3-26-2022

Dynamic QoS/QoE-aware reliable service composition framework for edge intelligence

Vahideh Hayyolalam
Koç University

Safa Otoum
Zayed University, safa.otoum@zu.ac.ae

Öznur Özkasap
Koç University

Cluster Computing
https://doi.org/10.1007/s10586-022-03572-9

Dynamic QoS/QoE-aware reliable service composition framework for edge intelligence

Vahideh Hayyolalam[1] · **Safa Otoum**[2] · **Öznur Özkasap**[1]

Received: 7 May 2021 / Revised: 6 February 2022 / Accepted: 16 February 2022

Abstract
Edge intelligence has become popular recently since it brings smartness and copes with some shortcomings of conventional technologies such as cloud computing, Internet of Things (IoT), and centralized AI adoptions. However, although utilizing edge intelligence contributes to providing smart systems such as automated driving systems, smart cities, and connected healthcare systems, it is not free from limitations. There exist various challenges in integrating AI and edge computing, one of which is addressed in this paper. Our main focus is to handle the adoption of AI methods on resource-constrained edge devices. In this regard, we introduce the concept of Edge devices as a Service (EdaaS) and propose a quality of service (QoS) and quality of experience (QoE)-aware dynamic and reliable framework for AI subtasks composition. The proposed framework is evaluated utilizing three well-known meta-heuristics in terms of various metrics for a connected healthcare application scenario. The experimental results confirm the applicability of the proposed framework. Moreover, the results reveal that black widow optimization (BWO) can handle the issue more efficiently compared to particle swarm optimization (PSO) and simulated annealing (SA). The overall efficiency of BWO over PSO is 95%, and BWO outperforms SA with 100% efficiency. It means that BWO prevails SA and PSO in all and 95% of the experiments, respectively.

Keywords Artificial intelligence · Connected healthcare · COVID 19 · Fault prevention · Meta-heuristics · IoT

1 Introduction

Due to the extensive application of the Internet of Things (IoT) and edge technology [1], an enormous number of high-tech devices are connected to edge-assisted IoT networks to satisfy different requirements of today's technological life [2]. Besides, artificial intelligence (AI) paves the way of facilitating the use of these technologies. AI enhances the edge-assisted IoT systems by adding smartness and automation to them. The integration of AI and edge computing introduces a novel concept, edge intelligence [3, 4], which sets the scene for the novel generation of technologies alongside highly compelling applications such as autonomous car driving systems, smart devices tracking, real-time critical systems, predictive and real-time healthcare systems [5].

✉ Safa Otoum
safa.otoum@zu.ac.ae

Vahideh Hayyolalam
vhayyolalam20@ku.edu.tr

Öznur Özkasap
oozkasap@ku.edu.tr

1 Department of Computer Engineering, Koç University, Istanbul, Turkey

2 College of Technological Innovation (CTI), Zayed University, Abu Dhabi, United Arab Emirates

1.1 Motivation

In today's technological era, IoT devices produce a vast amount of data that are used as fuel to AI methods in producing potential solutions for IoT applications. Thus, AI has an important role in smart IoT applications [6]. Since deploying AI needs apparatuses strong enough in terms of processing and storage, most popular AI-assisted data analysis methods are utilized on cloud infrastructures that use central servers to process the aggregated data of users [7]. AI features are condensed by the cloud and are provided as cloud-oriented AI services (e.g., Google Cloud

AI). The methodology of utilizing these kinds of services is to transmit IoT data accompanied by other factors to the cloud servers. Then, AI functionalities are performed via a cloud server to produce the results, which will be sent back to the IoT devices. As a matter of fact, during this type of process, AI functionalities are abstracted as a service to IoT devices to accomplish the smart IoT applications [6]. Although central cloud server infrastructure has numerous merits, this type of structural design is susceptible to some limitations such as data privacy, communication cost, long response time, high latency, and single point of failure [7].

To cope with the mentioned limitations, edge computing [8] is the best choice since it pushes the computation including cloud-oriented data storage services and AI services to the network edge on edge devices [9]. Considering the fact that AI services can be divided into subtasks and IoT data can be fragmented into pieces, in case of replacing a central cloud server with edge computing, each of these subsets of AI service or pieces of IoT data can be deployed in a distinct edge device. Thus, an edge device abstracts either an AI subtask of a whole AI method/service or store a small piece of IoT data to provide and deliver a service. Transferring services from the cloud to the edge devices introduces a new edge-assisted services environment for connected IoT applications in which AI and data storage services are provided close to service consumers on IoT end devices [6]. In this environment, edge devices play the role of services.

1.2 Challenges

Distributing AI subtasks belonging to a single AI service and IoT data among edge devices can bring out several challenges some of which are discussed as follows. *Firstly*, the distributed AI subtasks need to be integrated and delivered as a single output to the AI service requester. Thus, there should be a smart composition method to integrate the AI subtasks. However, this composition process is a challenging issue and with increasing the number of involved edge devices it becomes an NP-Hard problem and cannot be solved via deterministic methods [10]. Thus, non-deterministic methods, such as meta-heuristic algorithms are utilized for solving these kind of problems, via which the near optimal solutions can be achieved, since there is no guarantee for obtaining exact optimal solution for NP-Hard problems. Mentioning a few points can elaborate on clarifying this challenging matter. Since each edge device can merely accommodate a single AI subtask, for distributing an AI service the system should intelligently find a set of edge devices to fulfill the requirements of performing the corresponding AI service. Besides, there is a possibility of abstracting the same AI subtask via different edge devices with different quality of service (QoS) and quality of experience (QoE) values. In other words, various edge devices can have the same functionality with different non-functional features such as QoS and QoE. Therefore, there should be a smart approach to select a proper edge device for each AI subtask to obtain an optimal solution among all possible solutions. *Secondly*, with regards to the mobility and heterogeneity of edge devices, availability and reliability are two vital characteristics of choosing them [11]. Therefore, this paper has a special focus on these two QoS parameters. Furthermore, QoE of the service consumers is a crucial criterion of evaluation, which is mostly neglected by researchers in the scope of edge service composition. *As a final challenging point*, monitoring the system for fault tolerance is critical for real-time smart IoT applications, which is addressed in this paper as well.

The proposed framework comprises main parts, including *IoT devices*, *edge devices*, *cloud*, *service composer*, *QoS and QoE monitoring*, and *fault controller*. The IoT devices are responsible for collecting vital signs and transferring them to the closest edge devices. Cloud distributes AI subtasks among edge devices (the method of decomposing an AI service and distributing its components is out of the scope of this paper), and offloads service composer, QoS and QoE monitoring, and fault controller to the edge computing apparatuses. Edge devices which are defined as a service in this scenario, regarding their functionality and QoS and QoE values can be selected for performing a part of the whole procedure. The service composer component manages the AI subtasks composition with respect to QoS and QoE values of selected edge devices as the services for performing those subtasks. The QoS and QoE monitoring module should monitor and update the QoS and QoE values for each service (edge device) since they can be changed in each invocation (selection and usage) of the corresponding service. This component can help the overall framework to provide a dynamic service composition solution. The fault controller component is in charge of handling possible faults occurring during the composition process by controlling each of the selected composition components (edge devices). Detailed information are provided in Sect. 3. Figure 1 illustrates the overview of the proposed framework.

1.3 Contributions

To the best of our knowledge, this is the first study that discusses the edge intelligence service composition in the connected healthcare domain. The main objective of this paper is to facilitate the deployment of AI on the resource constrained edge devices.

The main contributions of this research can be listed as follows:

- *Proposing a reliable and dynamic framework for service composition for edge intelligence* in order to handle the stated challenges, we propose a dynamic and reliable framework for service composition, particularly for AI subtasks composition in the scope of remote healthcare systems.
- *Proposing edge intelligence service composition for the connected healthcare application* with respect to the fact that AI tasks can be decomposed into subtasks, we propose a method for integrating the subtasks to provide a single composite result as if there has not been any division taken place.
- *Introducing the term Edge device as a Service (EdaaS)* in the proposed framework, we consider EdaaS in the network, each of which can host a subtask of the system's main AI task. Each edge device has a particular functionality and a set of non-functional features. Thus, AI subtasks can be deployed into edge devices with relevant functionality. We assume that all the involved edge devices and users (service requesters) are settled in the same network.
- *Considering QoE as a QoS parameter for each service (edge device)* we have taken QoS and QoE into account and correlate them using a linear correlation. Then, we utilize QoE values as a QoS parameter in our proposed framework. Although we have adopted a real-world QoS dataset for our model's initialization, we have defined a QoS and QoE monitoring module to update the values during the composition process to model a dynamic service composition framework.
- *Adopting meta-heuristics to compose the distributed AI subtasks* for solving the service composition problem, which is an NP-Hard problem, we adopted meta-heuristics since they are suitable for solving these kind of issues in a reasonable time duration.
- *Delivering reliable composite services* as another contribution, our framework has the capability of providing reliable composite service by preventing possible faults in terms of QoS constraints.

The rest of the paper is organized as follows. Section 2 discusses the relevant state-of-the-art research works. Section 3 describes the proposed framework in detail. The experimental setup and results are provided in Sect. 5. Finally, Sect. 6 outlines the conclusions.

2 Related work

This section briefly reviews the existing works in the scope of this paper, which consists of three subsections. Section 2.1 investigates the current edge service composition works. Section 2.2 inspects the state-of-the-art works relevant to fault-tolerance on service composition since our proposed framework has a special focus on reliability and fault tolerance. The summary and comparison is provided in Sect. 2.3. Moreover, Table 1 demonstrates the side by side comparison of the investigated papers.

2.1 Edge service composition

Researchers in [12] have introduced a blockchain-based decentralized solution for service composition in the scope of complex multimedia service delivery to cloud subscribers. For authenticating and delivering the obtained composite service, the proposed method dynamically produces user-defined services needless of any transitional services or network provider units. This research outlines a scalable, flexible, secure, and reliable decentralized cloud solution that adopts software defined network (SDN), blockchain, and fog computing paradigms to integrate the existing services and deliver a composite and complex service at the edge of the network. To this end, the authors have fragmented the services into sub-services and utilized reinforcement learning for constructing appropriate composition paths with respect to the network configurations to deliver multimedia services. The proposed method achieves a high service delivery success rate and reduces power consumption and resource usage.

As another research, in [13] authors have proposed a context-aware and real-time collaborative framework lied at the network edge, including end-user devices and mobile edge cloud (MEC), aiming to achieve a swift composite service delivery system. In the suggested solution, they have decomposed cloud data into a set of services and files, then replicate them to different MEC nodes. Regularly invoked services/files are cached onto end-users' devices to accelerate accessing them. All the nodes in the system, either MEC or mobile users, push services into the collaborative user/edge room, from where services are delivered based on the users' demand. Authors have utilized a learning-based workflow-net method depending on the former composition outcomes for forming service composition models that can be adopted in upcoming compositions. The envisioned solution guarantees the delivery of composite services to the service requester with a short response time. Moreover, the method satisfies QoS demands and provides reasonable load balancing amongst the mobile and edge nodes.

In order to simplify the deployment of AI tasks in the edge environment, authors in [6] have designed an AI task composition framework with a focus on privacy. Their assumption is that complicated AI tasks can be decomposed into smaller AI subtasks and deployed on edge devices via task offloading and distribution techniques. The proposed method adopts the Skyline optimization method to provide an intelligent service selection mechanism. Moreover, the proposed framework employs an entirely homomorphic encryption-based privacy-preserving method to provide a privacy-aware service framework for edge computing. Besides, a Map-Reduce model has been used for performing a privacy-preserving service composition model on an edge environment. This research obtains short composition time and improved privacy.

Furthermore, authors in [14] have proposed a simulation-based optimization method for service composition with the goal of providing reliability. They have considered the whole system as two layers, including edge and cloud layers, and discussed both sides. They have utilized a stochastic Petri net model for formulating both layers and designed a model aggregation technique for service composition issues. Aiming to enhance model solving effectiveness, they have adopted a time scale decomposition technique for handling the state explosion issue in complex service processing and large-scale systems. Moreover, the authors have proposed simulation schemes to optimize and assess the performance of the system, and used an ordinary optimization technique to diminish the search space size. The proposed system achieves high reliability and short response time in both edge and cloud layers.

Since most of the conventional dynamic reconfiguration service composition methods have concentrated on service scheduling to deliver a composite service with normal operation, they do not have the capability of timely responding to the dynamic environmental changes. Thus, authors in [15] have proposed a dynamic reconfiguration for service workflow in mobile edge e-commerce environments to address the mentioned challenge. The service value and cost parameters are considered for validating the proposed model. They have defined service value as stability evaluation of the service, and the cost parameter refers to the service invocation cost. For predicting the services' stability, a long short-term memory (LSTM) neural network has been utilized. Afterwards, considering the confined available resources, the authors have adopted a method to compute the service invocation cost. Eventually, the system selects proper candidate services with regards to both service cost and stability. The proposed model obtains high stability, low energy usage, and high accuracy in service prediction.

Authors in [16] have concentrated on security and energy efficiency to propose a new hybrid meta-heuristic-based method for QoS-aware edge service discovery and selection in IoT. The proposed method adopts multi-objective Grey Wolf Optimizer and a Genetic Algorithm (GA). The proposed hybrid method effectively improves energy consumption, response time, and cost of service discovery and selection in IoT.

2.2 Fault tolerance on conventional service composition scenarios

Authors in [17] have designed a sensor-based and repair-oriented algorithm to obtain reliability during the process, at the component level, before a system crash happens. Aiming to detect the faulty section on a workflow, they have utilized a novel sensor strategy according to project management rules. The repaired algorithm, including a single service and multiple service configuration, has been applied, enabling the system to search for finding a novel adjacent solution systematically. In multi-service reconfiguration, for swiftly recovering the faulty section with no crashing on the system, the authors have mentioned a novel

Table 1 Related work comparison table

Ref	Achievements	EC	FT	QoS	QoE
[12]	High service delivery success rate, reduces power consumption and resource usage	✓	✗	✓	✓
[13]	Short response time, satisfies QoS demands, addressing load balancing	✓	✗	✓	✗
[6]	Short composition time and high privacy	✓	✗	✓	✗
[14]	High reliability and short response time	✓	✗	✓	✗
[15]	High stability, low energy usage, and high accuracy in service prediction	✓	✗	✗	✗
[16]	Low energy consumption, response time, and cost of service discovery and selection	✓	✗	✓	✗
[17]	High reliability, short response time	✗	✓	✓	✗
[18]	High reliability, short latency	✗	✓	✓	✗
[19]	High reliability, reduce the number of service rollbacks in a time-efficient manner	✗	✓	✓	✗
[20]	Improves success rate and overall reliability enhancement	✗	✓	✓	✗
[21]	Improves success rate and decrease the computational time	✗	✓	✓	✗
[22]	Improves reliability, decrease response time	✗	✓	✓	✓

EC edge computing, *FT* fault tolerance multi-level valued constraint satisfaction problem with the harmony search meta-heuristic algorithm. They have used a replacement strategy for conducting a fault tolerance system. Their method is able to recover the faulty section without user integration. Besides, with regards to the decomposition, the process is done in reasonable time duration.

Moreover, researchers in [18] have presented a self-healing model for web service composition, enabling the system to automatically discover and heal the composite web service failure without user intervention and interrupting the web service composition process. The proposed model has adopted the integration of Q-learning-based parallel GA and k-means clustering to optimize the service composition process. In order to cope with the failure, they have used a replacement strategy by replacing the faulty component with another equivalent service from the set of candidates. Aiming to facilitate finding the best possible solution, the authors have used the k-means clustering algorithm to reduce the web services in the search space. Their proposed model can dynamically replace the faulty component in an acceptable time.

The failure recovery of a composite service is indicated as the rollbacking of particular service transactions through the recovery process of the composition procedure. Due to the failure of constraint verification, most of the research works did not consider the service rollback minimization in their failure recovery methods for composite services. In this regard, authors in [19] have outlined a constraint-aware failure recovery method for failure prediction within a composite service with the aim of reducing the number of service rollbacks for the failures causing by constraint verification. The suggested model has adopted a planning-based algorithm along with a new processing technique for the constraint to predict and recover service failure. The proposed method can reduce the number of service rollbacks in a time-efficient manner.

Researchers in [20] have designed a reliability assessment technique targeting component cloud service (CCS) according to the failure probability through constant user-side invocation assessments. They proposed a perturbation-aware reliability sensitivity evaluation for service selection to deliver a reliable composite service. Their proposed method first examines the negative perturbations in the historical reliability time series of the CCS. Afterward, investigating the impact of CCS reliability perturbations on the reliability of the whole cloud system, the proposed method computes the reliability sensitivity of CCS. The first-order Markov Chain rule is adopted to illustrate the development regularity of the updated reliability time series of CCS. This research obtains an acceptable success rate and overall reliability enhancement in cloud service selection.

As another research, authors in [21] have proposed a new deadline-constrained and reliability-aware method for mobile service composition in opportunistic network environments, where users have the permission of combining and exploiting the resources of the adjacent devices via device-to-device communications. This research utilizes a method based on the Krill-Herd algorithm for deciding on composition schedules in a real-time manner with the aim of maximizing reliability with a certain deadline. Due to runtime mobility of services, the time-changing availability has been considered rather than presuming a fully available mobile services environment. The proposed method attained a good success rate and short computational time.

The authors in [22] have proposed a fault-tolerant architecture for service composition in a smart cities environment adopting fog computing technology. The proposed method is able to handle service composition problem via RESTful IoT technology. Besides, the designed method effectively can enhance scalability and reliability. The authors have adopted particle swarm optimization (PSO), GA, and artificial bee colony for solving the fault tolerant service composition strategy. The experimental results show that the proposed method can solve the problem with a short response time.

2.3 Summary and comparison

According to the investigated papers above, several approaches have been done to solve service composition problem in edge computing. However, some points have not been addressed by the current works, including:

- Existing works mostly have neglected QoE evaluation.
- AI subtask composition as one of the vital issues in the edge intelligence domain has been addressed by only a few papers like [6].
- The application of AI subtask composition for connected smart healthcare has not been performed by now.
- Although there are various cutting-edge studies dedicated to fault tolerance/prevention and reliability, they are not in the scope of edge intelligence.
- There is no prior work devoted to AI subtask composition in edge intelligence with the special focus on reliability and QoE evaluation.

Therefore, these limitations motivated us to perform a new research work by addressing and solving these limitations. As Table 1 illustrates, some papers address edge computing and neglect fault tolerance and vice versa. Also, most of the investigated papers ignore QoE evaluation. Our proposed work highlights both QoS and QoE and solves service composition in an edge computing environment with a special focus on fault tolerance.

3 Proposed framework

In this section, first, the basic concepts will be discussed in Sect. 3.1. The discussion will be followed by an example scenario. Then, the proposed service composition framework for edge intelligence with the aim of composing AI sub-tasks will be addressed in more details. The utilized notations are stated in Table 2.

3.1 Preliminaries

With the growth of technology, service consumers' demands are becoming more complex [23]. Thus, since a single service cannot satisfy them, the system needs to employ several different services to accomplish a request. To this end, the given service request, which is defined as a task, first should be decomposed into some subtasks. Then, via a service discovery method, the system tries to find relevant available services for each subtask. Afterward, adopting a service selection technique, the system will choose the appropriate services for performing each subtask. The selected services should be composed via a proper composition approach, and the final composition outcome will be sent to the requester as an answer to the service demand [24, 25]. The discussed flow is illustrated in Fig. 2. As shown in the figure, there are four components, including user, task decomposition, service discovery and selection, and service composition. The user requests a complex service/task that cannot be fulfilled by a single service. The decomposition module decomposes the requested task into some subtasks each of which can be satisfied by a single service. Then, the service discovery module finds appropriate and accessible services for fulfilling each subtask. Another responsibility of this module is to select one service among all the found services for each task. The selection must be done in a way that the resulting composite service becomes an optimal service. Finally, the composite service will be delivered to the user.

Generally, a service composition problem can be modeled as a directed acyclic graph (DAG) in which each node represents a subtask of a complex request and the edges represent the relation among the nodes [26]. In order to fulfill each subtask, several numbers of candidate services are suggested from the services pool with the same functionality and different non-functional features such as QoS and QoE values. The objective of the service composition method is to select one appropriate service from a set of candidate services for accomplishing the corresponding subtask in such a way that the total composite service is an optimal service among all other potential solutions, which should be able to satisfy the given complex request [27]. The evaluation and validation of the final composite service are mostly done by evaluating the QoS and QoE values of the composite service. The QoS values of the composite service can be calculated according to the QoS values of its components using aggregation functions. The aggregation functions are selected with regard to the adopted composition pattern. Table 3 depicts some the well-known composition patterns and their corresponding aggregation functions for four QoS parameters adopted in this paper, including *availability, reliability, response time,* and *latency*. Moreover, QoE is a metric for evaluating the users' satisfaction rate, which concentrates on the entire service from the users' perspective, while QoS involves individual characteristics of the service. In this research, the QoE value of a service is linearly correlated with the QoS values of the corresponding service.

Furthermore, since the QoS parameters have different measurement units, they should be normalized before being used for evaluation of the composite service. All in all, there are two types of QoS parameters, including positive (q^+) and negative (q^-) parameters, which should be optimized to become maximize and minimize respectively. Therefore, there are two formulas for normalization as follows. Equation 1 is for negative parameters and Eq. 2 is for positive parameters. In these equations, $Cs.q_i$ refers to the *ith* QoS parameter (q_i) of the selected candidate service

Table 2 Notations utilized in this paper

Notation	Description
q^-	Minimization quality of service
q^+	Maximization quality of service
q_i	The ith quality
Cs	Candidate service
$Cs.q_i$	The ith quality of the corresponding candidate service
S_i	The ith service
F_i	The ith functionality
r_{ij}	The weight of ith quality related to jth service
qoe_k	The kth user feedback (QoE) value among K users
$\overline{qoe}$	The mean QoE value
$\overline{q_i}$	The mean value for the ith QoS parameter
sol	Solution (a composite service)
w_i	The weight of ith quality
$sol.q_i$	The ith quality of the corresponding solution
$a(s_i)$	Availability value related to ith service
$r(s_i)$	Reliability value related to ith service
$rt(s_i)$	Response time value related to ith service
$l(s_i)$	Lateness value related to ith service

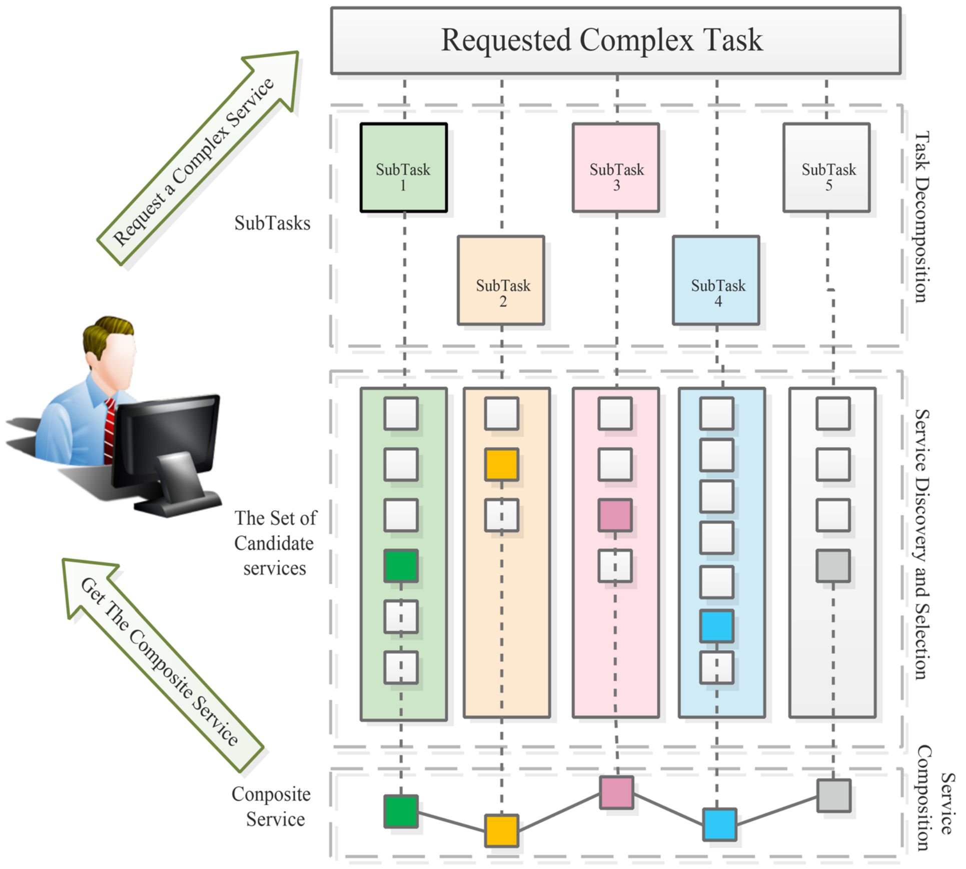

Fig. 1 General service composition flow

(Cs) that is being normalized, and $min\{q_i\}$ and $max\{q_i\}$ represent the minimum and maximum values of the corresponding QoS parameter in the whole dataset.

$$\begin{cases} \dfrac{Cs.q_i - min\{q_i\}}{max\{q_i\} - min\{q_i\}} & max\{q_i\} \neq min\{q_i\}, \\ 1 & max\{q_i\} = min\{q_i\}, \end{cases} \tag{1}$$

$$\begin{cases} \dfrac{max\{q_i\} - Cs.q_i}{max\{q_i\} - min\{q_i\}} & max\{q_i\} \neq min\{q_i\}, \\ 1 & max\{q_i\} = min\{q_i\}. \end{cases} \tag{2}$$

3.2 Example scenario

We utilize edge intelligence-assisted remote infection detection (S) application for pandemic cases such as COVID-19 as an example scenario to clarify the proposed framework as much as possible. This example scenario consists of several services such as vital sign analyzer service (S1), symptom checker service (S2), storage service (S3), and alert sending service (S4). The vital sign analyzer service gets the vital signs and based on the pre-defined threshold keeps the suspicious data by omitting the unimportant signs. We have taken three vital signs into account, including *oxygen saturation*, *heartbeat rate*, and *body temperature*. The symptom checker service controls the suspicious signs received from S1 by comparing them with

Table 3 Aggregation functions for QoS parameters based on the composition pattern

QoS parameter	Sequential	Parallel	Optional	Circular
Availability	$\prod_{i=1}^{n} a(s_i)$	$min\{a(s_i)\}$	$\prod_{i=1}^{n} p * a(s_i)$	$(a(s_i))^k$
Reliability	$\prod_{i=1}^{n} r(s_i)$	$min\{r(s_i)\}$	$\prod_{i=1}^{n} p * r(s_i)$	$(r(s_i))^k$
Response time	$\sum_{i=1}^{n} rt(s_i)$	$max\{rt(s_i)\}$	$\sum_{i=1}^{n} p * rt(s_i)$	$k * rt(s_i)$
Latency	$\sum_{i=1}^{n} l(s_i)$	$max\{l(s_i)\}$	$\sum_{i=1}^{n} p * l(s_i)$	$k * l(s_i)$

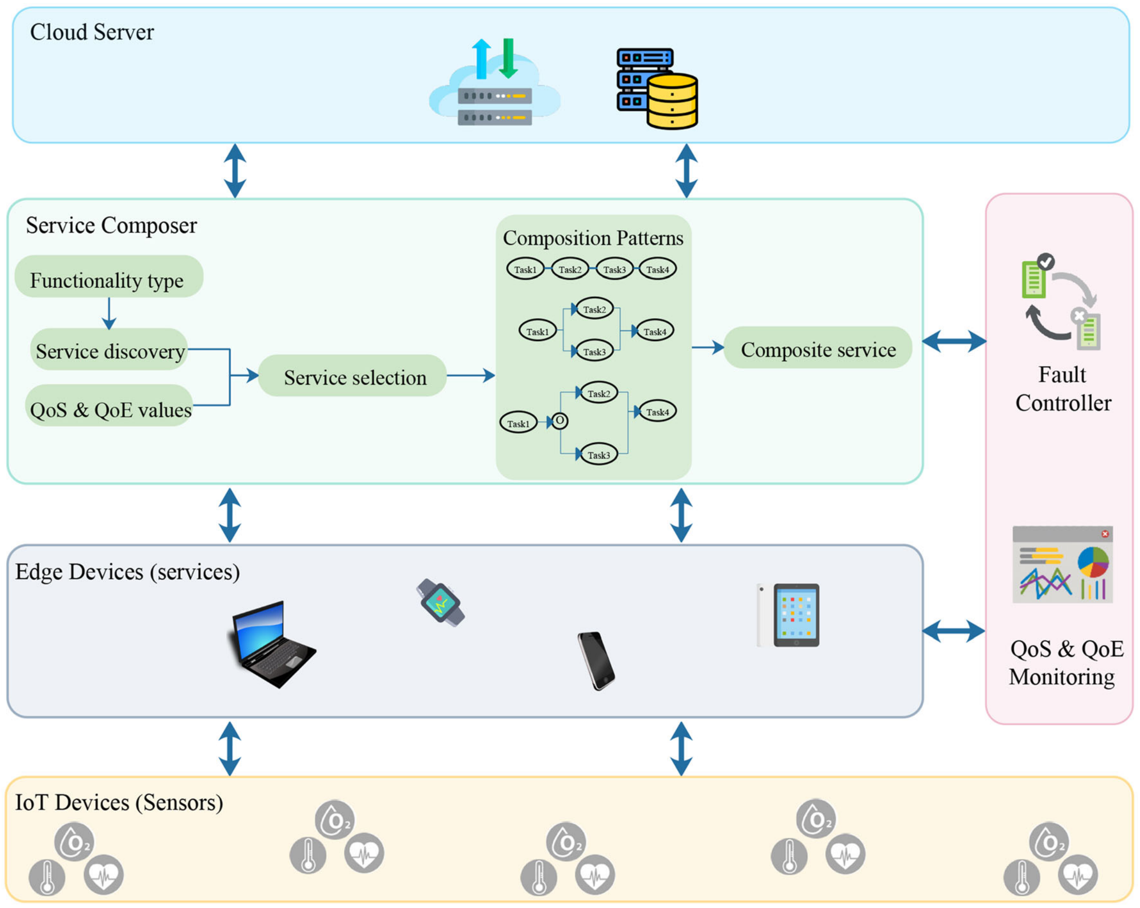

Fig. 2 The overview of the proposed example scenario

the symptoms in the database to detect whether the person is infected or not. The database consists of various combinations of vital signs that define the severity level of the patient's condition. The storage service oversees storing small pieces of the database. The alerting service is responsible to send an alert to the user, caregivers, and family members in case of infection detection/encounter with a suspected infection case. Each service can be performed via one or a set of edge devices in the proposed edge-assisted framework. Each edge device has functionality and a set of QoS and QoE values. In this scenario, the *S* can be considered as a complex service that can be accomplished via four subtasks including S1, S2, S3, and S4, each of which can be deployed in a single/a set of edge device(s). The results of these various components should be aggregated and delivered as a single result which can be defined as a service composition problem since we consider each edge device as an abstract service.

3.3 Framework overview

Let's assume that there are *m* edge devices in the edge computing environment. Each device has functionality and a set of QoS values and QoE values. In the proposed framework, we consider four QoS parameters for each edge service (edge device), including *reliability, availability, latency*, and *response time*. According to the example scenario, the functionality of edge devices can be a vital sign analyzing ($F1$), symptom checking ($F2$), data storing ($F3$), or alert sending ($F4$). As mentioned before, a single big AI task should be decomposed into several small AI subtasks to be able to be deployed on the resource-constrained edge devices in an edge network. Each subtask can be deployed on a single edge device or a set of edge devices in order to be performed. Thus, for implementing the composition process, we have considered a solution in a form of an array that includes four sections corresponding to each subtask. For more illustration, suppose that there are 30 edge apparatuses in the network with different functionalities, including $F1$ (10 edge devices), $F2$ (9 edge devices), $F3$ (7 edge devices), and $F4$ (4 edge devices). On the other hand, for the composition request, we need 3, 2, 2, and 1 devices with the functionalities of $F1$, $F2$, $F3$, and $F4$ respectively. Therefore, the structure of the possible solution can be presented as Fig. 4.

In this example, for performing $S1$ we need three edge devices each of which should be selected from the 10 available devices with the functionality of F1, and with different QoS and QoE values. These three are responsible for vital sign analysis. The considered vital signs in this scenario include oxygen saturation rate, heartbeat rate, and body temperature. Each of the selected edge devices will be in charge of one of these vital signs. Similarly, $S2$, $S3$, and $S4$ need the pre-defined number of edge devices to accomplish their responsibilities as mentioned before. It should be mentioned that there is a dependency between the main subtasks $S1$, $S2$, $S3$, and $S4$. It means that the result of Si may be used in Sj $(i < j)$ except $S3$ which is only in regular communication with S2. Figure 3 depicts the relation between subtasks clearly. In this figure, S1 refers to subtask1 (vital sign analyzing) which needs three edge devices with the relevant functionality (F1). The results of this subtask will pass to S2 which refers to subtask2 (symptom checking). This subtask needs two edge devices to check the symptoms for detecting the possible infection or suspicious cases. In this regard, S2 continuously communicates with S3, which stores different combinations of symptoms states, to understand whether the current case is in a severe condition or not. Subtask3 (S3) includes various symptom combination states and needs two edge devices to be deployed, one device for storing severe states, and one device for storing non-critical and normal states. The severe state can be, for example, a case with the combination of symptoms such as fever higher than 39.5, oxygen rate lower than 80%, and heartbeat rate higher than 90. There can be different combinations of these three symptoms, which indicate the critical state and vice versa for normal conditions. However, since our focus is not on detecting a disease we are not going to detail. The focus of this paper is to integrate the subtasks of AI and deliver a composite solution. It should be mentioned that we implemented the dependencies between subtasks by considering the order of subtasks in the solution structure as shown in Fig. 4 with respect to the functionality of sub services (edge devices).

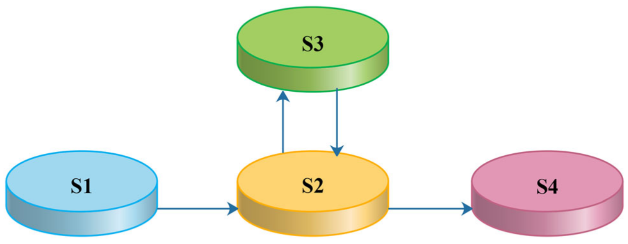

Fig. 3 Relation between the AI sub-tasks

3.4 QoS model

As stated before, in this paper four QoS metrics are considered for each edge service (device), including *availability, reliability, latency*, and *response time*. We adopted the values of these metrics from the QWS dataset [28] which is a popular dataset for service composition problems among researchers. It includes 2507 real services with 6 QoS metrics, 4 of which are used in this paper. The initial values of QoS parameters are obtained from the dataset. Then, as the process furthers, the QoS values are randomly updated each time a service is being invoked to perform a dynamic service composition since in real scenarios the QoS values are not static for the real services. In order to conduct the updating process, we consider QoS and QoE monitoring component in the framework to manage changing the values of QoS and QoE metrics in each invocation of the edge service randomly. The QoS updating process is fully random. We first generate a random number between 0 and 1. If the generated random number is greater than a pre-defined number (e.g., 0.3) then the QoS/QoE updating module will be invoked. This module is responsible to update QoS values of the corresponding service randomly. Being correlated with QoS values, the QoE value will also be updated accordingly. Moreover, since the QoS parameters have different units, the following normalization equations (Eqs. 1 and 2) are used to normalize the parameters. There are two types of considered QoS parameters in this paper, including positive (q_i^+), and negative (q_i^-) parameters. The positive parameters infer to the QoS parameters such as reliability, availability and QoE that should be maximized. On contrary, the negative parameters refer to QoS parameters that should be minimized, like latency and response time.

3.5 QoE model

For each service in the dataset, first, we generate K random QoE values (K denotes the number of subjective tests. It means that for each service K number of persons give their feedback in a number value format). Afterward, using Pearson equation (Eq. 3) [29], we calculate Q coefficient values (weights) for each service corresponding to its QoS values. In this paper, we have considered $K = 100$, and Q is equal to 4 since we have four QoS parameters in our proposed model. Finally, utilizing the calculated weights

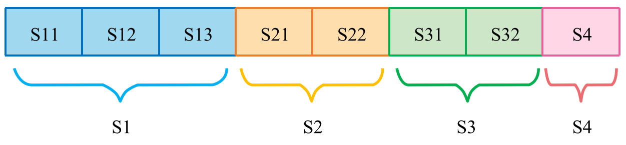

Fig. 4 The sample solution structure for AI sub-task composition

(r_{ij}) and QoS values given in the dataset, we can calculate the final QoE value for the corresponding service in the dataset. To this end, we have used Eq. 4, which linearly correlates QoE and QoS values, as follows:

$$r_{ij} = \left| \frac{\sum_{k=1}^{K} (q_{ij} - \overline{q_i})(qoe_k - \overline{qoe})}{\sqrt{\sum_{k=1}^{K} (q_{ij} - \overline{q_i})^2} \sqrt{\sum_{k=1}^{K} (qoe_k - \overline{qoe})^2}} \right|, \tag{3}$$

$$QoE(s_j) = \sum_{i=1}^{Q} r_{ij}(s_j) * q_{ij}(s_j), \tag{4}$$

where q ij is the current QoS value for jth service (s_j), that we intend to calculate its weight. $\overline{q_i}$ is the mean value for the ith QoS parameter considering the whole services in the dataset, and qoe_k is the kth user feedback value among $K(K = 100)$ users.

In each invocation of the service, the QoS and QoE values of the corresponding service would be randomly modified. Since the QoE value has a linear correlation with QoS values and QoS values will be changed, the QoE value will be updated based on the updated parameters of QoS. The generated/calculated QoE values are added to the dataset as another QoS value.

3.6 Evaluation function

In order to evaluate the potential solutions, the following function (Eq. 5) is used. This function is based on the QoS values (including the QoE value as a QoS criterion). The problem of edge intelligence service composition is a multi-objective problem, which includes five objectives (four QoS parameters and one QoE value) that should be optimized. We use the simple additive weight (SAW) method to linearly combine the objectives and transform the problem into a single-objective one. In Eq. 5, $w_i(s)$ are weights/coefficients for each of the QoS parameters of a composite service (solution) and they should sum up to one ($\sum w_i = 1$). These weights can be defined by the service requester. In this paper, since the main focus is on *reliability*, its coefficient is considered as $w_{reli} = 0.5$ and for the rest of parameters (*availability, response time, latency, and QoE*) the coefficients are considered as $w_i = 0.125$. Aiming to have a minimization problem, we put the positive parameters in the denominator. In this equation, *sol* denotes the current solution that we are calculating its fitness value.

$$Fitness(sol) = \sum_{i=1}^{2} w_i * sol.q_i^- + \sum_{i=3}^{5} w_i * \frac{1}{sol.q_i^+}. \tag{5}$$

Each solution (a composite service) consists of several sub-services, each of which has its own QoS and QoE values. The QoS values of the composite service, which should be used in Eq. 5, is the aggregation values of its sub-services. There are many different aggregation functions for various composition patterns, which are mentioned in Table 3.

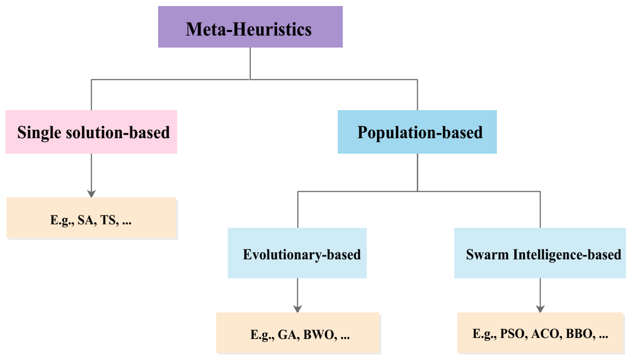

Fig. 5 Classification of meta-heuristic algorithms

3.7 Fault monitoring

In order to produce a reliable composite service, we have developed a fault controller module that monitors the composition process to prevent a possible fault in terms of the selected candidate services. During the composition process, this module takes care of the selected services and checks their non-functional features to find out whether they are satisfying the constraints or not. We defined the constraints for reliability (0.85) and availability (0.70). In case of any fault, the fault controller module will replace the selected faulty service with another relevant candidate service from the dataset with the same functionality. Thus, a replacement strategy is utilized to prevent any possible service fault in the proposed model.

4 Meta-heuristics for composing AI subtasks

This subsection discusses the adopted approaches for solving the AI subtasks composition problem in this paper. As discussed before, considering the fact that a complex AI task, such as ANN, cannot be deployed on the edge devices, it must be decomposed; then, being distributed among the edge devices. Once the AI subtasks accomplish, they need to be composed and be transformed into a single solution. Since meta-heuristic algorithms best suit the NP-Hard problems, we adopted three well-known meta-heuristic algorithms to evaluate our proposed framework. We classified the meta-heuristic algorithms as shown in Fig. 5 into two main groups, including single-solution-based [e.g., SA, tabu search (TS)], and population-based. The population-based algorithms are also classified into two groups, encompassing evolutionary-based [e.g., GA, black widow optimization (BWO)] and swarm intelligence-based [e.g., bio-geographical based optimization (BBO), ant colony optimization (ACO), PSO]. We adopted one algorithm from each group, including SA [30, 31], PSO [32], and BWO [33], to evaluate our proposed framework.

All in all, meta-heuristic algorithms utilize various strategies in exploring the problem's search space. However, they have some basic flows in common. For instance, three adopted algorithms have the following features in share.

- *Random initialization* they all generate their initial solution(s) randomly.
- *Solution encoding* all three algorithms adopt the same encoding for their solution in the form of an array with the length of n (n is the number of subtasks, which is equal to eight in our example scenario mentioned in Sect. 1).
- *Fitness function* the fitness function, cost function, evaluation function, or objective function is the function mentioned as Eq. 5, which is used by all three algorithms with the same input data.
- *Iteration* all three algorithms iteratively approach towards the near-optimal solution. It means that they iteratively repeat applying some operations on their current solutions to improve them and move them to the objective.
- *Operators* although each algorithm uses different operators to reach the near-optimal solution, some operators have similar behavior. For example, the mutation operator in BWO and neighbor generation in SA, both randomly choose a component in the current solution and replace it with a random value that fits the search space.

The exclusive information and pseudo-codes of the algorithms are discussed in the ensuing subsections. It should be noted that "The initial parameter settings and the composite service request, including the initial composition matrix" are the input and "Near-optimal composite service (AI task)" is the output for all three algorithms.

4.1 Simulated annealing algorithm

Simulated annealing (SA) is a meta-heuristic algorithm for the optimization problems. This algorithm imitates the physical procedure of heating a material and then gradually lowering the temperature to let the material get cool [30]. The developed SA algorithm is illustrated in Algorithm 1. The algorithm starts with a random potential solution for the related issue (line 3). Then, a pre-defined number of neighbors for the initial solution are produced based on a random modification on the original solution. Among the produced neighbors, the best one will be chosen to be used as the best solution, if it overcomes the initial solution (lines 5 to 15). If none of the neighbors could prevail over the initial solution, the best neighbor will have a chance to be selected as the best solution based on Boltzmann probability (lines 16 to 23). The initial value of temperature T is set as 100,000 and decreased in each iteration by the ratio of Eq. 6.

$$Temp\,Reduction\,Ratio = (T - TF)/MaxIter, \tag{6}$$

where TF is the minimum temperature, and $MaxIter$ is the maximum iteration of the algorithm, which are set to 100 and 200, respectively.

Briefly, SA algorithm starts with a single random solution (*sol*). Then, generates some neighbors for it and compare the best neighbor with the *sol*. If the best neighbor is better than *sol*, it will be assigned to *sol*; otherwise, based on Boltzmann probability (Eq. 7), the algorithm will decide on whether replace *sol* with the best neighbor or not. This will be repeated until the stop condition is satisfied. As stated in Sect. 5, the stop condition is considered as the maximum number of iteration, which is equal to 200 in all the experiments.

Algorithm 1: Simulated Annealing

```
1  Input The initial parameter settings and the
     composite service request, including the initial
     composition matrix
2  Output Near optimal composite service (AI
     task)
3  Generate initial solution sol randomly
4  Evaluate sol
5  while Stop condition is not satisfied do
6      $BestSol = Empty$
7      for $i = 1$:Number of Neighbors do
8          Generate a random neighbor $sol'$
9          Evaluate $sol'$
10         if $sol'$ is better than bestSol then
11             $bestSol = sol'$
12         end
13     end
14     if bestSol is better than sol then
15         $sol = bestSol$;
16     else
17             // Calculating Boltzmann Probability
18         $E = (bestSol - sol)/sol$;
19         $P = \exp(-E/T)$    // T is temperature
20         if $Rand(0,1) < P$ then
21             $sol = bestSol$
22         end
23     end
24     Decrease T
25     Display the bestSol as the best solution of
         this iteration
26 end
27 Return the bestSol as the final solution
```

$$P = -[(bestSol - sol)/sol]/T, \qquad (7)$$

where *bestSol* is the best neighbor, *sol* is the initial/current solution, and *T* is the current temperature.

4.2 Particle swarm optimization algorithm

PSO is a meta-heuristic algorithm that represents the movement of organisms like fish school and bird flock towards a new settlement of food source [32]. Algorithm 2 illustrates the pseudo-code of the PSO algorithm. PSO is a population-based swarm intelligence-based algorithm, which starts with a pre-defined number of initial random solutions named particle (line 3). Then, solutions are sorted with regards to their fitness value (line 4), and the best solution is defined as the global best (gBest) (line 5). Each solution keeps its own best state during the whole process as the local best (lBest). In each iteration, the algorithm, first updates the velocity for all solutions (line 8, where $R1$ and $R2$ are random numbers between 0 and 1, w is inertia weight, and $C1$ and $C2$ are constant values that should be determine as hyper parameters). Then, based on the updated velocity values, their position should be updated (line 9). Consequently, their fitness value will be recalculated using the fitness function (line 10). Then, considering the new fitness value, gBest and lBest should be updated (lines 11 to 16). These steps will be repeated until the stop condition. Ultimately, the last gBest will be returned as a final solution of PSO.

4.3 Black widow optimization algorithm

The implementation of the BWO algorithm is presented in Algorithm 3. This algorithm is a population-based evolutionary algorithm, which is inspired by the weird lifestyle of black widow spiders. The stages of the algorithm [33] are as follows. The algorithm is started by the initial random population of potential solutions (line 3 in Algorithm 3. Then, the procreate operation produces children for each pair of parents (line 9). The number of children will diminish by cannibalism operators (lines 10 and 11). The mutation operator is responsible for modifying the existing solutions and consequently producing novel solutions (lines 14 to 18). Finally, the number of current solutions should be decreased to the initial value by omitting the worst ones (lines 19 to 21). These steps will be repeated until the stop criteria are met. Then, the best solution will be returned as the concluding solution of the algorithm for the corresponding issue.

Algorithm 2: Particle Swarm Optimization

```
1  Input The initial parameter settings and the
     composite service request, including the initial
     composition matrix
2  Output Near optimal composite service (AI
     task)
3  Generate initial solution population as particles
4  Evaluate and sort the particles
5  Choose the best particle (solution) as gBest
6  while Stop condition is not satisfied do
7      for each particle in the population do
8          $V_{new} = w.V_{old} + R1.C1.(lBest - particle) + R2.C2.(gBest - particle)$
9          $Position_{new} = Position_{old} + Velocity_{new}$
10         Evaluate the updated particle by the
             fitness function
11         if $particle < gBest$ then
12             $gBest$=updated particle
13             if $lBest < gBest$ then
14                 gBest=lBest
15             end
16         end
17     end
18     Display gBest as the best solution of this
         iteration
19 end
20 Return gBest as the final solution
```

Algorithm 3: Black Widow Optimization

```
1  Input The initial parameter settings and the
     composite service request, including the initial
     composition matrix
2  Output Near optimal composite service (AI
     task)
3  Generate n initial solution population as widows
4  Based on the procreate rate choose $nr$ best
     widow from the population and keep them in
     pop1
5  Based on mutation rate calculate $nm$
6  while Stop condition is not satisfied do
7      for $i=1:nr$ do
8          Randomly choose parents from $pop1$
9          Produce children (solutions)
10         Omit father
11         Based on cannibalism ratio omit some
             children (solutions)
12         Keep the rest of children in pop2
13     end
14     for $i = 1 : nm$ do
15         Select a widow (solution) from $pop1$
16         mutate the selected widow
17         keep it in $pop3$
18     end
19     Add $pop2$ and $pop3$ to the initial widows
20     Sort the widows
21     Choose $n$ best widows and omit the rest
22     Display the best as bestWidow in this
         iteration
23 end
24 Return bestWidow as final solution
```

5 Experiments and simulation results

This section consists of two parts. The details of experiments including the dataset information, experimental scenarios, and the used system information are mentioned in Sect. 5.1. Then, the simulation results, comparisons, and discussion are stated in Sect. 5.2.

5.1 Simulation setup and metrics

To validate the proposed AI subtasks composition framework, evaluations have been conducted on a laptop with an Intel Core-i7-10750H CPU 2.60 GHz and 16 GB RAM, using MATLAB R2020b. In order to carry out a fair comparison, all three algorithms were fed with the same set of services collected from the dataset. As mentioned before, the QWS dataset has been used as the initial QoS values in all the experiments. However, the QoS values are updated randomly during the composition process to perform a dynamic composition framework. We have collected the information of four QoS parameters, encompassing *availability, reliability, response time*, and *latency*. Furthermore, as mentioned in Sect. 3.5 calculated QoE values have been adopted as a new QoS parameter.

The mentioned QoS parameters are adopted as our evaluation metrics besides the scalability and execution time that can be defined as follows:

- *Reliability* alludes to the rate of error messages to the total messages.
- *Availability* specifies the proportion of successful invocations to the total invocations of a service.
- *Latency* is defined as the communication delay, the time duration that message spends on the path from source to destination.
- *Response time* refers to the total time duration that takes the service requesters to get the response from the system for their request. It can be determined as the summation of service time and latency.
- *Scalability* implies the ability to conduct AI subtasks composition in large-scale network environments.
- *Execution time* indicates the time duration takes to run the corresponding algorithm in order to achieve the composition result.

Besides, three meta-heuristic algorithms, namely BWO, PSO, and SA, have been adopted to evaluate the proposed framework. In all experiments, the maximum number of iterations is set to 200 and is considered as a stop criterion for all algorithms. Moreover, the number of initial population for BWO and PSO is set to 100, and for SA is set to one since SA is a single-solution-based algorithm.

5.2 Simulation results and discussion

In this section, the experimental results are provided and discussed. The experiments have been done in five types, including scalability analysis (Sect. 5.2.1), fault monitoring analysis (Sect. 5.2.2), success rate (Sect. 5.2.3), similarity analysis (Sect. 5.2.4), and computation time comparison (Sect. 5.2.5). The detail information of each experiment is mentioned in the following relevant subsections.

5.2.1 Scalability analysis

In order to conduct the scalability analysis, we have considered different network sizes by varying the number of edge devices in the network. In the beginning, the number of edge devices is defined as 30. Then, it is increased up to 180 with the step of 30. For each scenario, we run the algorithm for 10 times and collect the mean value as the final result.

The results of this experiment are depicted in Figs. 6, 7, 8, 9 which represent mean fitness, mean reliability, mean availability, and mean QoE values respectively. Moreover, Table 4 shows the analysis results for the mentioned

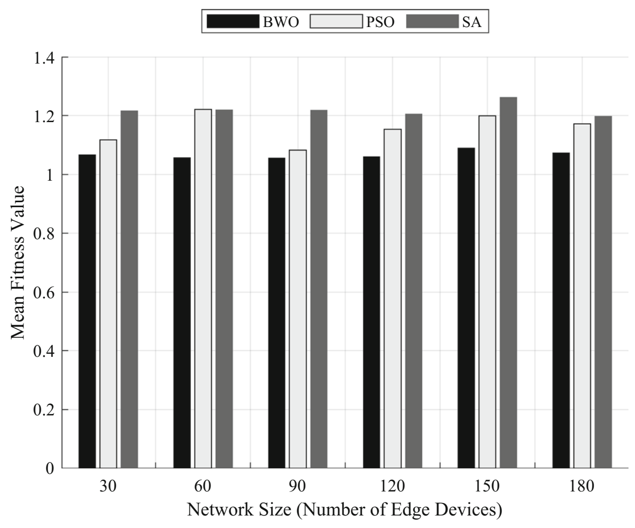

Fig. 6 Evaluation of Fitness value with different network size

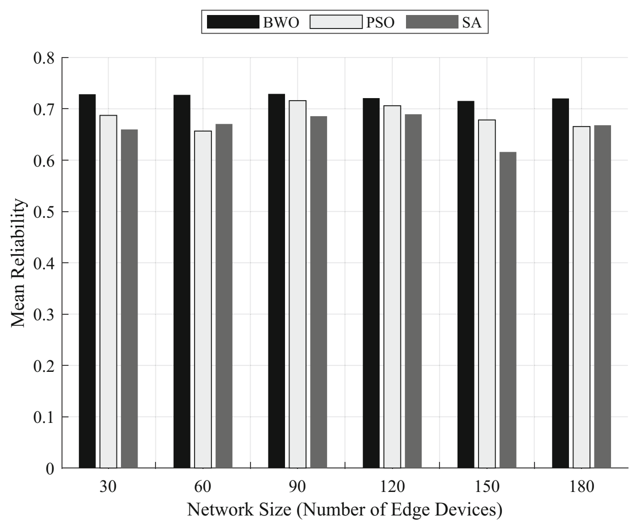

Fig. 7 Evaluation of Reliability value with different network size

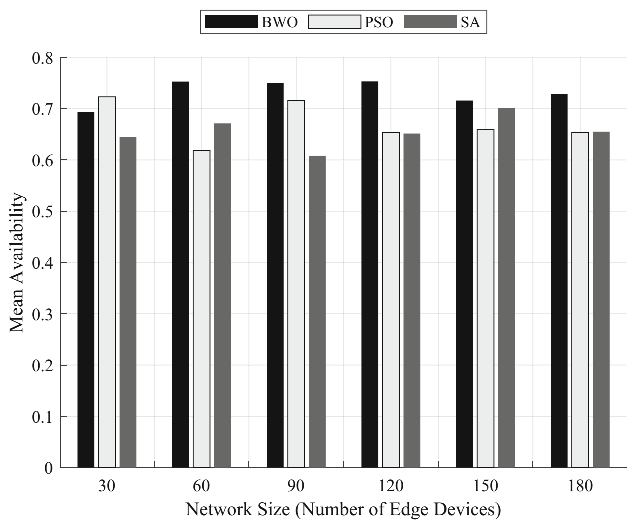

Fig. 8 Evaluation of Availability value with different network size

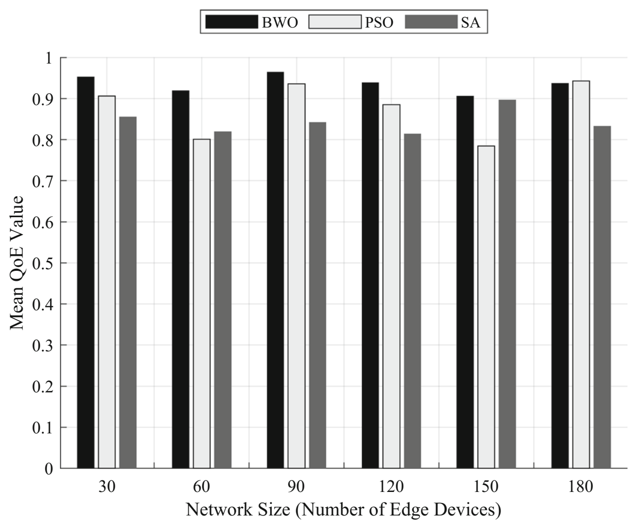

Fig. 9 Evaluation of QoE value with different network size

metrics. It can be seen that in almost all cases there are no remarkable changes in the results of various network sizes. Therefore, scalability is preserved.

5.2.2 Fault monitoring analysis

In order to prevent the possible service faults, we adopted the local fault monitoring (LFM) method mentioned in Sect. 3.7. We run the algorithms two times, one with applying LFM and the other without it, and compared their outcomes. Figures 10, 11, 12, and 13 demonstrate the comparison results for fitness value, the convergence figure for fitness value, reliability, and availability respectively. It can be seen that in all cases the results of the scenario with LFM outperform the other scenario. Also, comparing the algorithms show that BWO in the majority of cases has better performance. Moreover, Fig. 10 shows that, in this scenario, BWO has better convergence speed in comparison to PSO and SA algorithms.

5.2.3 Success rate analysis

One of the vital objectives of service composition strategies is to collect reliable services to produce and deliver a composite service that fulfills the service requester constraints. However, the uncertainty of QoS values in the selected services often contributes to the deviation of user constraints and preferences. Thus, service composition failure is one of the critical challenges in this scope. To this end, this section is addressed this challenge and compares the adopted algorithms in this regard.

Table 4 Scalability analysis

	Network size (number of edge devices)																	
	30			60			90			120			150			180		
Algorithm	PSO	SA	BWO	PSO	SA	BWO	PSO	SA	BWO	PSO	SA	BWO	PSO	SA	BWO	PSO	SA	BWO
Fitness	1.12	1.22	1.07	1.22	1.22	1.06	1.08	1.22	1.06	1.15	1.21	1.06	1.20	1.26	1.09	1.17	1.20	1.07
Availability	0.72	0.64	0.69	0.62	0.67	0.75	0.72	0.61	0.75	0.65	0.65	0.75	0.66	0.70	0.72	0.65	0.66	0.73
Reliability	0.69	0.66	0.73	0.66	0.67	0.73	0.72	0.69	0.73	0.71	0.69	0.72	0.68	0.62	0.72	0.67	0.67	0.72
QoE	0.91	0.85	0.95	0.80	0.82	0.92	0.94	0.84	0.96	0.88	0.81	0.94	0.87	0.90	0.91	0.94	0.83	0.94

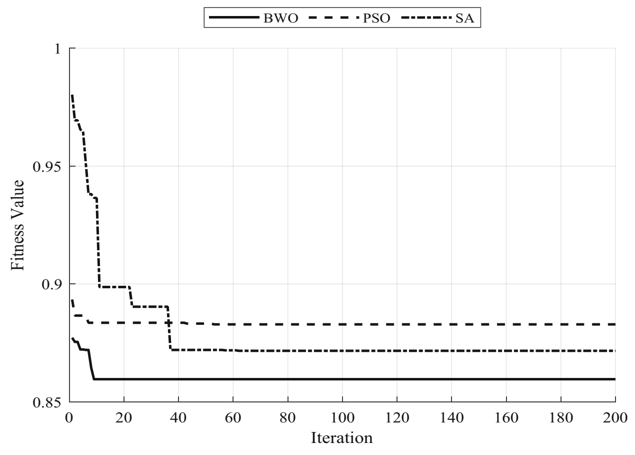

Fig. 10 Convergence of Solutions with LFM

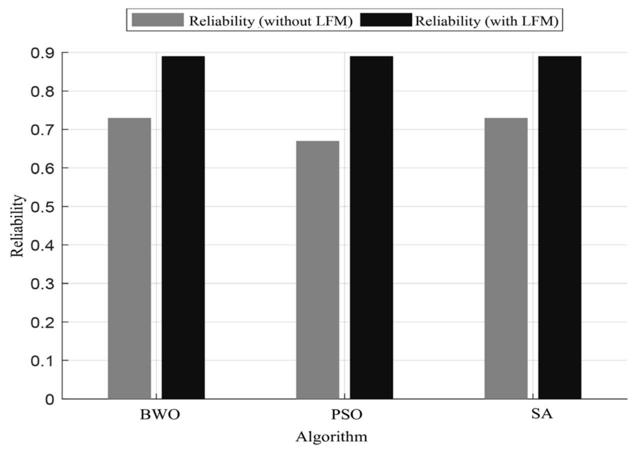

Fig. 12 Evaluation of Reliability value with(out) LFM

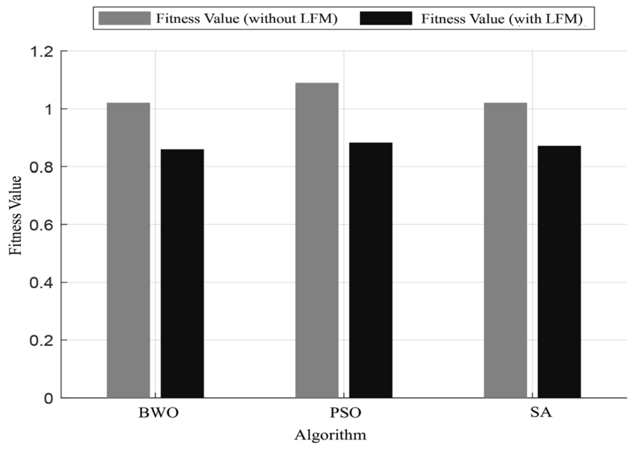

Fig. 11 Evaluation of Fitness value with(out) LFM

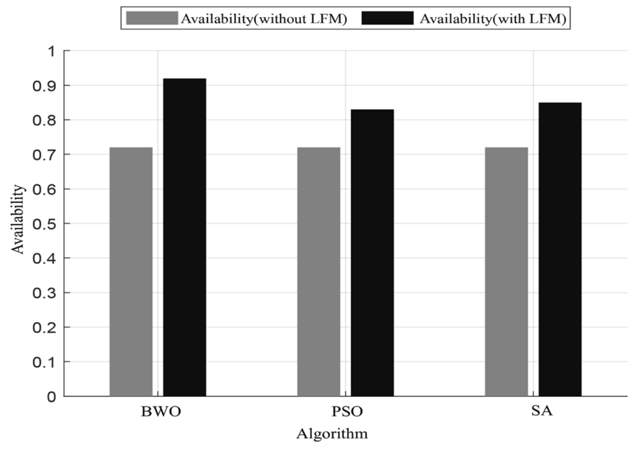

Fig. 13 Evaluation of Availability value with(out) LFM

Success ratio (SR) reveals how often the obtained aggregated QoS value meets the corresponding constraint value. To assess this, Eq. 8 is adopted to count the number of composite services, in ten times run, which meet the constraints for both reliability and availability. Then, the attained *Nsr* value is used in Eq. 9 to calculate the SR for the corresponding approach.

$$NSr = \sum_{i=1}^{run} \begin{cases} 1, & If \quad Reli(s_i) \geq C_{Reli}\, and \\ & \quad Avail(s_i) \geq C_{Avail}, \\ 0, & \quad Otherwise, \end{cases} \tag{8}$$

$$SR = \frac{NSr}{10} \times 100\%, \tag{9}$$

where *run* refers to the number of composite services that we evaluate, s_i is the *ith* service, *Reli* and *Avail* are the aggregated value of reliability or availability for the corresponding service (s_i, and C_{Reli} and C_{Avail} are the predefined constraint values for the reliability and availability, respectively.

In order to conduct this experiment, we have considered different network sizes starting from 30 edge devices increasing it to 180 with the step of 30. All three algorithms have been run 10 times for each network size, and in each scenario, the obtained best solution is evaluated based on two pre-determined constraints. Since this paper focuses on reliability and availability, we have considered the constraint on these two parameters and set them as $C_{Reli} = 0.85$ and $C_{Avail} = 0.70$. The result is depicted in Fig. 14. Also, the average SR for BWO is obtained as 83.33%, and 65%, and 38.33% respectively for PSO, and SA. Higher the SR, lower the failure ratio. Thus, according to this experiment BWO achieves a high SR and accordingly has a lower failure rate comparing to PSO and SA algorithms. With respect to Fig. 14, this is even correct for the large network sizes.

5.2.4 Similarity analysis

This experiment is adopted from [34] and intends to compare the similarity factor of the employed algorithms. The similarity factor evaluates the obtained fitness values between two algorithms one by one. Since BWO has outperformed the other two algorithms, namely, PSO and SA in the majority of the previous experiments, we have selected it as the main algorithm for this test. Thus, the similarity of PSO and SA against BWO has been assessed by Eq. 10. In this equation, Algorithm **X** can be either PSO or SA and Algorithm **Y** in both cases is BWO.

$$Similarity = \frac{FitnessValue_{\ Algorithm\mathbf{X}}}{FitnessValue_{\ Algorithm\mathbf{Y}}}. \tag{10}$$

Aiming to have a comprehensive analysis, the similarity of PSO and SA has also been calculated by Eq. 10 and the result is shown in Fig. 15 using green color. For this test, we have considered SA as Algorithm **Y** and PSO as Algorithm **X**.

The attained values from Eq. 10 show the similarity of the two algorithms in terms of their fitness value. In order to conduct this experiment, we run the algorithms 20 times, and their similarities have been calculated for each run. The network size is considered 30 edge devices. The other setups, such as the number of the initial population and the maximum number of iterations, have been kept the same as previous experiments. Figure 15 demonstrates the result of this experiment. If the obtained similarity value is equal to 1, it means that the compared algorithms are totally similar. Furthermore, the similarity value greater than 1 reveals the superiority of the selected main algorithm to the other one. The similarity value less than 1 means that the selected main algorithm behaves weaker than the compared algorithm. As mentioned before, BWO has been selected as the main algorithm (Fig. 15), and for the PSO–SA line, SA has been the selected main algorithm.

According to the attained outcomes shown in Fig. 15 it can be concluded that BWO has better performance in comparison to PSO and SA since the similarity values obtained for PSO–BWO and SA–BWO are greater than 1. Moreover, PSO has better performance compare to SA since the similarity values of SA against PSO are mostly less than 1. Overall, the achieved results from 20 executions of our experimental algorithms can be summarized as follow:

- BWO behaves better in 19 executions out of 20 execution in comparison to PSO. Figure 15 also depicts this fact. At the first execution, the line PSO–BWO (similarity of PSO against BWO) almost touches 1. The

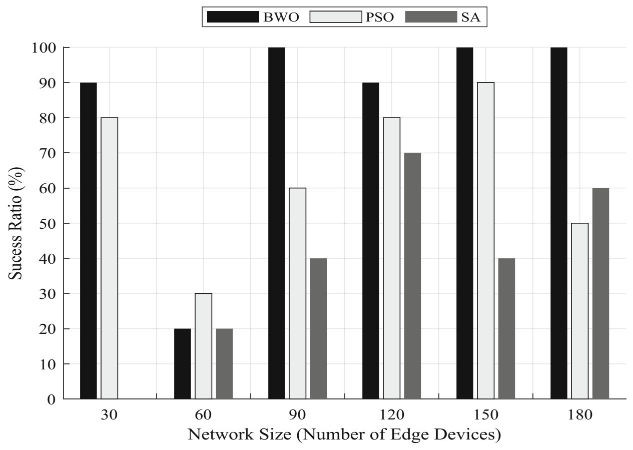

Fig. 14 Evaluation of success ratio

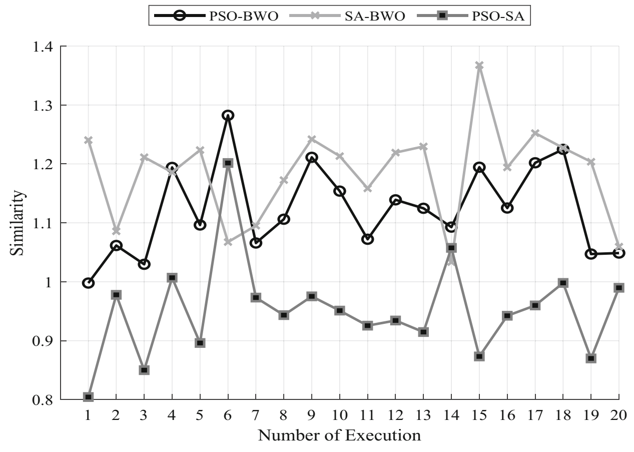

Fig. 15 Similarity evaluation of the adopted algorithms

real similarity value for the first execution is 0.99778, which is less than 1. However, the overall efficiency of BWO over PSO is 95%.

- BWO outperforms SA in all of 20 executions, which implies 100% efficiency of BWO against SA. Figure 15 shows this visually. Considering the SA–BWO, the similarity values for all 20 executions are greater than 1.
- PSO defeats SA in 17 executions out of 20 executions, which indicates 85% efficiency of PSO against SA. With regards to the PSO–SA line in Fig. 15, it can be seen that only three of similarity values are greater than 1.

5.2.5 Execution time comparison

In this experiment, we evaluated the adopted algorithms in terms of their execution time for finding the best possible solution for AI subtask composition in edge computing. To this end, we have conducted the experiment in different network sizes. The network size is varied from 30 edge devices to 180 edge devices with a step size of 30 as can be seen in Fig. 16. Although SA and PSO have achieved better results on large scales, their execution times have remarkable fluctuation. The result of this experiment shows that the BWO not only does obtain a short execution time, but it has negligible changes through the different network sizes.

6 Conclusions and future directions

This paper depicts a novel dynamic and reliable framework for AI subtasks composition in the edge computing environment for the connected healthcare application. The key motivation is to facilitate the deployment of AI tasks on resource-constrained edge devices. With regards to the fact that the combination of edge computing, service computing, and AI can contribute to offering extremely distributed heterogeneous smart devices abstracted as services, which can be adopted in different smart application cases. Thus, enabling AI deployment on these devices can pave the way for providing smart and efficient systems, such as connected healthcare systems. However, there are some challenges and limitations since the IoT/edge devices suffer from low storage and computation capacity. In this regard, we have addressed this challenge by considering the fact that big AI tasks can be decomposed, distributed on edge devices to be performed close to where data are being produced, and finally, the outcomes are composed and deliver the result to the requester. This issue is mapped on the service composition problem, which is an NP-Hard problem. Hence, since this problem can not be solved in a deterministic way, we have utilized three well-known meta-heuristic algorithms, including PSO, SA, and BWO. We also adopted the QWS dataset as our data source for conducting the experiments. The experimental results validate the applicability of our proposed framework since it demonstrates efficient performance in almost all the tests, particularly for fault prevention. Moreover, among the three adopted meta-heuristics, BWO performs better compared to PSO and SA. More precisely we demonstrate that BWO prevails SA and PSO in all and 95% of experiments respectively.

Despite the high performance of the proposed framework, there are some limitations that we intend to cover in our future research works. The number of subtasks for all the experiments is considered the same and constant in the proposed framework. The only factor for varying the size of the network is the number of edge devices in the network. Also, we have considered edge devices and service requester in the same network, so the fact that an edge device (service) can be moved to the out of network during the composition process is neglected in this research. Besides, another future research direction would be to evaluate the impact of our method on the accuracy and performance of AI algorithms.

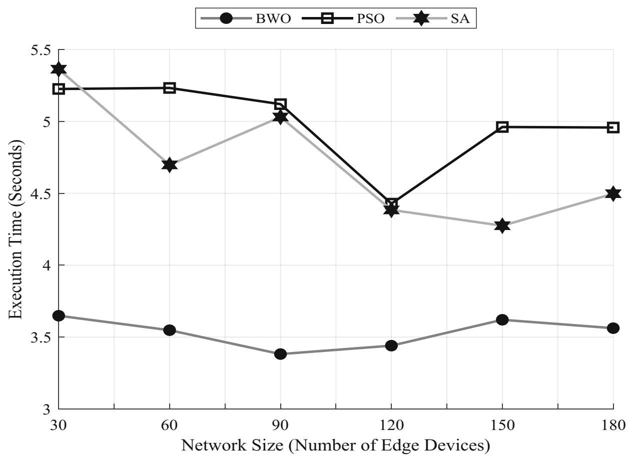

Fig. 16 Evaluation of execution time

Acknowledgements This research was supported by the Faculty of Technological Innovation, Zayed University (ZU), under Grant Number RIF-20130.

Funding The authors have not disclosed any funding.

Data availability Enquiries about data availability should be directed to the authors.

Declaration

Conflict of interest The authors have not disclosed any competing interests.

References

1. Balasubramanian, V., Wang, M., Reisslein, M., Xu, C.: EdgeBoost: enhancing multimedia delivery with mobile edge caching in 5G-D2D networks. In: IEEE International Conference on Multimedia and Expo (ICME) 2019, pp. 1684–1689 (2019)
2. Lu, H., He, X., Du, M., Ruan, X., Sun, Y., Wang, K.: Edge QoE: computation offloading with deep reinforcement learning for Internet of Things. IEEE Internet Things J. **7**(10), 9255–9265 (2020)
3. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet Things J. **7**(8), 7457–7469 (2020)
4. Hayyolalam, V., Aloqaily, M., Ozkasap, O., Guizani, M.: Edge intelligence for empowering IoT-based healthcare systems. IEEE Wirel. Commun. Mag. (2021). https://doi.org/10.48550/arXiv.2103.12144
5. Hayyolalam, V., Aloqaily, M., Özkasap, Ö., Guizani, M.: Edge-assisted solutions for IoT-based connected healthcare systems: a literature review. IEEE Internet Things J. **3**, 1 (2021). https://doi.org/10.1109/JIOT.2021.3135200
6. Rahman, M.S., Khalil, I., Atiquzzaman, M., Yi, X.: Towards privacy preserving AI based composition framework in edge networks using fully homomorphic encryption. Eng. Appl. Artif. Intell. **94**, 103737 (2020)
7. Zhao, J., Tiplea, T., Mortier, R., Crowcroft, J., Wang, L.: Data analytics service composition and deployment on edge devices. In: Proceedings of Workshop on Big Data Analytics and Machine Learning for Data Communication Networks, 2018, pp. 27–32 (2018)
8. Balasubramanian, V., Otoum, S., Aloqaily, M., Al Ridhawi, I., Jararweh, Y.: Low-latency vehicular edge: a vehicular infrastructure model for 5G. Simul. Model. Pract. Theory **98**, 101968 (2020)
9. Wang, X., Han, Y., Leung, V.C., Niyato, D., Yan, X., Chen, X.: Convergence of edge computing and deep learning: a comprehensive survey. IEEE Commun. Surv. Tutor. **22**(2), 869–904 (2020)
10. Hayyolalam, V., Kazem, A.A.P.: A systematic literature review on QoS-aware service composition and selection in cloud environment. J. Netw. Comput. Appl. **110**, 52–74 (2018)
11. Hamzei, M., Navimipour, N.J.: Toward efficient service composition techniques in the Internet of Things. IEEE Internet Things J. **5**(5), 3774–3787 (2018)
12. Al Ridhawi, I., Aloqaily, M., Boukerche, A., Jaraweh, Y.: A Blockchain-based decentralized composition solution for IoT services. In: ICC 2020—IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2020)
13. Al Ridhawi, I., Aloqaily, M., Kotb, Y., Al Ridhawi, Y., Jararweh, Y.: A collaborative mobile edge computing and user solution for service composition in 5G systems. Trans. Emerg. Telecommun. Technol. **29**(1), e3446 (2018)
14. Huang, J., Liang, J., Ali, S.: A simulation-based optimization approach for reliability-aware service composition in edge computing. IEEE Access **8**, 50 355-50 366 (2020)
15. Gao, H., Huang, W., Duan, Y.: The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective. ACM Trans. Internet Technol. **21**(1), 1–23 (2021)
16. Wang, R., Lu, J.: QoS-aware service discovery and selection management for cloud-edge computing using a hybrid meta-heuristic algorithm in IoT. Wirel. Pers. Commun. (2021). https://doi.org/10.1007/s11277-021-09052-4
17. Fekih, H., Mtibaa, S., Bouamama, S.: The dynamic reconfiguration approach for fault-tolerance web service composition based on multi-level VCSOP. Procedia Comput. Sci. **159**, 1527–1536 (2019)
18. Elsayed, D., Nasr, E., El Ghazali, A., Gheith, M.: A self-healing model for QoS-aware web service composition. Int. Arab J. Inf. Technol. **17**(6), 839–846 (2020)
19. Laleh, T., Paquet, J., Mokhov, S., Yan, Y.: Constraint verification failure recovery in web service composition. Future Gener. Comput. Syst. **89**, 387–401 (2018)
20. Wang, L., He, Q., Gao, D., Wan, J., Zhang, Y.: Temporal-perturbation aware reliability sensitivity measurement for adaptive cloud service selection. IEEE Trans. Serv. Comput. **3**, 1 (2020). https://doi.org/10.1109/TSC.2020.3046360
21. Peng, Q., Xia, Y., Zhou, M., Luo, X., Wang, S., Wang, Y., Wu, C., Pang, S., Lin, M.: Reliability-aware and deadline-constrained mobile service composition over opportunistic networks. IEEE Trans. Autom. Sci. Eng. **18**(3), 1012–1025 (2020)
22. Hosseini Bidi, A., Movahedi, Z., Movahedi, Z.: A fog-based fault-tolerant and QoE-aware service composition in smart cities. Trans. Emerg. Telecommun. Technol. **32**(11), e4326 (2021)
23. Hayyolalam, V., Pourghebleh, B., Pourhaji Kazem, A.: Trust management of services (TMoS): investigating the current mechanisms. Trans. Emerg. Telecommun. Technol. **31**(10), e4063 (2020)
24. Pourghebleh, B., Hayyolalam, V., Anvigh, A.A.: Service discovery in the Internet of Things: review of current trends and research challenges. Wirel. Netw. **26**(7), 5371–5391 (2020)
25. Hayyolalam, V., Pourghebleh, B., Chehrehzad, M.R., Pourhaji Kazem, A.A.: Single-objective service composition methods in cloud manufacturing systems: recent techniques, classification, and future trends. Concurr. Comput. Pract. Exp. **34**(5), e6698 (2021)
26. Hayyolalam, V., Pourhaji Kazem, A.A.: QoS-aware optimization of cloud service composition using symbiotic organisms search algorithm. J. Intell. Proced. Electr. Technol. **8**(32), 29–38 (2017)
27. Hayyolalam, V., Kazem, A.A.P.: Review of service composition approaches in cloud environment. In: First International Comprehensive Competition Conference on Engineering Sciences in Iran (2018)
28. Eyhab Al-Masri: QWS Dataset (2007). https://qwsdata.github.io/qws2.html
29. Lalanne, F., Cavalli, A., Maag, S.: Quality of experience as a selection criterion for web services. In: Eighth International Conference on Signal Image Technology and Internet Based Systems, pp. 519–526. IEEE (2012)
30. Aarts, E.H., Korst, J.H., van Laarhoven, P.J.: Simulated Annealing. Princeton University Press, Princeton (2018)
31. Abdel-Basset, M., Ding, W., El-Shahat, D.: A hybrid Harris Hawks optimization algorithm with simulated annealing for feature selection. Artif. Intell. Rev. **54**(1), 593–637 (2021)
32. Khanam, R., Kumar, R.R., Kumar, C.: QoS based cloud service composition with optimal set of services using PSO. In: 4th International Conference on Recent Advances in Information Technology (RAIT), pp. 1–6. IEEE (2018)
33. Hayyolalam, V., Kazem, A.A.P.: Black widow optimization algorithm: a novel meta-heuristic approach for solving engineering optimization problems. Eng. Appl. Artif. Intell. **87**, 103249 (2020)
34. Asghari, P., Rahmani, A.M., Javadi, H.H.S.: Privacy-aware cloud service composition based on QoS optimization in Internet of Things. J. Ambient Intell. Humaniz. Comput. (2020). https://doi.org/10.1007/s12652-020-01723-7

Vahideh Hayyolalam received her B.S. Degree in Applied Mathematics from Payam-Noor University of Tabriz, Iran, and her M.Sc. Degree in Computer Engineering from Science and Research Branch, Islamic Azad University, Iran, in 2016. Currently, she is a Ph.D. Student at Koç University, Istanbul, Turkey. Her research interests include optimization, IoT, cloud computing, edge intelligence, machine learning.

Safa Otoum M'19 is an Assistant Professor of Computer Engineering in the College of Technological Innovation (CTI), Zayed University, United Arab Emirates and a Researcher in the Field of Communications and Networks Security. Prior to joining the CTI, she was a Postdoctoral Fellow at the University of Ottawa and has been a Data Scientist in Cheetah Networks, Inc. Ottawa since 2019. She received her M.A.Sc., and Ph.D. Degrees in Computer Engineering from the University of Ottawa, Canada, in 2015 and 2019, respectively. She is actively working on several reputable events within IEEE and ACM. Her research interests include networks security, Blockchain applications, applications of ML and Aim, IoT, intrusion detection and prevention systems. She received several academic and research scholarships, including the prestigious NSERC Canada Graduate Scholarships-Doctoral, the NSERC FSS and the RIF-Zayed University grant. Currently, she is an IEEE Member and a Professional Engineer (P.Eng.) Ontario.

Öznur Özkasap is a Professor with the Department of Computer Engineering, Koç University, Istanbul, Turkey, which she joined in 2000. She was awarded the TÜBİTAK-NATO A2 Ph.D. Research Scholarship Abroad, and was a Graduate Research Assistant at Cornell University, Department of Computer Science from 1997 to 1999, where she completed her Ph.D. Dissertation. From 1992 to 2000, she has been a Research and Teaching Assistant at Ege University, Department of Computer Engineering. She received her Ph.D. Degree in Computer Engineering from Ege University in 2000. Her research interests include distributed systems, peer-to-peer systems, bio-inspired distributed algorithms, energy efficiency, cloud storage and computing, mobile and vehicular ad hoc networks, scalable reliable multicast protocols, security in distributed systems, and computer networks.