providers to present the selected services to the users based on some pre-defined rules and agreements. Cloud computing is a promising technology that facilitates the processing of large data. Allocating virtual machine optimization to physical hosts in a high-scale internet infrastructure will lead to energy reduction in the datacenters. Also, this can prevent environmental pollution and increase efficiency. Newly developed algorithms can increase the energy efficiency of cloud computing. Internet of Things (IoT) is a leading technology in always available computing and a key solution for providing smart environments. With the dawn of virtualization technology, the usage of computing resources as virtual machines begins.

This technology has many advantages, some of which are: improving resource productivity, reducing costs, and ease of server management. Different types of data need to be gathered, analyzed, divided by data type, and predicted in the Internet of Things application. Cloud computing is a model for giving networks access to a collection of configurable computing resources (for example networks, servers, storage, applications, and services) that can be provided with speed and with the least managerial effort or service provider interaction. In high-scale Internet of

Things infrastructure, the cloud plays an important role and many firmware technologies are designed for the Could of Things, and many new services are being provided for the smart city scenario.

Increasing the number of available Internet of Things services will lead to the increase of services with similar applications in different servers. These similar services are in different locations with distinguished QoS parameters. Therefore, when trying to select a single service from among many different similar services on separate servers, the appropriate techniques should be used to find the highest QoS degree based on the needs and priorities of the user. Service composition needs to be designed dynamically and automatically based on the substantial changes in the cloud space, available services, and the needs of the final user. Therefore, one of the important problems of this field is selecting simple and efficient services for creating a complex composite service. The problem of service composition in cloud computing can be defined as what simple services must be selected to create a complex composite service to meet the application and QoS needs of the final user. The service composition problem in the web is considered as an NP-hard problem, because of the abundance of different effective parameters and high numbers of available simple services in the service pool by the service providers.

We assume that any Composite Service (CS) in the cloud includes n Unique Service Solutions (USS) and P number of QoS parameters. To finish as CS, a composition of successive unique services must be sequentially implemented in the workflow (WF) of a person, just like Figure 1.

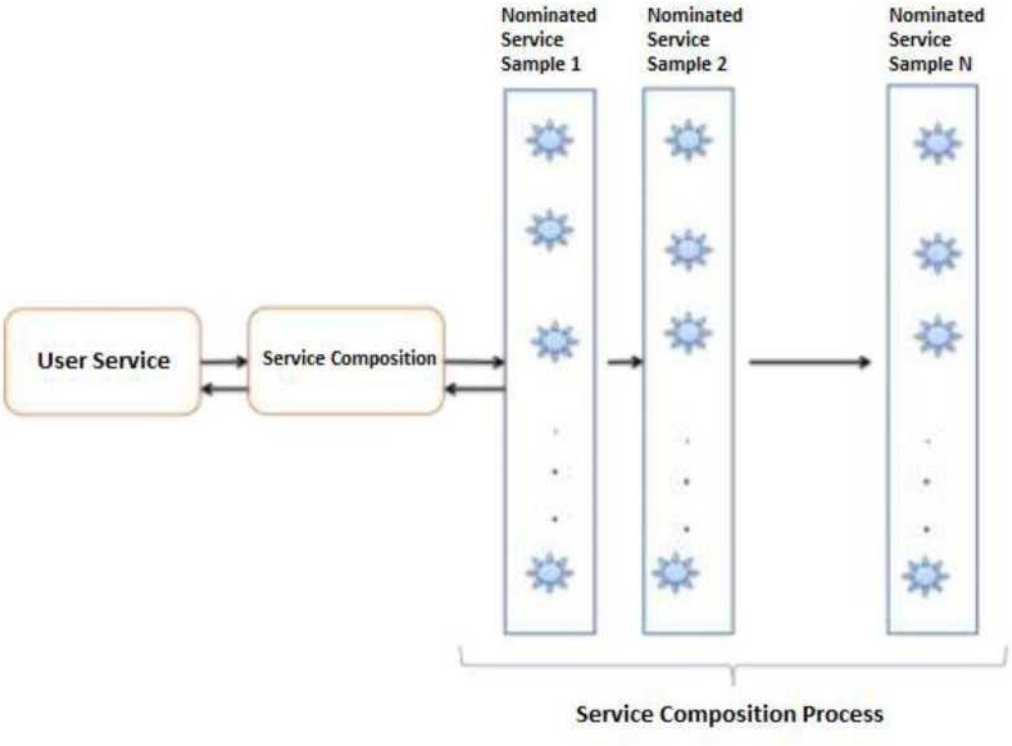

Figure 1: Web Service Composition Process

Because the cloud service composition problem in the Inter of Things ranks NP- Hard problem, different exploration methods can be used to solve this problem. The purpose of this thesis is to use the Moth-Flame algorithm to find the best near- optimized composite web service.

Research Background

Cloud service composition in the Internet of Things applications is a problem with many potential solutions, but there are only one or few efficient solutions among these. Therefore, cloud service composition is known as an optimization problem. Different solutions for cloud service composition in the Internet of things applications are presented in the following.

Zhu *et al*. (2018) have proposed linear programming for optimizing virtual machine resource allocation in the internet of things applications. Creating an efficient collection of media services has been considered to prepare a composite service for virtual machine resource allocation. To reach this goal, the authors have mapped the virtual machine resource allocation on the multi-dimensional binpacking problem and used linear programming as a solution. They also presented a custom subtractive best proportion method for solving this problem that will considerably increase the chance of finding proper results in linear programming but cannot completely guarantee an efficient solution. To analyze the two proposed methods, the problems of the created composite service were selected randomly and then analyzed using fractional knapsack mapping, and then the allocation results were compared using a round-robin method.

Yeganeh *et al*. (2010) and Rai *et al*. (2015) have proposed two-step methods for creating composite services with the least possible execution cost. The first includes using a state transition matrix for analyzing the dynamic execution process of composite services. In this phase, each state of the composite service is modeled in a state vector transition used for creating the state transition matrix. The execution cost of each composite service can be calculated using the transition matrix. In the second phase, the Business Process Execution Language for Web Services (BPEL4WS) is used for finding an optimized solution. Since the BPEL4WS process is rather time-taking as a recognized distributed traffic model, scientists have divided this process into three parts that are executed using a separate computer. A three-layer hierarchical structure was proposed for considering the user configuration and the features of different resources in the same place; first includes selecting an optimized service; the second layer is the benchmark layer and includes timeliness, stability, and security using which services are divided into three categories; and the third layer is used for showing nine other designed QoS parameters. Then the other steps of the proposed SSUP are performed, which are including creating and normalizing a user needs matrix, creating QoS configuration weight using a hierarchical analysis process, creating and normalizing a service specifications matrix, and the calculations similar to service demands. The empirical results show that this algorithm can have a better performance in solving the composite service problem than the analytic hierarchy process (AHP). Worm *et al*. were successful at achieving two opposite goals for providers which are maximizing income while guaranteeing quality.

Ben *et al*. (2017) presented the Cloud Feature Models (CFMs) based on variable modeling as a mechanism for explaining cloud services alongside the under needs for laying the proper groundwork for the process of selecting their web service. CFM is presented using a directional graph in which the nodes represent the services and the edges represent the relationship between the services. Moreover, the web service selection process, must meet the user goals and also respect their needs. To meet these goals, the user must first enter their goals and needs so that the process would start, afterwards, the web service selection service uses backup tools for creating a playlist of provided services based on the input model. Focusing on the dynamic features of cloud computing and updating the graph during its execution can improve the performance of this proposed method.

Karan *et al*. (2020) have presented a web-based semantic method for compositing web services using a Bayesian choice. The authors have implemented a Bayesian method for web service semantics that creates a cloud-based exploration diagram. They have also found the relationships in this graph that can be explained using the Markov chain. Also, they might be able to use an equation for the cosine theorem to find the similarities between the services, which play an important part in recognizing web service relationships and developing a composite system. The proposed method has been implemented in Amazon methods and can be compared with available approaches.

Rajeswari *et al*. (2014) propose their method using three decision-making criteria: service availability during the decision-making, execution cost, and rest time till the deadline. While considering the response time and emphasis on service availability, Diploma Programme (DP) was used to achieve the main purpose of this study, in which it's necessary to store average results for selecting the best service from all the available services. Also in case of a deadline, a decision rule will select the services with the least cost for the remaining work. The proposed algorithm is used for compositing static and dynamic services, but uses real-world datasets and criteria, and also this study omits calculation time and complexity.

Fekih *et al*. (2017) propose a two-step method for meeting the user's QoS needs, the first step of which tries to select a single service that provides the two basic functional needs of the user and eliminates the other remaining needs. Then a virtual network of selected service providers is created and modeled using a directed acyclic graph (DAG). In

the second step, if there are only the QoS needs of one user, there is only a need for the implementation of an algorithm for determining the shortest path in the graph. If there are two QoS parameters, this problem turns into a path with multiple limitations issues. CAGF uses a backup graph for considering the first QoS as its weight and merges the second parameter with the expanded connection between vertices, in which the primary two DAG weights are changed into one DAG weight (that can be solved using the last research's method). The researchers have also proposed a method for considering more QoS parameters. Run time and returned path weight (final QoS) are the two parameters that are considered for determining the method's performance while being compared with available DAG-based algorithms. Still, creating multiple graphs in the algorithm of one activity is time-consuming and increases the run time. This interesting study can be enriched by using known datasets.

Yu *et al*. (2012) have implemented a service quality value gathering algorithm based on communication process algebra to simplify service composition on cloud computing without weight prioritizing. In which an artificial neural network has been used to focus on the needed services without any pre-determined priority parameter from the user. Perceptron of the three-layer neural network of this study was used by implementing the backpropagation and mean squared error algorithms to identify the network and configure its weights with random initial inputs. In the presented neural network, the number of input layer neurons is equal to QoS parameters. The number of second-layer neurons is equal to the first layer, and the third layer has one neuron for one output. During the learning process, the primary user information is given to the neural network to understand the proper weight values. Weight parameter productivity was determined as the goal function by using the mean squared error. In this study, a series of data and parameters are not analyzed and the results are not compared. A new model for calculating the QoS of composite services was introduced to divide QoS parameters into the ascending, descending, and equal QoS characteristics with the help of simple weights for normalizing the values of these parameters. These authors also proposed a genetic roulette wheel selection algorithm for solving the service composition problem in which a chromosome is selected for launching a cross-operation. The proposed method uses QoS accessibility as the proportionality value.

Yan *et al*. (2013) used the Markov decision process model to solve the problem of automated composition in the dynamic environments of the cloud. They have considered a complete set of all possible service compositions alongside all possible actions for chaining the compositions because of adding new valid produced and presented services. Also, a reward layer was used for an optimized learning and composition policy for the request of users who use value tools such as the feedback of previous users and the information related to the previous components. The reward function results are used by the operator for the next decision-making and then updated using reinforcement learning techniques. Judging this study is impossible due to the lack of proper performance results analysis and comparison. Due to the ever-increasing importance of networks in cloud-based service composition, this approach was proposed separately by considering non-network QoS and service networks. Therefore, real network delay among the intended services and their users was modeled with low time complexity to select services with lower delay. The researchers, also propose a QoS equation for the calculation of the network's QoS, delay, and transfer rate. In the final step, the proposed method of creating service compositions was designed using genetic algorithms and then compared against the Dijkstra and random selection algorithms. The results of this interesting study can be enriched by using real-world datasets.

Puttonen *et al*. (2015) proposed a service level agreement (SLA) based service composition algorithm using game theory. In this paper, the agreements consist of four parts which are the main agreement information, the service seller and consumer information, the service type and parameters, and a series of responsibilities for the service seller and consumer. To create an agreement, service composition was considered as a dynamic multifold game in which the sellers and consumers are players that are after their own goals. In this competitive game, each consumer must present a price for each requested service based on effective parameters, and then communicate this proposed price with the other consumers so that the sellers could present their own price base on demand levels and service quality while considering these proposed prices. The reliability of this method is limited due to a lack of comparisons with other methods and real-world datasets.

Sun *et al*. (2019) designed a finite state machine (FSM) to consider service correlation and the task execution sequence; each machine contains a group of services with invocation limitations and an execution sequence. The proposed method contains two steps. In the first step, a service composition tree and a goal process are created based on society.

A pruning quality is used for omitting the wrong paths from the tree and improving the processing time. The quality of all remaining paths is calculated in the second phase based on the user's needs, and the path with the highest quality is selected as the final solution. A dichotomous simple additive weighting technique is used for reaching these goals in which, during the first step, the paths are scaled into positive and negative parameters, and then, during the second step, the QoS of all paths is calculated. The authors were able to prove the productivity of their proposed method by comparing its run time with the enumeration method run time.

Lee *et al*. (2011) proposed a method based on an index structure tree to simplify the information insertion and retrieval process for cloud service providers. In the proposed structure there are special locations for saving different attributes like service type, security, service quality, and unit measurement and pricing. To increase information management and proper provider discovery speed, service providers with similar attributes must be stored in adjacent rows. The authors also suggested a search algorithm based on the designed structure for finding the providers in the database. After finding K near-efficient providers, a refining method has been designed to reduce the number of selected providers and organize them based on their Hamming distance, which requires starting with an efficient seller and ascending movement to ease the selection of better providers.

Tout *et al*. (2015) provided a new framework for service composition based on content in which the composition operator is responsible for receiving the requests and presenting service management. This operator analyzes each request and divides them into the needed basic services. The service dependencies are recognized using a knowledge base and a service retrieval section, which extracts the information of similar services. There is a packing motor that creates new software packs using current and new compositions and submits them in the catalog. Finally, an information provider section presents this service catalog to the service provider to get the proper permission.

Han *et al*. (2016) presents a systematic similarity-based framework for the automatic recognition of service involvements and providing the user policies and needs. The first phase of this proposed method is the conflict analysis section that is created from the comparator and analyzer sub-sections. The comparator checks for the user policies and needs following their priorities and the analyzer tries to discover conflicts between the user needs and its dependent relationships while using Satisfiability Modulo Theories (SMT). Filter, recognizer, and solver are the three sections of the second phase which is called solution deduction. This section finds proper simple services and removes services that oppose the policy. A series of simple services are determined according to the user needs and the best possible service composition based on user policies and needs is created. This method uses the backtracking algorithm for the assigned tasks.

Wang *et al*. (2019) have proposed a branch-and-bound algorithm for solving the MMKP problem and finding the advantages of a heterogeneous multiprocessing environment; in which each node of the decision tree is a sample from a real process that can be assigned to separate computing nodes. To analyze this algorithm, the authors have randomly produced four different problems with different sizes, run them in a series and simultaneously, and then compared their results. The biggest problem of this proposed algorithm is its runtime and visual complexity. This performance analysis can be more complete and reliable if the results are compared with similar algorithms in real-world datasets.

A two-phase solution has been proposed in the Calfer method for solving the high run time problem. In this method, the first stage is similar to the last research, while the second step takes solutions usually used for similar requests and tweaks them based on the new situation. After proposing a new method of conserving cloud resources, a new accordance algorithm called SMA is implemented for checking the compatibility of output parameters of service with the input parameters of another one. In this algorithm, the service compatibility problem has been considered for calculating the semantic similarities of the input and output of different services. The compatibility degree of each set of services that are saved in the table will lead to the production of a weighted directional graph in which finding all possible paths between two nodes will include all possible composition methods. The researchers have proposed an improved fast EP algorithm called the B+EP fast algorithm for finding all the possible paths in the least possible time. Yet, creating and searching in both graph stages leads to an increased runtime and lower algorithm performance, especially when the problem size and number of needed services increases.

Bashari *et al*. (2018) proposed a trust-based method and framework for solving the proposed service composition problem while considering trust as a conceptual probability by which the composite service performs the user's intended activities. Using trust in the service composition process is divided into three sections which are service

selecting reliability, composition process reliability guarantee, and created design connection reliability. The necessity analyzer categorizes the different user necessities such as functional and non-functional needs and expected input-output parameters. The service recovery agent retrieves the service information from the resource pool using inquiry requests. Inadequate services are removed from the candidate list using a service filter; the remaining services' names and service types are recognized using the WSDL analyzer. The cluster component, format creator, and connection optimizer agent are responsible for checking the possibility of service composition, checking mathematics equations, and evaluating connection design reliability.

CloudRecommender is a three-layer cloud-based service composition system proposed by Laleh *et al*. (2018). The first layer is a configuration management layer that includes web service ontology and cloud quality ontology. Due to the existence of two sections for discovering services based on their performance and QoS parameters, the services are mapped into a logical model and data structure. The second layer is about logic which is implemented for selecting single services in SQL phrase format and includes criteria, views, and stored measures. The third layer is a widget that divides the user interface into four objects such as computing resources, storing resources, network, and recommendation resources.

Xu *et al*. (2018) presented a parallel form of particle swarm optimization with a chaos algorithm for solving the service composition problem. The researchers have tried to dynamically change the sequence length following the evolutionary position of the solution. Also, they implemented the roulette wheel algorithm before executing the chaos operator to eliminate the randomly created inappropriate solutions and escaping their destructive consequences. Since the main goal of this study is to reduce the runtime, parallelization of the proposed method was considered as a solution. To achieve this goal, a full connection topology was selected due to its high searchability and message passing interface. In the end, a new migration method called Way-Reflex Migration was introduced to decrease the fully connected topology connection workload. The proposed method showed better appropriation and runtime results in comparison with the genetic, chaos genetic, and chaos optimization algorithms.

Yu and Li (2015) proposed a new framework for selecting service compatibility in web service composition using the ant colony algorithm. This framework extracts customer preferences immediately after receiving a request. Then, the Euclidean distance is used for selecting the closest services to the user priorities and then suggests them to the service adaptor. Finally, the service adaptor dedicated the best possible option from the suggested services to the user based on base device compatibility and chosen service effectiveness. To achieve the context compatibility service based on the input information, a recognized fuzzy map model is used in the service adaptor module. The weakness of this method is the fact that the proposed framework can be used for selecting only one simple service. Also, this approach has not been compared with other methods.

Kotekar *et al*. (2016) also used the particle swarm algorithm with a dynamic fitness function. In this method, the genome array structure is of integers and each of its entries refers to one service. Here, the crossover operator is two-point and the mutation operator acts randomly. But on in other researches, the genome structure is considered as an array each entry of which refers to a qualitative feature instead of a service. Here the crossover operator is single-point and the mutation operator acts as before.

Boukh *et al*. (2019) present another difference between different multi-purpose algorithm implementation in the coding discussion. Some encoding measures select a single-dimensional chromosome, although the increase in the number of candidate services will reduce the reliability of these chromosomes; on the other hand, this method cannot show semantic information. Therefore, some methods are based on the relationship matrix encoding algorithm; although this method creates unauthorized figures and reduces efficiency.

Zeng *et al*. (2018) discuss tree encoding which can determine the relationship of different compositions. This model supports design during runtime; and as was mentioned a lot of composition work was done in this research to avoid the local optimization trap. Here, two different security system algorithms were used alongside the genetic algorithm with different encoding methods and mutation processes.

Mousa *et al*. (2016) proposed an improved version of the Imperialist Competitive Algorithm for solving the service composition problem based on qualitative characteristics. In this method, the revolution operator was added to the base algorithm to escape the minimum trap. Adding other operators such as the crossover and mutation operators from the genetic algorithm to the Imperialist Competitive Algorithm is another solution for increasing the effectiveness and

solving the problems of this algorithm. For example, the crossover operator makes sure that the environment is fully searched or the mutation operator helps prevent the iteration of algorithms in the loop.

Liu *et al*. (2014) analyze service transaction properties and functionally focus on unique service composition methods, and then formalized dynamic service transaction and composition based on service quality. The ant colony algorithm was used for modeling and solving the loopless directional graphs by searching for a near-optimized solution. The optimization algorithm is designed to search for a near-optimized solution for the composite service problem in a reasonable time frame by supporting real-time and dynamic applications. Based on experimental studies, the results show that the proposed method can present an optimized method.

Materials and Methods

The proposed Moth-Flame algorithm is created from different components that are discussed later on. The Moth-Flame algorithm is an optimization method that uses the natural selection theory. This study uses the meta-heuristic moth-flame algorithm for finding active nodes in social media. The moths experience two important stages in their lives: the first worm moth stage and the second adult moth stage. The most important truth about moths is that they move at night. They fly at night using the moonlight. They use a transverse navigation method for moving at night. In this mechanism, the moth moves at a fixed angle with the moon. This mechanism is appropriate for direct movement in long distances.

We have witnessed that the moths move in a spiral manner around smart lights. This is due to the ineffectiveness of transverse movement when the light source is close to the moth. The moth tries to find a transverse path towards its destination, but since the light source is too close to the moth, it chooses a deadly and aimless spiral for reaching the light.

Therefore, finding the search space around the best possible locations is guaranteed due to the following reasons:

A. The moths update their location in fields surrounding the best available solution to date

B. The moths are after the flame, therefore, the best solution in each iteration changes, and the moth must update its location based on the flame position. So, the moth situation change might happen around different flames.

One of the problems of this algorithm is the fact that updating the moth's locations considering the different search spaces locations can reduce the effectiveness of this algorithm in finding a solution. To solve this, a flame number mechanism is defined as follows:

In which, ***l*** is the iteration number, ***N*** is the maximum flame numbers and ***T*** is the number of algorithm iteration. In the first iteration, there are ***N*** flames. Yet, the moth only updates its location in the final step based on the best flame.

$$\mathbf{FlammeNumber = round(N - l \times \frac{N-1}{T})}$$

The gradual decrease in flame numbers will balance the search space exploration and exploitation.

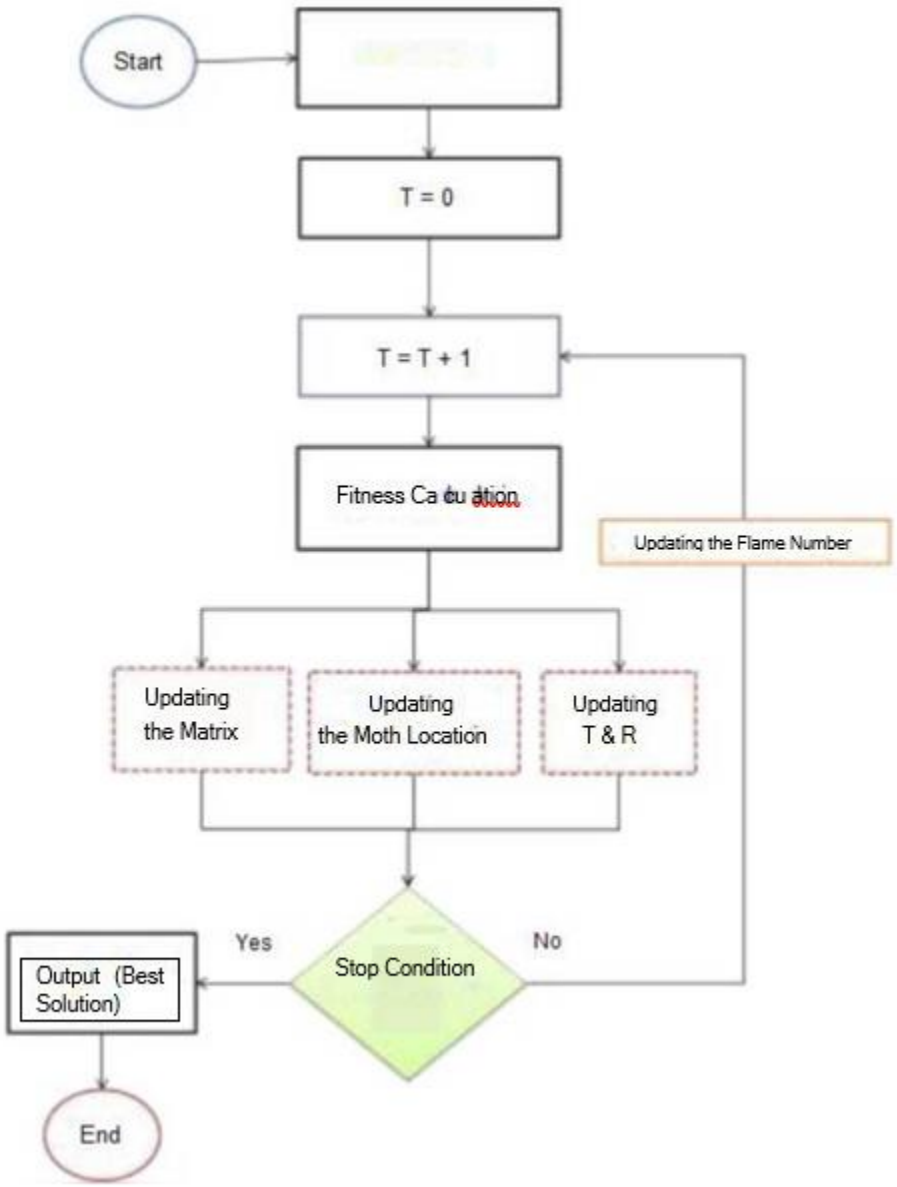

Figure 2: Moth-Flame Algorithm Flowchart

The Problem's Mathematical Model

The cloud service composition problem in the Internet of Things is a problem of finding a set of candidate cloud services with different applications that follow the user limitations and optimize the goal function. This section formally addresses this issue. A sample of the cloud service composition with service quality awareness in the Internet of Things is formally proposed as follows:

A service composition request in the Internet of Things can be modeled as the workflow using a Directed Acyclic Graph (DAG), G = (V,E)

- $V = (T_1, 72,..., T_n]$ in which n is the number of tasks in the workflow.
- E: Set of edges that show the priority of tasks.
- Each task *Ti* $(l < i < ri)$ in the workflow has a series of service candidates $CS_i = (CS^1, OS'^2, ..., CS™^1\}$ in which $CSj\ (l < 1 < m_i)$ is a candidate cloud service.
- $m\{.$ The overall available candidate services for the *Ti* task.

- Each candidate service ***CS-*** has a series of service quality information $QoSj = (Q_1, Q_2, ..., Q_k\}$ in which Q_i ***(l < 1 < K)*** shows one of the qualities of that cloud service.
- Cloud service-related quality information is saved in the quality storage.
- K: The number of cloud service qualities used in the service quality model.
- QC: Global restrictions series determined by the user $QC = (C_V\ C2, ..., C_K)$.

Considering these points, the purpose of the quality-aware service composition problem in the Internet of Things is to find near-optimized composite cloud services.

Service Quality Model

$$\frac{/,\_\_\_\_\_}{\blacksquare}\ Qj < {}^{c}j \quad \text{if Qjis additive}$$

$$(1) \qquad \begin{matrix} i=1 \\ n \\ \wedge i=1 \end{matrix} \quad \text{if Qj is multiplicative}$$

In the proposed method, the Al-Masri *et al*. dataset was used as the service quality parameters of the Internet of Things applications. This database is called Quality of Web Service (QWS) and it presents a basis for service researchers[1]. The QWS dataset includes a collection of 2507 web services and their quality measurements. The quality parameter values are performed using their suggested Web Service

[1] http://www.uoguelph.ca/~qmahmoud/qws/index.html

Broker Framework. Each QWS record includes eleven parameters for a web service. The first nine parameters of each record are service quality parameters which are measured by the Web Service Broker Framework uring six days. The service quality values available in the dataset are the average of the measurements in the intended period. Table 1 presents a simple description of the nine service quality parameters in the Internet of Things applications.

Table 1. Describing the Available Service Quality Parameters in the QWS dataset

Parameter Name	Description	Unit
Response Time	Time for sending a request and receiving the answer	Millisecond
Availability	Number of successful recalls out of all recalls	%
Throughput	Number of all recalls during a certain timespan	Recall per second
Successability	Number of responses to all requests	%
Reliability	Number of errors to all messages	%
Compliance	Compliance of the WSDL to WDFL characteristics	%
Best Practices	Compliance of service from the base WS-I profile	%
Latency	Time for the service provider to process a request	Millisecond
Documentation	Documentation measurement (descriptive labels) in WSDL	%

Because cloud servers are an expanded type of web service in the Internet of Things, we can use the QWS dataset for its service quality parameter values. By paying attention to the service quality parameter descriptions in table 1, you can infer that the Compliance, Best Practices, and Documentation parameters are fixed during the service runtime due to the successive recalls; therefore, we omit these three values and use the other six service qualities.

Service Quality Parameter Normalization

Different service quality-related parameters of a cloud service in the Internet of Things are measured using different units. This is while the goal function calculations require all parameters to be measured using the same scale. Considering this, all quality service parameters in an Internet of Things application must be normalized on the same

scale. In reality, service quality parameter normalization gives us a uniform measurement of these values. The general approach for this problem is to normalize all the values on a zero to one scale. Quality parameters can be divided into two groups: maximization and minimization parameters. Maximization parameters require their values to be maximized and minimization parameters require their values to be minimized.

Formulas (2) and (3) show the normalization rules for maximization and minimization parameters.

In these equations ***CS. Q**^l* is the i-th service quality value related to the CS candidate service and ***Ncsqi*** is the normalized value. Also ***Qmax*** and ***Qmin*** are the maximum and minimum values of the i-th parameters in all services.

(2) $$N_{CS.Q_I} \quad \text{Qmax CS- Q} \quad \text{Qmax\_\^Qv}_{-min} \quad \text{Qmax} - \text{Qmin} \quad \underline{1} \quad \text{Qmax} \quad \text{Qmin}$$

(3) $$N_{CS.Q^I} \quad \text{CS. Q}^{\text{I}} \text{ - Qi}_{\text{min}} \quad \text{Qmax ^ Qmin} \quad \text{Qmax} \quad \text{Qmin} \quad \text{1} \quad \text{Qmax Qmin}$$

Initialization

In the proposed algorithm, the moths are adjacent to the candidates, the problem variables are considered as the special position. Therefore, moths can move in a one-dimensional, two-dimensional, or multi-dimensional space. Since the Moth- Flame algorithm is a population-based algorithm, the Moth set is a n*d matrix shown from Equation (4).

In initialization, there needs to be an initial particle summation. In the Moth-Flame optimization algorithm, each flame is a path in which a composite cloud server is depicted with an n length array (number of active tasks). The ***i*** subscript of the array shows the candidate service ID that will execute the ***Ti*** task. Considering the fact that the initial particle summation will be equal to P, we will have the P*n matrix as the initial combination of the solutions.

(4) $$M = \begin{matrix} [^{m}i,i & {}^{m}1,d' \\ l^{m}n,1 & \hat{}n,d. \end{matrix}$$

Fitness Function

Providing the limitations set by the user alongside the optimization of a fitness function are the most important tasks that quality-aware composite cloud services face. The fitness function must optimize the service quality parameters for the composite cloud service. Considering the fact that the proposed model uses the six parameters of Response Time (Resp), Availability (Avail), Throughput (Throu), Successability (Succ), Reliability (Reli), and Latency, the Fitness function is defined as follows:

In which w1, w2, w3, w4, ws, and w6 are positive weights that determine the importance of each service quality parameter determined by the user.

After the initialization, the P function is iterated till the T performance is fixed. The P function is the main function that explores the surrounding space. As was mentioned, we use transverse orientation. To mathematically show this,

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

the position of each moth in respect to the flames is updated using the following:

(6) $$M_t = S(M_t, Fj)$$

In which ***Fj*** shows the j-th flame and ***Mi*** the i-th moth, and s is the spiral function of the Moth-Flame algorithm.

(7) $$S(M_b Fj) = Dj \times e^{bt} \times cos(2nt) + Fj$$

Here, D_i represents the i-th moths' distance from the j-th flame. b is a fixed value for determining the logarithmic spiral and t is a random number between [1,-1]. D_i is calculated as follows:

(8) $D_i = |F_j - M_i|$

Considering these facts, the proposed method's semi-code is presented in Figure (3).

Input: Composition request as a workflow (DAG) and QoS constraints **Output:** Near-optimal composite cloud service

Update the number of flames (FlameNumber)

Initializing all the parameters Initialize the population of moths Calculate the objective values Equations (1) for all moths

for all parameters

update r and t

Calculate D with respect to the corresponding moth by Equation (3)

Update the matrix M with respect to the corresponding moth by Equations (4) and

(5)

Calculate the Influence of each node end

Calculate the objective values Update j-th flames by greedy strategy as Equations (6)

Apply later flames S the current best individual in the population. **end**

return S.

End

Figure 3: Proposed method semi-code

In the general moth-flame algorithm, all newly created moths are accepted and help for the next generation, yet our proposed method uses a greedy method for accepting moths with the appropriate performance in the algorithm. This greedy strategy can be explained as follows:

Here $xf+_w$is the newly created moth for the next generation. /(xf) and /($xf+^1$) are the fitness levels of the xf and $xf+^1$ moths.

(8) $\underline{yt+}\ 1$ X_i^{t+1} |/(^,$^{t+1}$)</(x?)|

$A^T i,new$ **X** **otherwise**

Figure (2) shows the moth structure in an M-dimensional space of attributes. The moth attributes are configured following Figure (2) for exploring the moth-flame algorithm space. A unique code is considered for each attribute in all moths P.

To escape local optimization and covering a bigger search space in each Internet of Things service, we have selected the ***t — th*** worst vertex instead of the worst one; meaning the K vertex calculated in equation (9) will be selected as the replacement.

i

(9) $$k = (1 + (n^{1-T} — 1)(rand()))1—$$

In which k refers to the service number selected from the organized rank list; n is the number of graph vertices and rand() is the random number generator function in the [0,1] range. The fixed **t** parameter was calculated using the trial and error method. This parameter limits the search space and searches for an answer more rigorously.

Results and Discussion

In the first section of the QWS dataset, there are 60 processes with a server capacity of 100. The best possible answer for the allocation problem of the following list is the allocation of 20. Our proposed method reached the approximate answer of 23. Our algorithm performed better than the Gray Wolf, and the Learning and Teaching algorithms. These results are shown in Table 1.

41.4000 30.7000 26.3000 43.0000 36.6000 26.8000 36.6000
25.2000 47.4000 25.4000 25.1000 49.5000 29.9000 28.7000
27.5000 25.2000 47.3000 36.1000 26.9000 37.0000 27.4000
43.9000 29.8000 25.8000 44.4000 26.9000 25.9000 47.2000
26.1000 41.9000 28.3000 27.2000 44.5000 28.8000 27.3000
37.2000 27.3000 35.5000 36.6000 27.1000 36.3000 32.0000
39.5000 35.1000 29.2000 35.7000 27.2000 26.2000 46.6000
25.2000 45.0000 34.7000 39.3000 35.0000 35.0000 25.5000
31.5000 27.5000 41.0000 29.8000

Table 1. Allocation with the proposed algorithm in the QWS5 dataset

Dataset	Grey Wolf	Learning and Teaching	Proposed Method	Optimized Answer
First Section of the Data Set	24	26	23	20

Table 2 shows service allocation in the QWS5 dataset in which each serve is shown with bin and the following numbers are the server ID. For example the bin1 server is followed by the 49.5,47.4 value, which means processes with the 47.4 and 49.5 weights are place in the first server. The other servers house different processes in a similar fashion.

bin1 49.5,47.4, bin2 47.3,47.2, bin3 46.6,45, bin4 44.5,44.4, bin5 43.9,43, bin6 41.9,41.4, bin7 41,39.5, bin8 37.2,37,25.5, bin9 36.6,36.6,26.3, bin10 36.6,36.3,27.1, bin11 36.1,35.7,27.5, bin12 35.5,35.1,29.2, bin13 35,35,29.9, bin14 34.7,32,31.5, bin15 30.7,30.3,29.8, bin16 29.8,28.8,28.7, bin17 28.3,27.5,27.4, bin18 27.3,27.3,27.2, bin19 27.2,26.9,26.9, bin20 26.8,26.2,26.1, bin21 25.9,25.8,25.4, bin22 25.2,25.2,25.2, bin23 25.1,

The empty unallocated space in the QWS5 for load balance is equal to 0.1304 or 13.04% which means that 0.8696 of this space is allocated.

In the optimal scenario, the sum of process values divided by the server space of 100 should be equal to 20. Of course, since this is a discrete problem there is a chance of load unbalance even in the optimal scenario. The comparison of these methods using more learning data is presented in Table 2. The test results for the Grey Wolf, and Learning and Teaching algorithms are taken from [6,7].

This table shows that the data increase trend leads to the reduction of the proposed method's accuracy. Yet, this method shows better results in solving the resource allocation problem than the Grey Wolf, and Learning and Teaching methods. In the second set of experiments, the problem gets more complex due to the higher data sparsity and higher server capacity, yet the proposed method's results seem more appropriate. Metaheuristic methods have inadequate performances while dealing with a high volume of processes which is due to the inappropriate local behavior in allocating processes in the server. This inadequate performance is sometimes present in the Learning and Teaching and sometimes in the Grey Wolf algorithm. Naturally, using the Moth-Flame algorithm improves this performance, because this algorithm is better suited for dynamic environments. But still, the percentage error from the optimal answer gradually decreases with an increase in the number of processes.

Table 2: Differences of the proposed method and other methods in allocation processes

Optimized	MFO	GWO	MFO[23]	Capacity	Number of	Dataset	Test
99	100	100	100	150	250	QWS1-U250_00	First
100	101	101	102	150	250	QWS1-U250_01	First
99	100	100	104	150	250	QWSU250_00	First
100	101	101	104	150	250	QWSU250_01	First
198	**201**	**203**	**206**	**150**	**500**	**QWSU500_00**	**First**
201	204	207	208	150	500	QWSU500_01	First
399	403	407	413	150	1000	QWSU1000_00	First
406	411	419	423	150	1000	QWSu1000_01	First
20	23	23	23	100	60	QWS5-T60_00	Second
20	23	23	23	100	60	QWS5-T60_01	Second
40	45	45	46	100	120	QWS6-T120_00	Second
40	45	45	47	100	120	QWS6-T120_01	Second
83	94	96	99	100	249	QWS7-T249_00	Second
83	95	98	103	100	249	QWS7-T249_01	Second
167	190	198	203	100	501	QWS8-T501_00	Second
167	191	201	207	100	501	QWS8-T501_01	Second

Comparing the results with the optimized conditions showed a good performance. A recent 2018 study showed that the CGA-CT and HI-BP methods had the best performance in resource allocation. The proposed method had a better performance even in comparison with these two methods. You can see this comparison in Table 3.
In Table 3, the U250 dataset had to allocate 250 processes which were done in a series of four experiments. The experiments were done using different quantities of the test data. These data were presented with 500, 1000, 120, and 501 quantities in the different U500, U1000, T120, T249, and T501 datasets.

Table 3: Comparing the proposed method with HI_BP and CGA-CGT

Optimized	MFO	CGA-CGT	HI_BP	Capacity	Number of	Dataset	Test
99	100	100	100	150	250	QWS1-U250_00	First
100	101	101	101	150	250	QWS1-U250_01	First
99	100	100	100	150	250	QWSU250_00	First
100	101	101	101	150	250	QWSU250_01	First
198	**201**	**201**	**204**	**150**	**500**	**QWSU500_00**	**First**
201	204	204	204	150	500	QWSU500_01	First
399	403	404	404	150	1000	QWSU1000_00	First
406	411	413	414	150	1000	QWSu1000_01	First
20	23	23	23	100	60	QWS5-T60_00	Second
20	23	23	23	100	60	QWS5-T60_01	Second
40	45	45	45	100	120	QWS6-T120_00	Second
40	45	45	47	100	120	QWS6-T120_01	Second
83	94	94	96	100	249	QWS7-T249_00	Second
83	95	97	101	100	249	QWS7-T249_01	Second
167	190	194	202	100	501	QWS8-T501_00	Second
167	191	199	204	100	501	QWS8-T501_01	Second

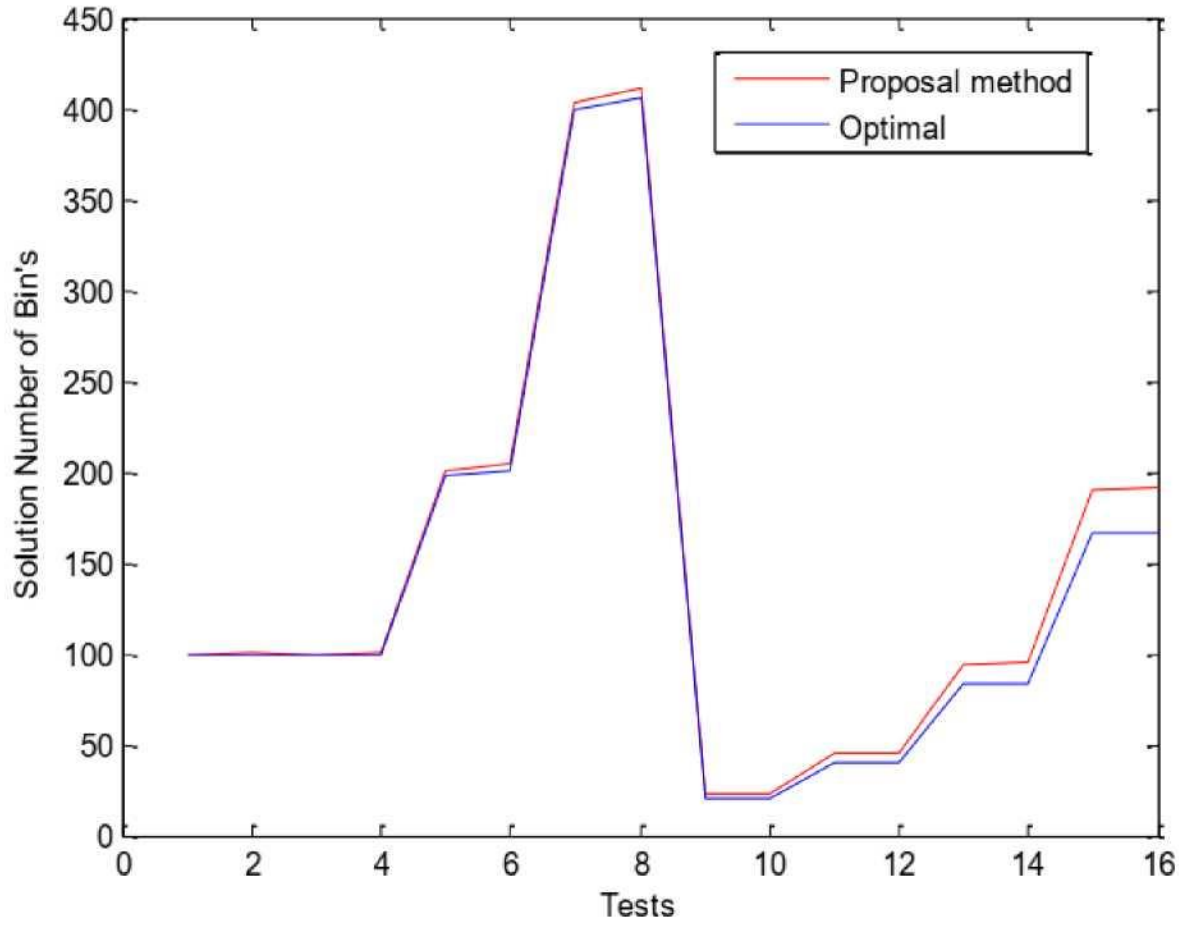

Figure 4: Showing the difference between the proposed Moth-Flame algorithm with the

optimized answers

Figure 4 shows the difference in the proposed method with the optimized answers which is completely acceptable. Table 4 shows the calculation of load balance quality in different experiments.

Table 4: Comparing load unbalance in the proposed Moth-Flame algorithm with other algorithms

Nflb-Proposal	Nflb-GWO	Nflb-TLBO	Capacity	Number of	Dataset	Test
0.0145	0.0145	0.0145	150	250	QWS1-U250_00	First
0.0193	0.0195	0.0195	150	250	QWS1-U250_01	First
0.0138	0.0145	0.0145	150	250	QWSU250_00	First
0.0178	0.0195	0.0195	150	250	QWSU250_01	First
0.0159	0.0163	0.017	150	500	QWSU500_00	First
0.0143	0.0147	0.0155	150	500	QWSU500_01	First
0.0097	0.0103	0.0113	150	1000	QWSU1000_00	First
0.0119	0.0127	0.014	150	1000	QWSu1000_01	First
0.1304	0.1304	0.1304	100	60	QWS5-T60_00	Second
0.1304	0.1304	0.1304	100	60	QWS5-T60_01	Second
0.1023	0.1111	0.1111	100	120	QWS6-T120_00	Second
0.1013	0.1111	0.1111	100	120	QWS6-T120_01	Second
0.101	0.109	0.117	100	249	QWS7-T249_00	Second
0.1203	0.1226	0.1263	100	249	QWS7-T249_01	Second
0.1107	0.1139	0.1211	100	501	QWS8-T501_00	Second
0.1219	0.1213	0.1257	100	501	QWS8-T501_01	Second

The flb is a load balance indicator, the higher values of which show a better load balance. A higher load balance is an indicator for the better usage of resources. In other words, the free recourse capacity has decreased; therefore an increase in the data will lead to an increase in the load balance of the proposed algorithm. The load balance of the current study is 80%. A higher than 80% usages of server resources will drastically increase the consumption quality. Also, using less than 20% of the resources will lead to much quality. Therefore, recourses that have filled their 20% maximum capacity must move to another server. The nflb load unbalance indicator is the opposite of the load balance indicator. Its decrease leads to a better load balance. The Learning and Teaching method due to its wrong recognition of teaching in experiences cannot deliver a good load balance with higher amounts of data. The proposed method showed good load balance even with data increase. The load balance indicator can show the proper allocation by itself.

7. Results

The potential of emerging Inter of Thing technologies for sensors and other connected equipment plays an important role in the next generation of industry and applications. High security and quality are rather important in the Internet of Things applications. Based on this application data security and quality resource consumption are important matters that will have a major effect on the success of the Internet of Things. Resource efficiency and quality saving are major problems in the Internet of Things applications. While creating a composite service, selecting the best possible services is important. Considering the ever-increasing number of cloud servers, the needed time for finding such composites will belong. This paper represents a new framework for creating quality-aware cloud-based service composites in the Internet of Things. The purpose of this method was to present a service composite with a proper fitness value. The proposed algorithm got implemented with different parameters and its results got analyzed. The Moth- Flame algorithm decreases the time needed for creating service composites, but its results are just near-optimized. To show the performance of each algorithm, it must be compared with previous ones. The comparisons with three other algorithms showed that the Moth-Flame algorithm response time was lower than similar methods and also this algorithm has proper stability that allows it to present near- optimized results.

Acknowledgments: None

Conflict of Interest: None

Financial Support: None

Ethics Statement: None

References

1. Asghari, P., Rahmani, A.M., Javadi, H.H.S., Service composition approaches in IoT: A systematic review, Journal of Network and Computer Applications (2018)

2. Z. Zhong Liu, Dian-Hui Chu, Cheng Song, Xiao Xue, Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition. Information Sciences 326 (2016) 315-333.

3. R T. Baker, M. Asim, H. Tawfik, B. Aldawsari and R. Buyya,An Energy-aware Service Composition Algorithm for Multiple Cloud-based IoT Applications, Journal of Network and Computer Applications,.2017.03.008

4. Son N.HanNoelCrespi, Semantic service provisioning for smart objects: Integrating IoT applications into the web, Future Generation Computer Systems,Volume 76, November 2017, Pages 180-197

5. Liu, Min, Wang, Mingrui, Shen, Weiming, Luo, Nan, & Yan, Junwei (2012). A quality of service (QoS)-aware execution plan selection approach for a service composition process. Future Generation Computer Systems,28, 1080-1089.

6. Mardukhi, Farhad, NematBakhsh, Naser, Zamanifar, Kamran, & Barati, Asghar (2013). QoS decomposition for service composition using genetic algorithm. Applied Soft Computing, 13(7), 3409-3421.

7. Youtao Zhang,(2017) Multi-objective Optimization based Ranking Prediction for Cloud Service Recommendation, Decision Support Systems (pp. 2254-2261).

8. Jiajun Zhou, Xifan Yao, (2017), Multi-population parallel self-adaptive differential artificial bee colony algorithm with application in large-scale service composition for cloud manufacturing, Applied Soft Computing Journal. 24, pp-360-371.

9. Fuzan Chen, Runliang Dou, Minqiang Li, Harris Wu, (2016), A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing, Computers & Industrial Engineering, 124, pp-568-573.

10. X. Xue, S. Wang, B. Lu, (2016), Manufacturing service composition method based on networked collaboration mode, J Netw Comput Appl, 59. 28-38.

11. C.atotha , G.R.GangadharanbUgo Fiore, Optimal fitness aware cloud service composition using modified invasive weed optimization, Swarm and Evolutionary Computation, Volume 44, February 2019, Pages 1071091

12. Ming Zhu, Jing Li, Guodong Fan, Kunsheng Zhao, Modeling and Verification of Response Time of QoS-aware Web Service Composition by Timed CSP, Computer Science, Volume 141, 2018, Pages 48-55

13. Gopal N. Rai, G. R. Gangadharan, Set Partition and Trace Based Verification of Web Service Composition, Computer Science, Volume 52, 2015, Pages 278-285

14. Soheil Hassas Yeganeh, Jafar Habibi, Habib Rostami, Hassan Abolhassani, Semantic web service composition testbed, Computers & Electrical Engineering, Volume 36, Issue 5, September 2010, Pages 805-817

15. Rihab Ben, Raoudha Ben Jemaa, Ikram Amous Ben Amor, Graph Planning Based

Composition For Adaptable Semantic Web Services, Computer Science, Volume 112, 2017, Pages358-368

16. Rajesh Karun, Ferhat Khendek, Roch H. Glitho, A novel architecture for Web service composition, Journal of Network and Computer Applications, Volume 35, Issue 2, March 2012, Pages 787-802

17. M. Rajeswari, G. Sambasivam, N. Balaji, M. S. Saleem Basha, P. Dhavachelvan , Appraisal and analysis on various web service composition approaches based on QoS factors, Journal of King Saud University - Computer and Information Sciences, Volume 26, Issue 1, January 2014, Pages 14152

18. Hela Fekih, Sabri Mtibaa, Sadok Bouamama, Local-Consistency Web Services Composition Approach Based On Harmony Search, Computer Science, Volume 112, 2017, Pages 1101111

19. Qing-mei YU, Lan WANG, Dong-mei HUANG, Fishery Web Service Composition Method Based on Ontology, Journal of Integrative Agriculture, Volume 11, Issue 5, May 2012, Pages 79799

20. Dan-feng YAN, Yuan TIAN, Jun-lin HUANG, Fang-chun YANG, Privacy-aware RBAC model for web services composition, The Journal of China Universities of Posts and Telecommunications, Volume 20, Supplement 1, August 2013, Pages 30-34

21. Juha Puttonen, Andrei Lobov, Maria A. Cavia Soto, Jose L. Martinez Lastra, Planning-based semantic web service composition in factory automatio, Advanced Engineering Informatics, Volume 29, Issue 4, October 2015, Pages 1041-1054

22. Ziheng Sun, Liping Di, Juozas Gaigalas, SUIS: Simplify the use of geospatial web services in environmental modelling , Environmental Modelling & Software, Volume 119, September 2019, Pages 228-241

23. Daewook Lee, Joonho Kwon, Sangjun Lee, Seog Park, Bonghee Hong, Scalable and efficient web services composition based on a relational database , Journal of Systems and Software, Volume 84, Issue 12, December 2011, Pages 2139-2155

24. Hanine Tout, Azzam Mourad, Chamseddine Talhi, Hadi Otrok, AOMD approach for contextadaptable and conflict-free Web services composition, Computers & Electrical Engineering, Volume 44, May 2015, Pages 200-217

25. Xu Han, Binyang Li, Kam-Fai Wong, Zhongzhi Shi,Exploiting structural similarity of log files in fault diagnosis for Web service composition, CAAI Transactions on Intelligence Technology, Volume 1, Issue 1, January 2016, Pages 61-71.

26. Hongbing Wang, Shunshun Peng, Qi Yu, A parallel refined probabilistic approach for QoS-aware service composition, Future Generation Computer Systems, Volume 98, September 2019, Pages 609-626.

27. Mahdi Bashari, Ebrahim Bagheri, Weichang Du, Automated composition and optimization of services for variability-intensive domains, Journal of Systems and Software, Volume 146, December 2018, Pages 356-376.

28. Touraj Laleh, Joey Paquet, Serguei Mokhov, Yuhong Yan, Constraint verification failure recovery in web service composition, Future Generation Computer Systems, Volume 89, December 2018, Pages 387-401.

29. Xiaolong Xu, Hanzhong Rong, Ella Pereira, Marcello Trovati, Predatory Search-based Chaos Turbo Particle Swarm Optimisation (PS-CTPSO): A new particle swarm optimisation algorithm for Web service combination problems, Future Generation Computer Systems, Volume 89, December 2018, Pages 375-386.

30. Qiang Yu, Ling Chen, Bin Li, Ant colony optimization applied to web service compositions in cloud computing, Computers & Electrical Engineering, Volume 41, January 2015, Pages 18-27.

31. Sunaina Kotekar, Sowmya S. Kamath, Enhancing service discovery using cat swarm optimisation based web service clustering, Perspectives in Science, Volume 8, September 2016, Pages 715-717

32. Sabrine Boukh, Marouane KessentiniSalah BouktifHanzhang Wang, Ali Ouni, Improving web service interfaces modularity using multi-objective optimization, Automated Software Engineering, June 2019, Volume 26, Issue 2, pp 275-312

33. Liangzhao Zeng, Anne H. H. NguBoualem BenatallahRodion PodorozhnyHui Lei, Dynamic composition and optimization of Web services, Distributed and Parallel Databases, December 2018, Volume 24, Issue1-3, pp 45-72.

34. Afaf Mousa Jamal Bentahar, An Efficient QoS-aware Web Services Selection Using Social Spider Algorithm, Computer Science, Volume 94, 2016, Pages 176-182.

35. C.W.LiuX.X.Jin1L.S.Li, A web services-based multidisciplinary design optimization framework for complex engineering systems with uncertainties, Computers in Industry, Volume 65, Issue 4, May 2014, Pages 585-597