Web Service Composition Based on Quality of Service on the Internet of Things Using Bee Colony Optimization Algorithm

mihan hossennazhd gargari,Aliali.valiyan kharvanagh
Department of Computer Engineering, Aras Branch, Islamic Azad University, Aras, Iran
Email: mihan.hossennejad@gmail.com ,ali.valiyan73@gmail.com
Receive Date: 28 December 2024 Revise Date: 15 January 2025 Accept Date: 27 February 2025

Abstract

Today, the rapid development and growth of hardware, network technology, and various types of smart devices that can connect to the Internet and send and receive data have led to the emergence of a new technology called the Internet of Things (IoT). Using IoT technology, many objects in our environment are connected to the Internet and can be managed and controlled through applications available on smartphones and tablets. Service-oriented architecture has been successfully applied in various fields, including grid computing, cloud computing, wireless sensor networks, automotive networks, and the IoT. Service-oriented architecture is an architecture style that supports the loose coupling of services, enabling business flexibility and interoperability independent of technology. It consists of combining a set of business-based services. Since a single service may sometimes fail to meet user needs, industrial organizations prefer service composition to create more complex composite services. The problem of combining IoT services with respect to their Quality of Service (QoS) involves finding a set of candidate services with different non-functional characteristics that satisfy user-specified constraints and optimize an objective function. Thus, combining IoT services based on their QoS is classified as NP-Hard. This paper uses the bee colony optimization (BCO) algorithm to solve the problem of IoT service composition. The results of the simulation on the QWS dataset in the MATLAB 2019 environment show that the proposed method outperforms the genetic algorithm (GA) and the particle swarm optimization (PSO) algorithm.

Keywords: bee colony optimization algorithm, Internet of Things, service composition, service quality

1 Introduction

The advent of the Internet of Things (IoT) has significantly improved efficiency and productivity in smart environments, albeit with associated costs [1, 2]. The IoT is crucial in meeting user needs with a combination of services [3, 4]. When objects are interconnected, we can talk about an intelligent environment. The IoT is a natural extension of intelligent convergence that creates smart convergent environments by integrating numerous intelligent objects with diverse capabilities [5, 6].

The IoT and composite services have made trade and information exchange more dynamic than ever. The IoT offers new opportunities for developing software that integrates real-time operations in the industry [1]. While web services serve a similar function, industrial companies must consider user requirements, which has created composite services to bridge this gap. Quality of service (QoS) criteria vary with user needs, and combining services optimizes these criteria [7]. Users typically have two types of requirements: functional and non-functional. These are often formalized in a contract between the user

and the service provider, which legally binds the provider to meet the customer's needs. These user requirement constraints are specified during the implementation phase of composite services [8]. To achieve value and production activities with an extra return, it is necessary to consider the QoS composition [3].

Research has been extensive in the IoT field, covering middleware, routing [9], and data quality [10]. However, comparative studies on composite services in the IoT remain underexplored. Due to the IoT applications, devices in which the IoT is used are distributed across very dynamic environments. Where communications are unreliable, service interruptions may occur. Therefore, devices must be evaluated based on energy, communications, and computational capabilities. Security is an indispensable factor in improving the IoT. It encompasses four laws, i.e., the laws of perception, networking, support, and software, as well as other issues, e.g., management key, security strategy, and security regulations. In addition, trust management is vital in the IoT. A major challenge in service composition is selecting the most appropriate service instance from available options, which is an abstract function of service composition [11].

The IoT faces numerous challenges, but the challenge dealt with in this research is service composition. In the contemporary world, users are preoccupied with access to their target needs or services. When the user request is so that a single service cannot meet it, services are combined to optimize responses to user criteria [3, 13]. The main issue in service composition is selecting the best service among those that perform equivalent tasks but differ in quality criteria such as response time, cost, reliability, and system availability [1]. QoS and its paramount significance in service composition are the most crucial challenges in this field. Combining services is an NP-Hard problem, and using genetic algorithms for optimal service selection [14] is inefficient due to slow convergence and poor local search. Therefore, the bee colony optimization (BCO) algorithm has been used to address this issue.

This paper chiefly focuses on IoT service composition using the BCO algorithm. It is important how to use services. In some cases, it is necessary to combine services to meet complex user needs. Service composition aims to ensure reusability and interoperability by organizing existing services [8]. The choice of service composition is driven by the increasingly complex needs of modern societies, which are addressed through this approach. The main contributions of this paper are as follows:

- Solving the problem of combining IoT services using the BCO algorithm
- Evaluating the proposed method on the QWS dataset
- Examining the proposed method based on convergence and fitness function
- Comparing the proposed method with a genetic algorithm (GA) and the particle swarm optimization (PSO) algorithm

The paper is organized as follows. Section 2 reviews previous studies on quality of service using different models.

Section 3 explains the steps of the proposed method. Section 4 reports the results of evaluating the proposed method. Finally, Section 5 discusses the concluding points and lists suggestions for future work.

2 Literature Review

This section reviews previous studies on service composition in the IoT.

A QoS-aware service composition approach that exploits a novel bat algorithm (QC-NBA) has been proposed in [15]. Unlike many service composition approaches, the NBA method improved the mechanisms of exploring and exploiting the composition search space using quantum techniques. The QC-NBA algorithm was evaluated using the QWS dataset, which consists of 2507 services described by nine QoS attributes. The performance of the QC-NBA algorithm was enhanced in terms of execution time and composition quality. Simulation scenarios showed that the QC-NBA approach achieved good composition in terms of QoS application.

In [16], a method based on the ant colony optimization algorithm has been proposed for service composition. In this method, artificial ants collaborate to find the optimal solution. Pheromone information (chemicals that real ants leave behind when they move along a path) is assigned to the edges of a graph. Artificial ants in this algorithm travel through the graph, searching for suitable paths following pheromone and discovery information. The amount of pheromone on each edge evaporates by a certain amount in each iteration. The pheromone levels contained on the edges are also updated based on the paths traveled by the ants. Artificial ants are usually integrated with a list that stores their previous activities. They may perform additional operations (e.g., local search, crossover, and mutation) to improve the quality of the results. In this method, artificial ants move on a graph to create composite services. In this graph, each node represents an IoT hub, and edges connect these nodes. Each ant chooses its path based on pheromone and discovery information. The more the number of ants that pass through a path, the more the pheromone accumulates on that path, increasing the likelihood of its selection by subsequent ants. The pheromone degradation value ranges between zero and one. This method is applied iteratively to find suitable solutions. The ant colony algorithm has several advantages over the GA, including positive feedback and distributed computing. Another advantage is its efficiency in finding the right combination compared to other algorithms. However, a disadvantage is workload imbalance.

In [17], a method has been presented for service composition, where service centers are sorted in descending order based on the number of services they offer. This sorting helps quickly select the center with the most services first. The algorithm then checks which services from this center can help meet the user's needs. If no suitable service is found, it moves to the next center, checking its services without including it in the list of suitable centers. This continues until the user's needs are met. Once a suitable service center is found, it is added to the composition list. If all the user's needs are not met by the services in a center, the next center is selected. If two centers are similar in conditions, i.e., they have the same number of services, one is chosen

randomly. Typically, the center with the lower number is selected first, but if it is the second choice, the center with the highest user quality criterion is selected. The advantage of this method is that it finds the appropriate composition with the fewest centers needed to meet the user's needs. However, the disadvantage is that it does not distribute the workload evenly among the providers.

In [18], a method called cloud-based service composition has been introduced, where all possible solutions involving different cloud compositions are considered recursively. The algorithm examines all compositions to find a suitable solution. The idea is to first consider compositions with one cloud, then those with two clouds, and so on, until compositions that include all clouds in the multi-cloud environment are considered. This algorithm can find the appropriate cloud composition that includes the fewest clouds necessary to meet the user's request. However, the algorithm is in the worst condition in terms of complexity since it examines all possible compositions. For each c_i cloud, where i is a value between [0, n], there are c_i^n different cloud compositions. Each cloud has services, and these values are summed together.

In [1], a method based on a GA has been presented for combining services in the IoT. In this algorithm, when a user request is received, several resources are nominated for each task. From these resources, the one that best satisfies QoS criteria such as execution time, cost, reliability, and availability is selected. QoS criteria describe non-operational characteristics such as reliability and cost. The candidate resources for each request perform the same task but have different quality criteria. The main question for each composite request given by the user is to select a service that improves the user's knowledge and satisfies the user's requested QoS criteria. In this method, the algorithm [19] first generates an initial population and then iteratively tries to find the appropriate solution. In each iteration, the individual fitness value is calculated for each resource to check the efficiency of each solution, and a new population is generated through selection, crossover, and mutation operations. This process is repeated until the convergence criteria are met. The QoS criteria are divided into two parts: positive and negative. The response time and cost are in the negative part, and reliability and availability are in the positive part. All these four criteria are considered for each resource. Then, all the services are combined and encoded, and each gene in this list encodes a single service for each task. The initial population is randomly selected for the GA. In the selection operation, the probability that a resource with a certain fitness value is selected from the population is examined. In the crossover operation, some pairs of possible solutions, or resources categorized in different lists, are randomly selected and combined to reproduce new solutions. In the mutation operation, a resource from one list is selected and replaced with another resource from a different list. The GA is based on generating an initial population suitable for the hybrid model considered for this method. By imitating the principle of natural selection, less suitable roots have a lower chance of survival.

In [20], a clustering-based method called CL-ACO has been proposed. This algorithm is based on clustering along with

the ant colony algorithm. Hierarchical clustering (tree) is used for clustering services, which reduces time complexity. The main goal is to provide a set of individual services with the highest ability to combine, which allows for responding to complex user requests. Criteria considered include accuracy, selection of appropriate compositions, and reduction of response time. This algorithm consists of two parts. In the first part, called publication, the provider interprets the services using a semantic model with the OWL-S standard and publishes them in the semantic network. The generated services are then added to similar clusters simultaneously with their publication in the semantic network. The second part involves the client's actions, where the client first makes a request to the system, including defined inputs and outputs. Next, the user request along with the clusters at the root of the tree is compared using a matching algorithm. The clusters most similar to the user request are selected. The client request is then compared with the children of the selected cluster. This process continues until the leaves of the tree structure, representing the available services, are reached. In this method, the number of ants is equal to the number of services. The ants are placed on the nodes of the graph where the initial services are located. All the ants then start to traverse the graph and select the edge with the highest similarity degree and better conditions in terms of non-operational characteristics for traversal.

In [21], a particle swarm algorithm method has been proposed, combined with another hybrid optimization algorithm initially proposed as the Hungarian algorithm and later developed as the Munkres algorithm by Harold Kuhn in 1955 [22]. The Munkres algorithm is applied to 10% of the particles after each iteration of the particle swarm algorithm. One particle is randomly selected, and 10% of the workflow is also randomly selected to run the Munkres algorithm in this iteration. If an improvement is obtained after using the Munkres algorithm, those specific particles are updated with the workflow optimization, and the next iteration continues. However, the fitness value is only about 80% of the optimal value, so it can be used for services simultaneously. There is a balance between the fitness value and the runtime. The composite time complexity is a drawback of the Munkres algorithm, and premature convergence is a drawback of the particle swarm algorithm. By combining these two algorithms, a balance is achieved between these drawbacks.

In [5], the typical features of CMFG have been analyzed, and the implementation of the entire service composition life cycle has been discussed. Several key issues for service composition, such as modeling, evaluation, and selection optimization, have been studied in detail.

3 The Proposed Method

This section presents the proposed method in two parts. The first part defines the service composition problem in the IoT and provides relevant definitions. The second part discusses the BCO algorithm for service composition in the IoT. Figure 1 depicts a flowchart of the proposed method.

3.1 Problem definition

The problem of combining IoT services based on their QoS involves finding a set of candidate services with different performance characteristics. The goal is to satisfy the constraints specified by the user and optimize an objective function. This section asserts this problem. A sample IoT service composition based on their QoS can be expressed as follows. A service composition request is modeled as a workflow using a directed acyclic graph (DAG) as G = (V, E) in which V = {T_1, T_2, …, T_n} denotes the number of tasks (n) in the workflow and E represents the set of edges indicating task priority.

Each task (T_i) ($1 \leq i \leq n$) in the workflow has a set of candidate services $CS_i=\{CS_i^1,CS_i^2,...,CS_i^{m_i}\}$, where CS_i^j ($1 \leq j \leq m_i$) is a candidate IoT service. Parameter m_i represents the total number of candidate services available for task T_i. Each candidate service CS_i^j has a set of different QoS formations $QoS_i^j=\{Q_1,Q_2,...,Q_K\}$, where Q_l ($1 \leq l \leq K$) represents a QoS attribute of the IoT services. The QoS information related to the IoT services is stored in the QoS repository. Parameter K is also the number of QoS attributes related to the IoT services used in the QoS model. Parameter QC represents the set of global constraints specified by the user ($QC=(C_1,C_2,...,C_K\}$). Considering the above, the objective of the QoS-aware service composition problem is to find the near-optimal composite IoT service such that Eq. (1) holds [23].

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

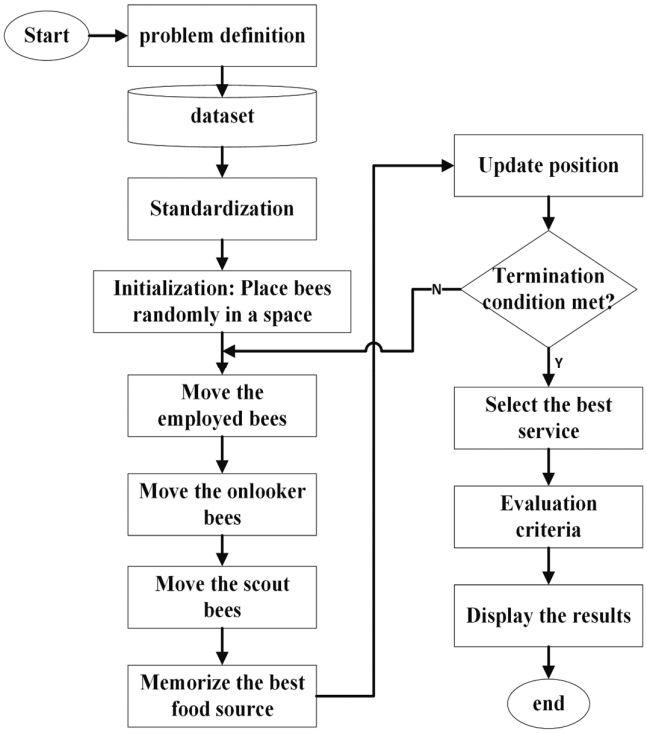

Fig.1. The flowchart of the proposed method

3.2 Dataset

The Quality of Web Service (QWS) dataset was used in the proposed method as the QoS parameter values [24]. The dataset consists of 2507 web services and their QoS measurements. The QoS parameter values [25, 26] of the services have been measured by their proposed Web Service Broker framework. Each record in the QoS dataset contains the values of eleven parameters for each web service. The first nine parameters are the QoS parameters measured by the Web Service Broker framework over a six-day period [27]. The QoS values in the dataset are the averages of the measurements performed during this period. Table 1 provides a simple description of the nine QoS parameters.

Table 1. The description of Quality of Service (QoS) parameters in the QWS database

Parameter	Description	Unit
Response time	The time required to send a request and receive a response	ms
Availability	Number of successful invocations per total invocations	%
Throughput	Total number of invocations for a certain time period	Invocations/s
Successability	Number of responses per number of request messages	%
Compliance	The extent of the compliance of the WSDL document with WSDL specifications	%
Best practices	The extent of the compliance of a service from the base WS-I profile	%
Latency	The time required by a service provider to process a request	ms
Documentation	Extent of documentation (descriptive labels) in WSDL	%

Considering that IoT services are an extension of web services in the IoT environment, the QWS dataset was used for the values of QoS parameters. By carefully describing the QoS parameters in Table 1, it can be seen that the values of the three parameters of compliance, best practices, and documentation are constant for multiple invocations of a service at runtime. Therefore, four of the above parameters were used.

3.3 Normalizing QoS parameters

Different QoS parameters of an IoT service are measured with different units. So, to calculate the objective function, all these parameters need to be measured on the same scale. Therefore, the values of all QoS parameters must be normalized on the same scale, which would allow us to have uniform measurements of the values. The general approach for this purpose is to normalize the values of all parameters in a range from zero to one. QoS parameters can be divided into two categories: maximization and minimization parameters. Maximization parameters are those whose values should be maximized, while minimization parameters are those whose values should be minimized. Eq. (2) and (3) show the normalization rules for maximization and minimization parameters, respectively [28].

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

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

where $CS.Q^i$ is the value of the ith QoS parameter of the candidate service CS, and $N_{CS.Q^i}$ is its

normalized value. Additionally, Q^i_{max} and Q^i_{min} are the maximum and minimum values of the ith parameter among all services.

3.4 Building initial population

In the BCO algorithm, bees are divided into three groups: worker bees, onlooker bees, and scout bees. Worker bees search for food around food sources using their memory. They also share their information about these sources with the onlooker bees. Onlooker bees tend to choose good food sources from those found by worker bees. A food source with better quality (fitness) has a higher chance of being selected by the onlooker bees. Scout bees also search for new food sources, filling the worker bees that have abandoned their own food sources.

In the BCO algorithm, half of the initial population consists of worker bees and the other half of onlooker bees. Therefore, in the proposed approach, the random solutions that are generated are as many as the number of population size (PS), half considered worker bees and the other half onlooker bees.

3.5 Bee movement

3.5.1 Worker and onlooker bees

In the first stage of the BCO algorithm, each worker bee finds a new food source, which essentially means creating a new solution. If X_i is the ith worker bee and V_i is the location of the new food source, V_i is obtained from Eq. (4).

$$V_{ik} = X_{ik} + \varphi_{ik} \times (X_{ik} - X_{jk}) \quad (4)$$

where X_j is a randomly selected solution ($i \neq j$) and k is a subscript randomly selected from the numbers 1 to N. Also, φ_{ik} is a uniformly distributed random number in the interval [-1, 1]. After a new food source – or, indeed, a solution – is created, a greedy search is used. If the fitness of V_i is better than the fitness of its parent (X_i), then X_i is replaced with V_i and updated; otherwise, X_i remains unchanged. After all worker bees have completed their search process, they share their updated food source information with the onlooker bees through a dance. An onlooker bee evaluates the nectar information collected from all worker bees and selects a food source with a probability related to its nectar content. The probabilistic selection at this stage is actually a roulette wheel selection, calculated using Eq. (5).

$$P_i = \frac{Fit_i}{\sum_j Fit_j} \quad (5)$$

where Fit_i is the fitness value of the ith solution in the population. As shown in Eq. (6), the food source with higher fitness has a higher probability of being selected by the onlooker bees.

3.5.2 Scout bees

If, after a predefined number of iterations (called the bound), some food sources (or solutions) do not improve in fitness, they are abandoned. If X_i is considered an abandoned food source, the scout bee finds a new food source and replaces X_i using Eq. (6).

$$X_{ik} = LB_j + rand(0,1) \times (UB_j - LB_j) \quad (6)$$

where rand(0,1) represents a random number with a normal distribution in the interval [0,1], and LB and UB denote the lower and upper bounds of the ith dimension, respectively.

3.6 Coding

In the initialization phase of the BCO algorithm, an initial population of random solutions must be created. In this algorithm, each bee represents a solution to the problem, and a solution here is the composite IoT service represented by an array of length n (the number of tasks in the workflow). The number stored at subscript i of the array represents the ID of the candidate service that will execute the task T_i. Considering that the number of solutions in the initial population will be p, the initial population of solutions will be a $p \times n$ matrix.

3.7 Fitness function

The main objectives of the QoS-aware IoT service composition problem are to satisfy the constraints specified by the user and to optimize a fitness function. The fitness function should optimize the values of the QoS parameters for the constructed composite IoT service. Given that four parameters – response time (Resp), availability (Avail), reliability (Reli), and latency (Late) – have been used in the proposed method, the fitness function for a solution is defined as Eq. (7).

$$\text{Fitness}(\text{Sol}) = w_1 * \text{Sol.Avail} + w_2 * \text{Sol.Reli} * w_3 * \text{Sol.Resp}^{-1} + w_4 * \text{Sol.Late}^{-1} \quad (7)$$
$$w_1 + w_2 + w_3 + w_4 = 1$$

where the coefficients w_1, w_2, w_3, and w_4 are positive weights indicating the importance of each QoS parameter and specified by the user.

4 Evaluation and Results

This section reports the results of the simulation and evaluation of the proposed method using the BCO algorithm. It is described below how the proposed method was simulated and how various experiments were conducted to evaluate it. All experiments were performed on a Dell computer with a 2.0 GHz Core i7 processor and 4 GB of main memory. The QWS dataset was also used as the QoS data on candidate IoT services. The proposed method was simulated and evaluated using MATLAB software. Since the proposed method uses a meta-heuristic algorithm, the results were evaluated in terms of convergence and stability. Additionally, the results were compared with those of the GA and PSO algorithm. Table 2 shows the parameters related to the GA, PSO algorithm, and the proposed method.

Table 2. Initialization of the parameters

Algorithm	Parameter	Value
Genetic algorithm	Initial population size	100
	Crossover operator	Binary
	Selection operator	Roulette wheel
	Crossover rate	0.8
	Mutation rate	0.05
Particle swarm optimization algorithm	Initial particle number	100
	C_1	1
	C_2	2
Artificial bee colony optimization algorithm	Number of initial bees	100
	Number of worker bees	50
	Number of onlooker bees	50

4.1 Testing QoS criteria

This test was designed to evaluate the quality of the created composite IoT service. For each QoS criterion, a service request with 20 tasks and a maximum of 100 candidate services was tested 10 times for all three algorithms. The average results are reported in Table 3. Furthermore, Figures 2-5 depict the bar graphs for each criterion. The results indicate that the composite service generated by the BCO algorithm has better quality in terms of all QoS criteria than the GA and PSO algorithm.

Table 3. The results of the simulation of QoS criteria

Number of tasks	Algorithm	Availability	Reliability	Latency	Response time
20	GA	0.8724	0.7961	0.0652	0.0681
	PSO	0.8873	0.8471	0.0584	0.0592
	BCO	0.9217	0.9627	0.0472	0.0484

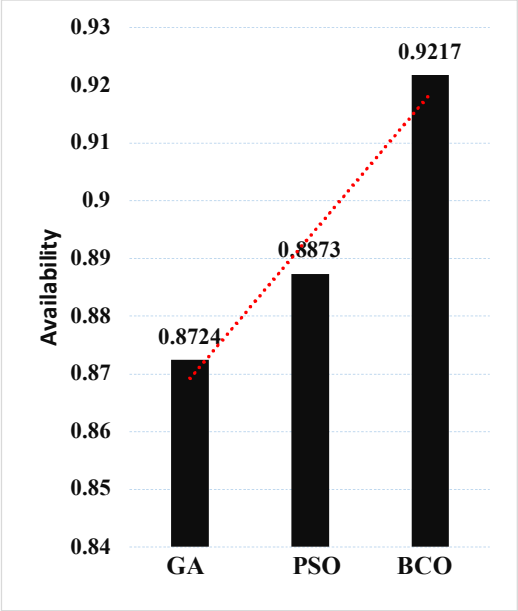

Fig.2. The availability of the generated composite service

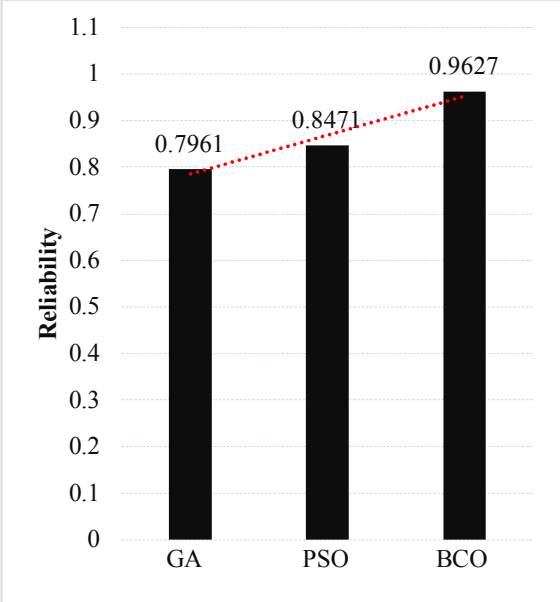

Fig.3. The reliability of the generated composite service

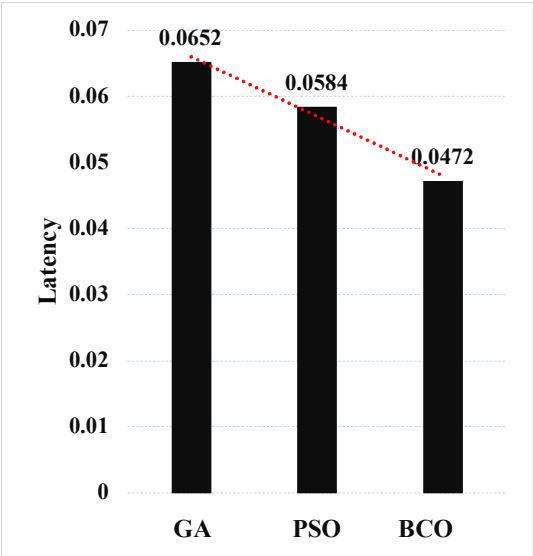

Fig. 4. The latency of the generated composite service

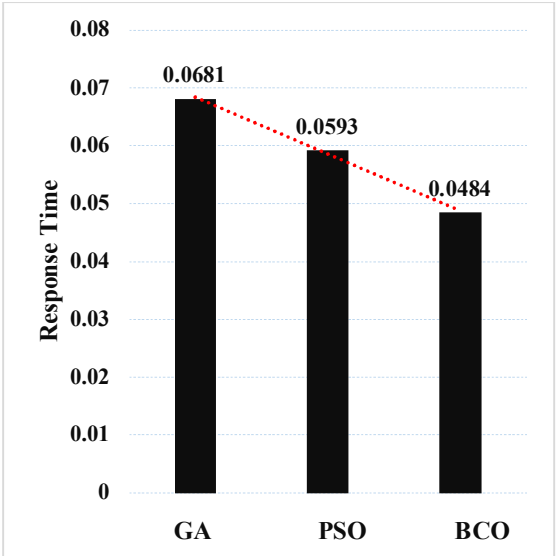

Fig.5. The response time of the generated composite service

4.2 Testing convergence

For the convergence test, the proposed method, the GA, and the PSO algorithm were run for three composition requests consisting of 10, 20, and 50 tasks. Figures 6 to 8 show the convergence to the final solution. In these graphs, the horizontal axis represents the algorithm iteration order, and the vertical axis represents the best fitness value of each iteration.

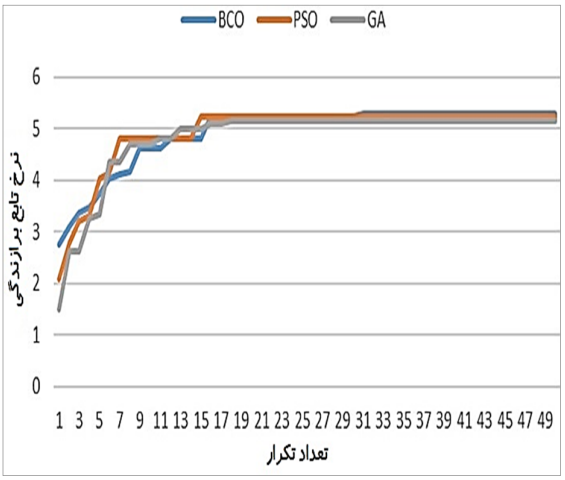

Fig.6. The convergence of the proposed method compared to the genetic algorithm and the particle swarm optimization algorithm (Number of tasks: 10)

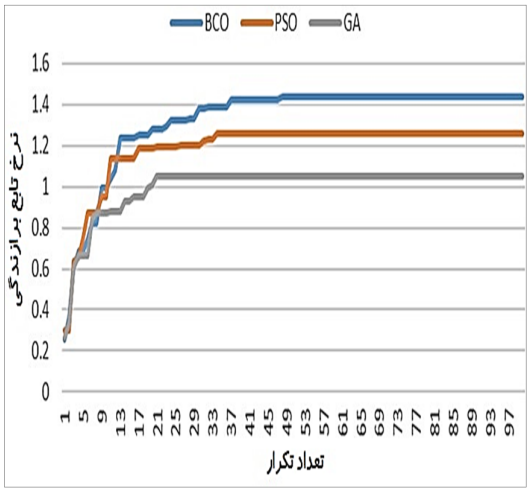

Fig. 7. Convergence of the convergence of the proposed method compared to the genetic algorithm and the particle swarm optimization algorithm (Number of tasks: 20)

The results reveal that the proposed method for combining IoT services has a high degree of convergence and finds a near-optimal composite IoT service. Additionally, the results indicate that the quality of the composite service created by the proposed method is better than that of the PSO algorithm and GA.

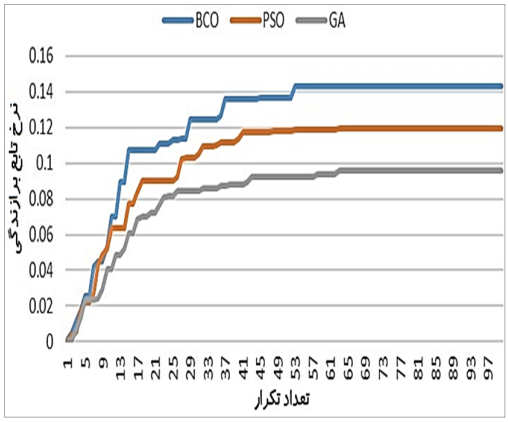

Fig. 8. The convergence of the proposed method compared to the genetic algorithm and the particle swarm optimization algorithm (Number of tasks: 50)

4.3 Testing stability

Another important test for heuristic algorithms is to examine their stability. Since heuristic algorithms, including the BCO algorithm, have a random and non-deterministic nature, it is essential to assess their stability. The stability of an algorithm refers to whether it produces the same or similar responses for different executions. To test the stability of the proposed algorithm, it was executed 10 times for three different composition requests with 10, 20, and 50 tasks. Figures 9-11 display the fitness value of the IoT composite service created in each execution. The horizontal axis indicates the order of the algorithm execution, and the vertical axis indicates the fitness value of the IoT composite service created in each execution.

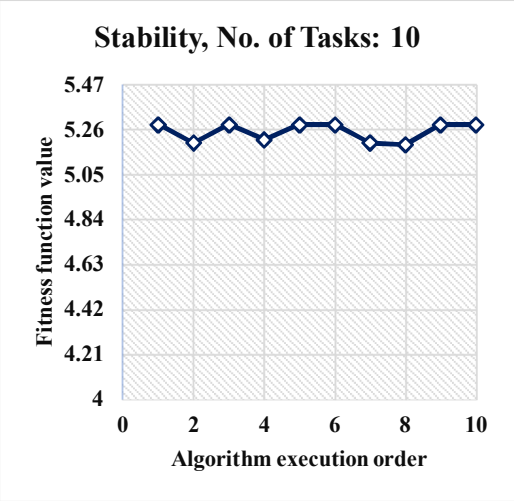

Fig.9. The stability of the proposed method (Number of tasks: 10)

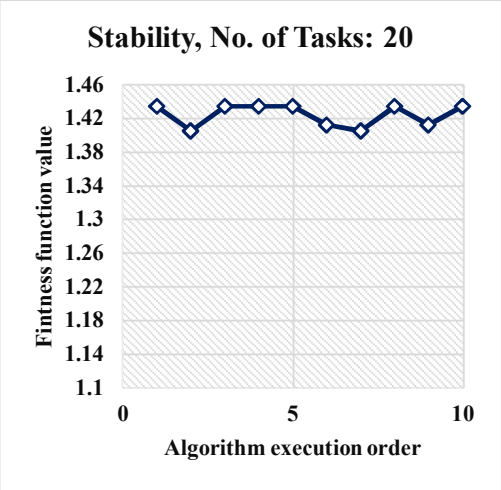

Fig.10. The stability of the proposed method (Number of tasks: 20)

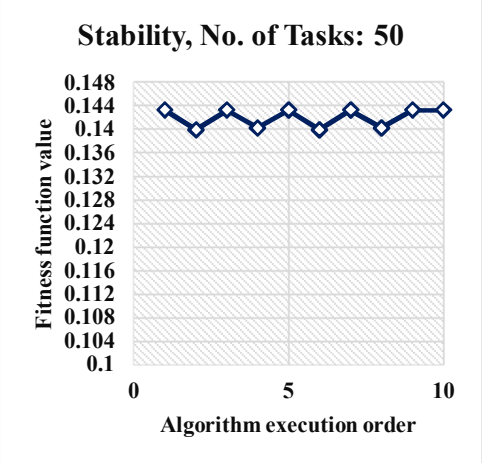

Fig.11. The stability of the proposed method (Number of tasks: 50)

A review of the stability graphs shows that the proposed method is, in general, highly stable, which is reflected in the lack of fluctuation in the fitness value of the near-optimal IoT composite service across different executions of the algorithm. However, the results also show that the algorithm is much more stable for composition requests with a smaller number of tasks than those with a larger number of tasks.

5 Conclusion and Future Works

This paper presented a BCO algorithm to address the problem of combining IoT services while considering their QoS. The proposed approach was simulated in the MATLAB software environment alongside two other algorithms: GA and PSO algorithm. We examined and compared their efficiency in terms of convergence and stability. The results from various test scenarios demonstrated that the proposed approach had a good convergence rate and was highly stable. Additionally, the latency of the proposed method was found to be lower than that of the other two algorithms. There are numerous other metaheuristic approaches that can be effective in combining and selecting IoT services. Therefore, researchers in their future works are recommended to explore other metaheuristic approaches to the problem of combining IoT services and compare them with the approach presented in this paper.

References

[1] [1] Q. Li, R. Dou, F. Chen, G. Nan, "A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of things," International Journal of Computational Intelligence Systems. Vol. 7, No. 2, pp. 26-34, 2014.

[2] [2] Y. Xu, W. Xiao, X. Yang, R. Li, Y. Yin, Z. Jiang, "Towards effective semantic annotation for mobile and edge services for Internet-of-Things ecosystems," Future Generation Computer Systems. Vol. 139, No. 1, pp. 64-73, 2023.

[3] [3] A. Saied, N. Nima Jafari, "Service Composition Mechanisms in the Multi-Cloud Environments: A Survey," International journal of new computer architectures and their applications. Vol. 6, No. 2, pp. 40-48, 2016.

[4] [4] F. Seghir, G. Khababa, "An improved discrete flower pollination algorithm for fuzzy QoS-aware IoT services composition based on skyline operator," The Journal of Supercomputing. Vol. No. pp. 2023.

[5] [5] J. Liu, S. Xu, F. Zhang, L. Wang, "A hybrid Genetic-Ant Colony Optimization Algorithm for the Optimal Path Selection," Intelligent Automation & Soft Computing. Vol. 23, No. 2, pp. 235-242, 2017.

[6] [6] M. A. khelili, S. slatnia, O. kazar, A. merizig, S. mirjalili, "Deep learning and metaheuristics application in internet of things: A literature review," Microprocessors and Microsystems. Vol. 98, No. 2, pp. 104792, 2023.

[7] [7] A. Mousa, J. Bentahar, "An Efficient QoS-aware Web Services Selection Using Social Spider Algorithm," Procedia Computer Science. Vol. 94, No. 1, pp. 176-182, 2016.

[8] [8] Y. Zhang, D. Xi, H. Yang, F. Tao, Z. Wang, "Cloud manufacturing based service encapsulation and optimal configuration method for injection molding machine," Journal of Intelligent Manufacturing. Vol. 30, No. 7, pp. 2681-2699, 2019.

[9] [9] J. Jiang, G. Han, C. Lin, "A survey on opportunistic routing protocols in the Internet of Underwater Things," Computer Networks. Vol. 225, No. 1, pp. 109658, 2023.

[10] [10] S. Nayak, N. Ahmed, S. Misra, "Deep Learning-Based Reliable Routing Attack Detection Mechanism for Industrial Internet of Things," Ad Hoc Networks. Vol. 123, No. 2, pp. 102661, 2021.

[11] [11] G. Sambasivam, J. Amudhavel, T. Vengattaraman, P. Dhavachelvan, "An QoS based multifaceted matchmaking framework for web services discovery," Future Computing

and Informatics Journal. Vol. 3, No. 2, pp. 371-383, 2018.

[12] [12] B. Andrei, B. A. Elena, "Quality Control in Logistics Activities through Internet of Things Technology," Scientific Bulletin of Naval Academy. Vol. 19, No. 1, pp. 27-30, 2016.

[13] [13] L.-l. Shi, L. Liu, L. Jiang, R. Zhu, J. Panneerselvam, "QoS prediction for smart service management and recommendation based on the location of mobile users," Neurocomputing. Vol. 471, No. 1, pp. 12-20, 2022.

[14] [14] P. Asghari, A. M. Rahmani, H. H. S. Javadi, "Privacy-aware cloud service composition based on QoS optimization in Internet of Things," Journal of Ambient Intelligence and Humanized Computing. Vol. 13, No. 11, pp. 5295-5320, 2022.

[15] [15] A. Kouicem, M. E. Khanouche, A. Tari, "Novel bat algorithm for QoS-aware services composition in large scale internet of things," Cluster Computing. Vol. 25, No. 5, pp. 3683-3697, 2022.

[16] [16] Q. Yu, L. Chen, B. Li, "Ant colony optimization applied to web service compositions in cloud computing," Computers & Electrical Engineering. Vol. 41, No. 2, pp. 18-27, 2015.

[17] [17] H. Kurdi, A. Al-Anazi, C. Campbell, A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition," Computers & Electrical Engineering. Vol. 42, No. 2, pp. 107-113, 2015.

[18] [18] Z. Guobing, C. Yixin, X. Yang, H. Ruoyun, X. You, AI Planning and Combinatorial Optimization for Web Service Composition in Cloud Computing, in A nnu al Internatio nal Co nf erence o n Clo u d Co m p u t ing a nd Virtu a lizatio n (CCV 2010). 2010. p. 1-8.

[19] [19] D. Wang, Y. Yang, Z. Mi, "A genetic-based approach to web service composition in geo-distributed cloud environment," Computers & Electrical Engineering. Vol. 43, No. 1, pp. 129-141, 2015.

[20] [20] R. Narges Hesami, K. Esmaeil, J. Mehrdad, "An Optimized Semantic Web Service Composition Method Based on Clustering and Ant Colony Algorithm," ArXiv. Vol. abs/1402.2271, No. pp. 2014.

[21] [21] S. A. Ludwig. Applying Particle Swarm Optimization to Quality-of-Service-Driven Web Service Composition. in 2012 IEEE 26th International Conference on Advanced Information Networking and Applications. 2012.

[22] [22] H. W. Kuhn, "The Hungarian method for the assignment problem," Naval Research Logistics Quarterly. Vol. 2, No. 1-2, pp. 83-97, 1955.

[23] [23] J. Zhou, X. Yao, "A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition," The International Journal of Advanced Manufacturing Technology. Vol. 88, No. 9, pp. 3371-3387, 2017.

[24] [24] E. Al-Masri, Q. H. Mahmoud, Investigating Web Services on the World Wide Web, in Proceedings of the 17th International Conference on World Wide Web. 2008, Association for Computing Machinery. p. 795–804 , numpages = 10.

[25] [25] E. Al-Masri, Q. H. Mahmoud. QoS-based Discovery and Ranking of Web Services. in 2007 16th International Conference on Computer Communications and Networks. 2007.

[26] [26] E. Al-Masri, Q. H. Mahmoud, Discovering the Best Web Service, in Proceedings of the 16th International Conference on World Wide Web. 2007, Association for Computing Machinery. p. 1257–1258 , numpages = 2.

[27] [27] M. S. Das, A. Govardhan, D. V. Lakshmi. A classification approach for web and cloud based applications. in 2016 International Conference on Engineering & MIS (ICEMIS). 2016.

[28] [28] R. Boucetti, O. Hioual, S. M. Hemam, "An approach based on genetic algorithms and neural networks for QoS-aware IoT services composition," Journal of King Saud University - Computer and Information Sciences. Vol. 34, No. 8, Part B, pp. 5619-5632, 2022.