Received: 3 April 2023 | Revised: 19 June 2023 | Accepted: 28 June 2023

IET Networks

DOI: 10.1049/ntw2.12097

ORIGINAL RESEARCH

Optimal intelligent edge-servers placement in the healthcare field

Ahmed M. Jasim[1,2] | **Hamed Al-Raweshidy**[1]

[1]College of Engineering, Design and Physical Sciences, Brunel University London, London, UK

[2]Department of Computer Engineering, College of Engineering, University of Diyala, Baqubah, Iraq

Correspondence

Hamed Al-Raweshidy, Brunel University London, Kingston Lane, Uxbridge, London UB8 3PH, UK.
Email: hamed.al-raweshidy@brunel.ac.uk

Funding information

Brunel University London

Abstract

The efficiency improvement of healthcare systems is a major national goal across the world. However, delivering scalable and reliable healthcare services to people, while managing costs, is a challenging problem. The most promising methods to address this issue are based on smart healthcare (s-health) technologies. Furthermore, the combination of edge computing and s-health can yield additional benefits in terms of delay, bandwidth, power consumption, security, and privacy. However, the strategic placement of edge-servers is crucial to achieve further cost and latency benefits. This article is divided into two parts: an AI-based priority mechanism to identify urgent cases, aimed at improving quality of service and quality of experience is proposed. Then, an optimal edge-servers placement (OESP) algorithm to obtain a cost-efficient architecture with lower delay and complete coverage is presented. The results demonstrate that the proposed priority mechanism algorithms can reduce the latency for patients depending on their number and level of urgency, prioritising those with the greatest need. In addition, the OESP algorithm successfully selects the best sites to deploy edge-servers to achieve a cost-efficient system, with an improvement of more than 80%. In sum, the article introduces an improved healthcare system with commendable performance, enhanced cost-effectiveness, and lower latency.

KEYWORDS

computer network management, computer networks, medical computing, optimal edge-servers/cloudlet placement, priority mechanism

1 | INTRODUCTION

The importance of healthcare has increased worldwide due to its direct relationship with the quality of human life. An effective and robust healthcare system leads to a strong and confident society. However, governments and the public sectors face many challenges due to the rising number of patients year after year, as well as growing epidemiological concerns. These challenges become more serious with the elderly population requiring constant monitoring. As a result, it is becoming increasingly difficult for traditional healthcare systems, which require one-to-one contact between the caregiver and patient, to expand to accommodate the growing patient population [1]. Moreover, the financial burden brought on by administrative and operational expenses has a propensity to lead to system burnout. Thus, smart healthcare (s-health) systems are becoming increasingly necessary to meet these requirements. Furthermore, finding smart, cost-efficient systems that have the ability to remotely monitor and supervise patients is a significant advancement in medical science, offering reasonable solutions to both current and future problems. These systems should provide high-quality medical care at a limited cost to guarantee their long-term viability [2].

S-health is anticipated to make a substantial contribution towards reducing hospitalisation rates and delivering telehealth services to remote patients at a reasonable cost. This is along with other advantages, such as accuracy, scalability, energy efficiency, configurability, and maintainability. However, the question is how to switch from conventional healthcare systems to s-health systems [3].

The transition of conventional healthcare practices to s-health has been expedited by advancements in computational intelligence, mobile communication technologies, and Internet of Things (IoT) technologies [4]. The development of artificial intelligence (AI) will have a great impact by enhancing the intelligence, autonomy, dynamic nature, and adaptability of healthcare systems. These services are predicted to revolutionise healthcare by expediting diagnosis and treatment procedures, lowering the cost of doctor visits, and improving the standard of patient care.

Moreover, combining edge computing with smart health is so versatile that it can provide short delay, save network bandwidth, lower power consumption, as well as deliver security and data privacy benefits [4]. The strategic placement of edge-servers plays a crucial role in achieving these advantages, including reduced deployment cost, minimal latency, load balance etc. That is, the question of where edge-servers should be placed within a network must be considered. Moreover, the edge-servers placement problem (ESPP) becomes much more important when taking into account the utilisation of edge-servers in a wireless local area network.

This article aims to explore the potential of using AI in conjunction with edge computing to enhance healthcare systems and make them more sophisticated. Firstly, an intelligent priority mechanism is proposed to identify critical patients who require urgent medical assistance. The authors build upon their previous HMAN environment [5] as a baseline to achieve better Quality of Service (QoS) and Quality of Experience (QoE) when utilising AI with edge/fog computing in the healthcare sector. The HMAN system [5] treated all data equally, without considering urgency, which could cause delays in treating critically ill patients. By categorising users' data into different classes, based on patients' conditions, it becomes easier to identify and prioritise the most severe cases, enabling them to receive prompt and appropriate services such as sending an ambulance or preparing necessary medical staff. The ultimate objective of this study is to develop intelligent monitoring and healthcare systems for medical facilities that can dynamically regulate patient flow based on each individual health status.

Secondly, strategic placement of edge-servers in the HMAN environment can significantly reduce deployment costs and minimise the overall delay between patients and edge-servers, thus enhancing the performance of healthcare applications. In regions where multiple MCs exist, it is not necessary to deploy edge-servers at each MC due to financial limitations. However, it is essential to ensure that the delay requirements for each task are met. Since all MCs are identical, any MC can be a potential location for placing edge-servers. Nonetheless, a subset of the MCs can be selected to reduce the deployment budget while maintaining an acceptable level of delay. The question that arises now is how to select the best MCs subset to achieve the research objectives.

As human life takes precedence in research pertaining to the design and proposal of healthcare systems, the top research priority is to ensure that health services arrive as quickly as possible with minimal cost. The key objective is to design an intelligent healthcare system while reducing the cost of server deployment and maintaining acceptable latency. The following is a summary of the contributions of this paper:

1) We introduce a priority processing/offloading mechanism based on AI, aimed at identifying critical patients who require urgent medical assistance, thereby enhancing QoS and QoE.
2) We investigate the edge-server placement problem and propose an optimal edge-servers placement (OESP) algorithm, which aims to achieve cost-efficient architecture with lower delay and comprehensive coverage. The main objective is to deploy edge-servers to subsets of MCs within a given area, providing edge health services to monitored patients.
3) Finally, we conduct simulations to evaluate the performance of the proposed algorithms. The results demonstrate favourable outcomes, showcasing a cost-effective system with lower latency when compared to existing algorithms in the literature.

1.1 | Paper organisation

The remainder of this paper is organised as follows: Section 2 reviews the related work; Section 3 presents the proposed priority mechanism; Section 4 introduces the optimal edge-servers problem formulation; Section 5 describes the proposed OESP; Section 6 presents and discusses the simulation results; and finally, Section 7 draws the conclusions along with discussing avenues for future work.

2 | RELATED WORK

As our study can be divided into two parts, we have divided the related work section into two subsections: priority mechanism and edge-servers placement approaches.

2.1 | Priority mechanism approach

In recent years, significant advancements in IoT and body sensor devices, along with the integration of emerging technologies such as edge or fog computing with the cloud, have spurred numerous research projects dedicated to developing IoT-based smart health monitoring frameworks. However, existing healthcare systems, exemplified by studies [6–15] and the baseline paper [5], often adopt a uniform approach to processing and predicting data, following a first-in-first-out concept, without considering the urgency of cases. As a result, patients with severe illnesses may encounter delays in receiving timely treatment. To tackle this challenge, this study proposes enhancements to the HMAN healthcare system [5] by incorporating an AI-based data classification method. The main objective is to establish a priority mechanism based on the urgency of patients, with the aim of improving both the QoS and the QoE in healthcare delivery. By introducing a prioritisation framework that considers the urgency of cases, this approach seeks to ensure that critical patients receive prompt attention and care.

2.2 | Edge-servers placement approach

The placement of edge-servers, commonly referred to as cloudlets, is a crucial factor that significantly impacts the efficiency of a system. However, this topic has received limited attention in the existing literature, with only a few studies addressing it. The key question revolves around determining the optimal placement of edge-servers in a system to maximise benefits while considering different goals, such as reducing costs or minimising latency.

Several papers have explored server placement in large-scale environments, specifically the wireless metropolitan area network (WMAN), and are particularly relevant to our study. Jia et al. [16] investigated how to place a number of cloudlets and allocate users among them in a way that minimises the average system response time. To address this issue, they also suggested the density-based clustering approach. To reduce the average cloudlet access delay, Zhao et al. [17] suggested using a ranking-based heuristic for K cloudlets deployment. In order to reduce the latency between the users and cloudlets, Xu et al. [18] proposed an exact solution to the problem by placing K cloudlets in strategic locations within a large-scale WMAN.

Similarly, a cost-aware cloudlet deployment technique was suggested by Fan et al. [19] to improve the trade-off between deployment cost and latency. To minimise communication latency, Meng et al. [20] proposed some algorithms for deploying cloudlets in a group of access points and routing mobile task requests. Firstly, they derived an approximation algorithm to choose candidate locations based on historical data. Then, they presented an iterative algorithm to select the best locations to deploy cloudlets in. Finally, an online job routing algorithm was proposed to route the request to the cloudlet with minimum latency. Yao et al. [21] proposed a low-complexity heuristic algorithm to cost-effectively deploy cloudlets without compromising a pre-determined QoS. They essentially assumed that the cloudlet servers are heterogeneous, meaning that they have different cost and resource capacities. A novel framework named EdgeON was presented by Santoyo-Gonzalez et al. [22], which was aimed at reducing the total cost when placing and operating the edge-servers network. To reduce the average response time, Li et al. [23] suggested two methods for edge-servers placement: flat and hierarchical. They found that the hierarchical approach has a better performance in reducing system response time. A different solution by Li et al. [24] presented the problem of an energy-aware edge-servers problem as a multi-objective optimisation one. Then, they suggested a particle swarm-based energy-aware algorithm for reducing energy consumption in computing resource utilisation.

There are also other studies in edge-servers placement. The edge-server placement problem was formulated, firstly, by Lahderanta et al. [25] as a constrained optimisation mode to reduce the sum of weighted distances between the edge-servers and access points. Then, they designed the PACK algorithm to minimise the distances while balancing the load among servers. Lovén et al. [26] proposed a new algorithm to choose the optimal number of edge-servers to be placed and optimally reallocate access points, accordingly, in order to improve QoS.

In addition to the aforementioned studies, recent papers have made significant contributions to the field of edge-server placement. For instance, [27] proposed novel approaches for addressing the challenges in deploying edge servers. Their work focused on optimising the placement of edge servers in order to minimise latency and enhance network performance. Through innovative algorithms and techniques, they achieved improved efficiency and effectiveness in edge-server deployment, paving the way for further advancements in this area. In [29], cloudlet deployment in IoT networks was investigated with the aim of optimising deployment cost and network latency. The authors proposed a fault-tolerant cloudlet deployment scheme based on software-defined network technology. Their binary-based differential evolution cuckoo search algorithm showed promising performance in terms of cost and latency optimisation. Lastly [29] focused on deploying edge servers effectively and economically in wireless metropolitan area networks. They addressed the problem of minimising the number of edge servers while ensuring specific QoS requirements by extending the definition of dominating set and formulating it as a graph theory problem. Their proposed greedy-based algorithms showed feasibility and effectiveness in achieving efficient edge-server deployment. These papers contribute to the advancement of edge-server placement and offer valuable insights for future research in this area.

To summarise, the reviewed research studies present innovative methods for placing edge-servers, showcasing notable achievements in terms of cost and/or time. The literature has explored four main perspectives regarding edge-server placement:

1. Minimising response time (latency) by employing various distance measures.
2. Minimising the cost of server deployment while maintaining a maximum acceptable delay.
3. Optimising the trade-off between delay and costs.
4. Maximising coverage, that is, the number of users served.

In comparison, this study introduces a novel approach that involves placing a flexible number of servers to minimise costs while ensuring acceptable latencies based on patient conditions through a priority mechanism. It presents a fresh perspective on selecting and locating edge-servers within a specific area, considering different variables such as the number of connections between competing sites, distances, and historical workloads. Furthermore, this study stands out in the literature as the first to address the problem of edge-server placement specifically in the healthcare domain.

3 | PRIORITY MECHANISM BASED ON ARTIFICIAL INTELLIGENCE

This work involved the creation of an intelligent healthcare system for data classification of patients, which would increase data collection effectiveness and streamline processing. By prioritising data based on the urgency of patients, we can achieve better QoS and QoE. The proposed model, illustrated in Figure 1, comprises two main levels: the patient level (IoT layer) and the local MC level (Edge/Fog Layer).

The patient level (a smartphone or watch) has two functions: data collection and filtering (preprocessing). At the local MC level, the arriving data is processed through three steps: classification, prioritisation, and decision-making.

At the patient level, a wearable device (smartphone or watch) collects data and performs initial data preprocessing and analysis (i.e. aggregation, fusion, filtering, and classification). As a result, based on a predetermined threshold assigned according to the patient's condition, the collected data are classified as normal or abnormal. While the abnormal data is offloaded to the next layer for additional actions, the normal classified data are temporarily stored locally without requiring any further action.

At the local MC level, the received data undergoes three steps to determine the appropriate course of action. Firstly, an AI method, such as machine learning or deep learning, is employed to classify the data into multiple categories enabling the determination of the level of risk associated with each patient. Subsequently, a priority processor assigns a priority level to each patient based on their historical record, which is stored in a database. Finally, a decision is made regarding whether the data should be processed locally or offloaded to other units based on their priority levels. For example, if the local MC receives a simultaneous influx of 100 patients, but can only provide medical services for 50, it prioritises the first 50 patients on the list, while the remaining patients are offloaded to other units within the architecture using the HOSSC algorithm outlined in ref. [5]. Algorithm 1 represents the proposed priority processing/offloading scenario, which dynamically updates patient lists based on their priority levels. This ensures that the most urgent cases receive medical attention at the local MC, while other cases are handled by other units in the architecture.

Algorithm 1 Priority processing/offloading algorithm

```
Input: λsum, λmax
Output: Priority mapping patient offloading
For a given situation at a particular time
instant t the following will be done.
If λsum ≤ λmax
  All patients can be served by the local MC
  else % The priority mechanism will be
  turned on
  λsum is classified into classes by the AI
classifier
  ClassA → Priority = 0 //Normal cases,
  Discarded
  ClassB → Priority = 1 //Mild case
  ClassC → Priority = 2 //Moderate case
  ClassD → Priority = 3 //Severe case
  ….
  …. // (if needed depending on the type of
disease)
```

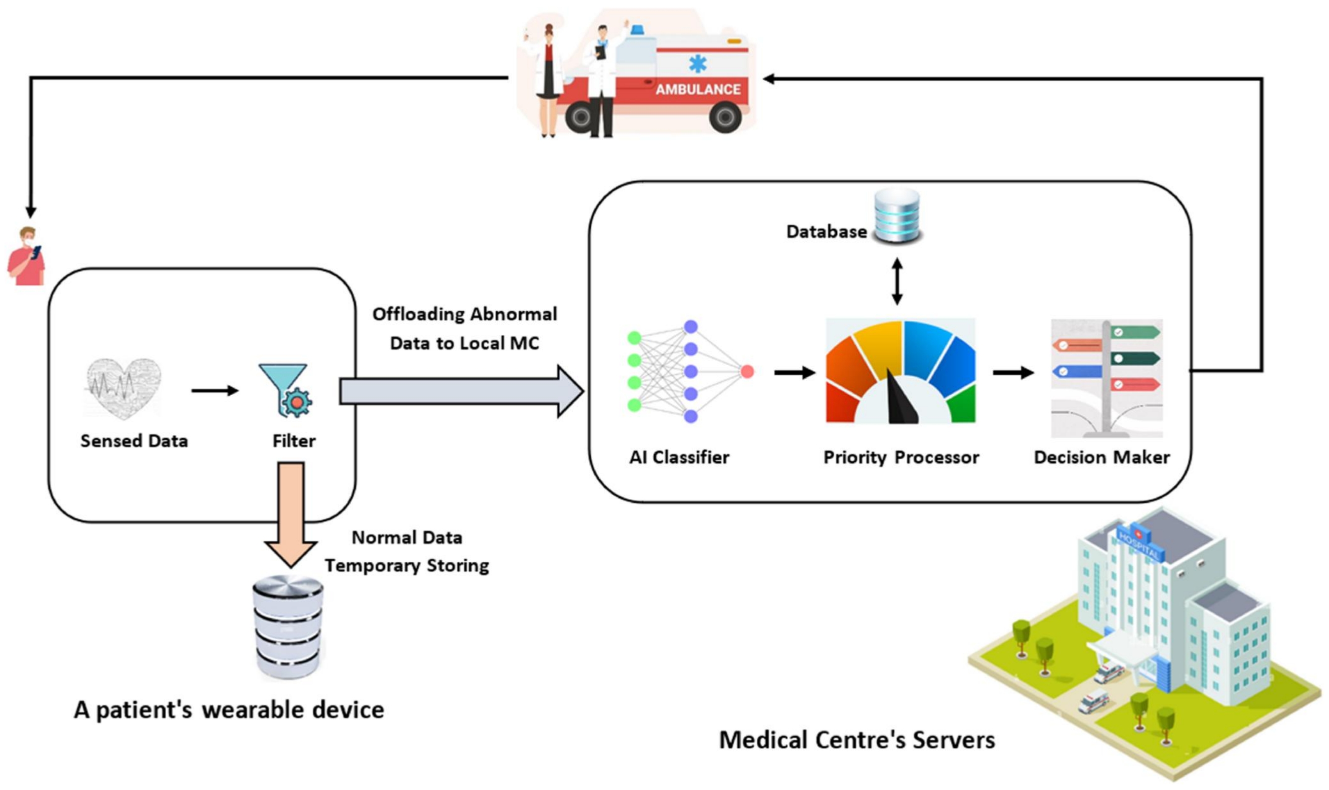

FIGURE 1 Model of the proposed priority mechanism.

```
  SortedPatientsMatrix = [a high_priority_
  patient … a less_priority_patient]
  Then:
  The Local MC will serve patients based on
  their order in SortedPatientsMatrix
end if statement
```

4 | OPTIMAL EDGE-SERVERS PROBLEM FORMULATION

Our goal is to effectively deploy edge-servers to specific MCs within a region to meet the needs of all patients under monitoring. We represent the area as a two-dimensional space (grid) in which patients (IoT devices) and MCs may coexist. The patients can be everywhere in the area.

For a simple representation of our system, we made a few of assumptions. The first is that all MCs inside the grid are candidates to be placed with edge-servers. Next, we assume that all MCs are connected virtually with each other, if and only if, the d (MCi,MCj) = K, in which, d (a,b) is a distance function for calculating the distance between two points (a and b), and K is the largest possible distance to connect the neighbours (MCs). The last assumption is that all MCs have different volume and distribution of user requests (historical loads) over a long period.

The set of MCs (candidate points) Is, hence, defined as MCs = {mc1, mc2, …; mcn}, where each refers to a pre-selected, feasible placement location in the grid (in coordinate axes) and noMCs = number of MCs in that area. A set of edge servers is denoted by ES (es1, es2, …; esk) and Nes = number of edge-servers needed to be placed for full coverage. We assume all edge servers are identical (homogeneous). A set of patients requiring edge computing services is denoted by p = (p1, p2, …; pm), where m = number of patients. A cost function Cost (esj, mck) is defined to refer to the incurred cost of placing an edge server es ∈ ES at an MC mck ∈ MCs.

We define two latency functions, Lij and D, that represent the latency when patient pi is served by edge servers placed at mcj of the grid and the latency between two edge servers respectively. We assume a homogeneous bandwidth across the grid in our model and as a result, latency is predominantly affected by distance. Our main objectives are to reduce the cost of deploying edge-servers in the region and reduce the latency in accessing edge services with full coverage to all patients.

We now formulate the ESPP as a multi objectives optimisation model. We define the following decision variables:

- X represents the placement of edge-servers at MCs

$$X = \left\{ x_j \,\middle|\, 1 \le j \le n \right\}$$

where:

$$x_j = \begin{cases} 1 \text{ if } es_i \text{ placed at } mc_j \\ 0 \quad \text{otherwise} \end{cases}$$

- Y represents the assignment of patents to MCs

$$Y = \left\{ y_{ij} \,\middle|\, 1 \le i \le m, 1 \le j \le n \right\}$$

where:

$$y_{ij} = \begin{cases} 1 & \text{if } \mathrm{p}_i \text{ is assigned to } \mathrm{mc}_j \\ 0 & \textit{otherwise} \end{cases}$$

- E represents the links between MCs

$$E = \{ e_{ab} \,|, 1 \le a, b \le n, a \ne b \}$$

where:

$$e_{ab} = \begin{cases} 1 & \text{if } \mathrm{mc}_a \text{ and } \mathrm{mc}_b \text{ are directly connected} \\ 0 & \text{otherwise} \end{cases}$$

Let λ_{sum} be the set of task arrival rate of patients,

$$\lambda_{sum} = \sum_{1=1}^{m} \lambda_i \tag{1}$$

The SRT is calculated based on [5]:

$$SRT = \frac{\sum_{i=1}^{n} t_{pi}}{n} \tag{2}$$

where:

$$\begin{aligned} t_{pi} = T_{wi} + E_{AP} * T_{AP} + (c_1 + 1) * t_{GP} + c_1 * V \\ + c_2 * S + c_2 * t_{GH} + c_3 * L + c_3 * t_{GH} + c_4 * B \\ + c_4 * t_{\mathrm{Cloud}} \end{aligned} \tag{3}$$

The ESPP is defined as follows. Given a system model parameter (G, noMCs [points], K, Nes, m, L, D, λMAX), the problem is to find X (the placement of edge-servers) among the MCs, such that the system response time SRT is minimised, that is,

$$\min \sum \mathrm{Cost}(esj, mck) \;.\; Xj \tag{4}$$

$$\min \mathrm{SRT} \tag{5}$$

$$\min \sum E \tag{6}$$

Subject to:

1) $N_{es} >= 3$: to ensure that at least three edge-servers are placed at 3 MCs to preserve previous achievements in [5].

2) $d(mc_a, mc_b) = k$: to ensure a shorter distance between a patient and an MC and to avoid the colocated problem.
3) $\sum E_{ij} = 1 \; or \; 2$: to ensure that a patient either connects directly with an edge-servers site or through only 1 MC to minimise the latency.
4) $\sum y_{ij} = 1 \quad (1 \leq j \leq n)$: to guarantee that all patients must be served, and each is served from exactly one candidate point.

5 | OPTIMAL EDGE-SERVERS PLACEMENT (OSEP)

The main idea is how to determine the best MC locations for deploying edge servers in order to provide services to all patients in a given area. The number of selected MCs depends on the total number of MCs in a certain area, that ensure that services are provided to all patients, and how they are distributed and connected to each other. This algorithm is built based on several stages, as follows.

Step 1: Locate all coordinates on the map.
- This step involves identifying and recording the coordinates of all relevant points on the map.
- Ensure that the map is valid and that there are no repeated points.

In this step, the algorithm focuses on locating and recording the coordinates of all relevant points on the map. It is important to ensure that the map is valid, meaning it accurately represents the desired area or region, and that there are no repeated points. The uniqueness of points is crucial to prevent any duplication or confusion during subsequent stages of the algorithm.

Step 2: Generate important matrices:
1) Distance Matrix (DistancesBtAll): Generate a matrix that represents the distances between all pairs of points. This matrix helps quantify the proximity between points.
2) Connectivity Matrix (ConnectivityMatrix): Generate a matrix that indicates the connectivity between points, where 1 represents a connection and 0 represents no connection.

After that, establish virtual connections between neighbouring points based on a specified maximum distance threshold (Max_DistanceToConnect).

Step 3: Finding and selecting the first, second and third best points.

In this step, the algorithm evaluates points based on multiple criteria, including the number of links per point (SumLinks), the sum of distances to other points (SumDistances), and historical loads (Load_History). The objective is to select the best points that ensure connectivity to the highest possible number of points.

The algorithm generates a matrix of points sorted based on these three criteria. The primary criterion is the number of links per point (SumLinks), which prioritises points that can establish connections with the greatest number of other points. In cases where two or more points have the same number of connections, the algorithm applies the second criterion, which is the sum of distances to other points (SumDistances). This criterion focuses on selecting points that are closest to other points in terms of distance. If a tie persists even after considering the second criterion, the algorithm resorts to the third criterion, which is historical loads (Load_History).

By sorting the points in the matrix according to these criteria, the algorithm aims to identify and select the first, second, and third best points. To ensure optimal connectivity, it is important to consider the following conditions for the second and third best points:

1. Direct Connection to the First Best Point: The second and third best points must be directly connected to the first best point, establishing a direct link between them.
2. No Connection between Second and Third Best Points: The second and third best points should not be connected to each other. This arrangement ensures that the second and third best points are chosen on both sides of the first best point, maximising the number of points connected on both sides.

By adhering to these conditions, the algorithm guarantees that the selected points promote effective connectivity and facilitate the connection of as many points as possible to the three best points in the network.

Step 4: Test Connectivity 1

In this step, the algorithm performs a connectivity test to evaluate the network's current state and identify connected and disconnected points. The goal is to determine if all points are successfully connected or if there are any points that remain unconnected.

The algorithm carries out the following actions:

1. Determine the Connected Points.
2. Determine the NotConnected Points.

By analysing the connectivity status of the points based on the previous selections, the algorithm can assess the network's current state. If all points are connected, indicating that every point is successfully linked to the network, the algorithm terminates as it has achieved its goal. However, if there are still unconnected points, the algorithm proceeds to the next stages to address and resolve the connectivity issues.

Step 5: Identify Points Unreachable for Connection (Distant Points) and Update the Unconnected Ones.

In this step, the algorithm focuses on identifying points that are too far away to establish connections with other points in the network. These distant points are unlikely to be reachable and cannot be effectively integrated into the current network configuration. The algorithm subsequently updates the list of unconnected points based on this assessment.

This step is significant as it helps determine which points are geographically distant and cannot be connected. By isolating these distant points, it becomes possible to consider alternative strategies, such as integrating them with other working regions or adjusting the network configuration to accommodate their unique circumstances.

Step 6: Determine the Other_Best points from the Connected points.

In this step, the algorithm identifies additional best points from the pool of connected points, one by one, to further enhance the network's connectivity. After each selection, the algorithm updates the list of unconnected points based on the newly established connections. The number of these additional best points is not fixed, but rather, a sufficient number should be chosen to achieve full network connectivity while ensuring that the previously mentioned conditions and constraints are met.

Step 7: Test Connectivity 2

In this step, the algorithm performs a second connectivity test to evaluate the network's current state and ensure that all points, except those previously determined as unreachable, are connected. The goal is to verify that the number of disconnected points is zero, indicating successful network connectivity. If there are still disconnected points remaining, the algorithm repeats Step 6 to further enhance the connectivity. Algorithm 2, the OESP algorithm, represents the proposed algorithm for optimising the placement of intelligent edge-servers in the healthcare field.

Algorithm 2 Optimal Edge_Servers placement (OESP) algorithm—pseudo code

```
Input: noMCs, X = {xi, 1 ≤ I ≤ noMCs} , Y = {yi, 1 ≤ I ≤ noMCs}, Max_DistanceToConnect,
Load_History
Output: Find the best points (Best_Points)
Constraints: Best_Points ≥ 3, no. hops to the Bests = 1
% Step 1: Determine the coordinates
% Step 2: Generate important matrices:
index ← 1
for k ← 1 to noMCs
  for m ← 1 to noMCs
    DistancesBtAll(index) ← point k—point m
    if DistancesBtAll(index2) ≤ DistanceToConnect then:
      ConnectivityMatrix(k,m) ← 1
    else
      ConnectivityMatrix(k,m) ← 0
    end if statement
    index++
  end for statement
end for statement
%% Step 3: Find Best three points
  - Create: Point_Features ← [SumLinks SumDistances Load_History]
  - Sort: Point_Features ← [SumLinks(desc) SumDistances(asc) Load_History(desc)]
a- Find: First_ Best ← First member of Point_Features (sorted).
b- Find: Second Best
for I ← 2 to noMCs then
  if Point_Features (I,1) is connected to First_Best_Point
    Second_Best ← Point_Features (I,1)
    break for loop
  end if statement
end for statement
c- Find: Third Best
  if Point_Features (I,1) in connected to First_Best_Point AND not connected to
Second_Best_Point
    Third_Best ← Point_Features (I,1)
    break for loop
  else
    Third_Best ← Point_Features (I,1);
end if statement
% Step 4: Test Connectivity 1
i ← 1 , k ← 1
for j = 1:noMCs
  if j ~ = First_Best AND j ~ = Second_Best AND j ~ = Third_Best
    if ConnectivityMatrix(j,First_Best) ~ = 1 AND ConnectivityMatrix(j,Second_Best) ~ = 1
AND
    ConnectivityMatrix(j,Third_Best) ~ = 1
      NotConnected(i) ← j
      i++
    else
      Connected(k) ← i
      K++
    end if statement
  end if statement
end for statement
% Step 5: Find Never connected Points (Distant points) → NeverConnected
k ← 1
for i ← 1 to no_Notconnected
  for j ← 1 to no_connected
    if ConnectivityMatrix(i,j) = = 0
      NeverConnected(k) ← i
NotConnected(i) []←
      K++
  end for statement
end for statement
% Step 6: Find other Best point (if needed)
while 1
  if no_NotConnected ≥1
    k = 1
    for s ← 1 to no_Connected
      if s is connected to NotConnected
        Current_Best(k) ← s % This matrix identifies the candidate point that has the
potential to be among the best points.
        k++
      end if statement
    end for statement
    k = 1
    for i ← I to no_NotConnected
      for j ← 1 to no_Current_Best
        if ConnectivityMatrix(I,j) = = = 1
          Choose a one with a shorter distance.
          Other_Best(k) ← j (with a shorter distance)
          k++
        end if statement
      end for statement
    end for statement
  else
    break while loop
  end if statement
end while statement
% Stage 7: Test Connectivity 2
i← 1 ,k ← 1
for j = 1:noMCs
```

```
  if j ! = NeverConnected
    if j ! = First_Best AND Second_Best AND Third_Best
      if ConnectivityMatrix(j,First_Best) ~ = 1 AND ConnectivityMatrix(j,Second_Best)
~ = 1 AND
      ConnectivityMatrix(j,Third_Best) ~ = 1 AND ConnectivityMatrix(j,Other_Best)
        NotConnected(i) ← j , i++
      else
        Connected(k) ← i , k++
end if statement end if statement
    end if statement
  end for statement
If no_NotConnected = = 0
  Print("ALL Points are connected Except the Never Connected one because they are far way")
  Display(NeverConnected)
  else Repeat Step 6.
end if statement
```

TABLE 1 SRT simulation parameters.

Symbol	Parameter	Value
Ps	Number of patients	100, 200, 300
n	Number of servers in each GP	5, 6, 7
m	Number of servers in each GH	10, 12, 14
r_{Wi}	Link rate between IoT device and AP	54 Mbps
r_{AP}	Link rate between AP and the local GP	100 Mbps
r_{Fib}	Link rate between the GPs and GHs	Up to 10 Gbps
λi	Packet size	30 KB
T_{wi}	Wireless delay	4 ms
T_{AP}	Delay between AP and the local GP	2 ms
V	Delay between two neighbouring GPs	$(1 - K_1).\boldsymbol{\lambda}_{\text{GPsum}}/10$ Gbps
S	Delay between the GPs and GHs	$K_2.\boldsymbol{\lambda}_{\text{GHsum}}/10$ Gbps
L	Delay between two neighbouring GHs	$(1 - K_2).\boldsymbol{\lambda}_{\text{GHsum}}/10$ Gbps
B	Internet delay	20 ms
μGP/μGH	Each GP/GH server service rate	100/200 KB per ms
μ_{Cloud}	Cloud service rate	1000 KB per ms
λ_{GPmax}	Maximum GP workload	200 × n KB
λ_{GHmax}	Maximum GH workload	200 × m KB

6 | RESULTS

In this section, we randomly create data to evaluate the performance of the proposed algorithms. Table 1 summarises the simulation parameters to calculate the SRT, which represents the average system latency required to deliver patient data to one of the health facilities within the HMAN architecture [5]. Additional parameters will be introduced later.

6.1 | Priority mechanism based on artificial intelligence

The initial step in processing the collected data involves wearable devices worn by patients to filter the data and transmit any abnormal data to the local MC for further processing. Upon reaching the local MC, priority is given to patients with the most severe conditions. In our simulation, three scenarios were conducted to evaluate the performance of the proposed priority mechanism. To assess the effectiveness of the algorithm, a comparison was made between the SRT for accessing the service with and without the application of the algorithm.

The first scenario assumed that 100 patients (data or workload) arrived at the local MC at the same time. The first step is to check whether the local MC is capable of handling the received data or not. If it can, the data is processed at the local MC and the services are delivered to the patients without needing to send out the data to other units. However, if it is determined that the MC cannot handle the data of all 100 patients, then it must offload a part of it to other units based on the HOSSC algorithms in ref. [5]. The main task of the proposed priority mechanism is to determine which patients are served by the local MC and which ones should be offloaded. In this scenario, the algorithm classified the received data into four classes and then, served the patients accordingly. Patients with the highest priority and greatest need are served in a shorter time compared to those with less severe conditions. The same approach is applied in the 200 and 300 patient scenarios, where the algorithm categorises the data and prioritises the patients accordingly.

It is worth noting that the absence of a priority mechanism, the timing of service provision to patients would be uncertain since the HMAN architecture does not distinguish between patients. Table 2 shows the results obtained from applying the above scenarios and demonstrates the effectiveness of the proposed algorithm in providing services with less delay to the patients with the highest level of urgency. In the case of HMAN [5], it is not possible to determine which patient should be treated first, because the system handles data equally without taking urgency into account. This could potentially have negative consequences on the treatment of patients with severe conditions.

6.2 | Optimal edge-servers placement algorithm

In this section, we report the conducting of several simulations to evaluate the performance of the suggested algorithms. We investigated the algorithm on randomly generated points. Every point represented an MC site. A number of parameters were defined for the OESP algorithm, as listed in Table 3. To demonstrate the effectiveness of our algorithm, we started by applying 10 randomly generated points and explained the steps in detail to reach the result of finding the best points to place edge-servers in. Subsequently, we present the final results obtained when applying the algorithm to 20 and 30 randomly generated points.

- ***Scenario 1 (10 nodes)***

Firstly, a MATLAB code generated 10 points in a geographic area of 100 km × 100 km. Every point was considered as an MC site, as mentioned before. Then, these points were assigned to their respective locations on a virtual map, as depicted in Figure 2.

The subsequent task involved establishing virtual connections between these points based on the first constraint, which specified a maximum distance of 40 km to connect

TABLE 3 Optimal edge-servers placement simulation parameters.

Symbol	Definition	Value
A	The region dimensions	100 km × 100 km
noMCs	Number of points (MCs)	10, 20, 30
x, y	Coordinates	Randomly created
D_{max}	Max. Distance between two neighbours	40 km

TABLE 2 SRT with each class with/without applying the proposed priority mechanism.

	Classes		SRT ($n = 5$, $m = 10$)	
No. patients	Type	No. patients	No priority mechanism (HMAN [5])	With priority mechanism
100	A	3	N/A	Discarded
	B	61		18–19 ms
	C	28		8–9 ms
	D	8		5–6 ms
200	A	5	N/A	Discarded
	B	103		25 ms
	C	63		16–17 ms
	D	29		7–8 ms
300	A	4	N/A	Discarded
	B	181		32 ms
	C	73		19–20 ms
	D	42		9–10 ms

two points. As depicted in Figure 3, there were multiple redundant links between certain points. The objective of the proposed algorithm was to minimise the number of these connections while ensuring that the entire network remained connected.

The next step involved determining the best three points based on the number of links per point, proximity to other points, and the historical loads per point, which were assumed randomly in this simulation. Referring to Table 4, which ranks the point according to these three parameters, it is clear that the point 4 is the first-best one, because it has more links than the others. The second-best point is 10, as it is directly connected to point 4 and has more links compared to the remaining points. In order to enhance network coverage, the algorithm disregarded point 2 (or point 9) as the third-best point, despite its higher rank compared to the others. This decision was based on the fact that point 2 (or point 9) shares a direct link with point 10, causing them to fall on the same side of the best point. Instead, point 5 was chosen as the third-best point, fulfilling the specified conditions and situated on a distinct side from the best point. After choosing these three best points, the algorithm establishes connections between all the other points and its closest best point while removing all other links. Figure 4 depicts the network after applying the previous steps. Note that the point 8 remains unconnected as it is located more than 40 km away from all the best points. Therefore, the next algorithm task will be to identify other best points to ensure complete connectivity.

Before moving on to the next step, it is crucial to highlight that the algorithm generated four matrices that depict the current state of the points within the network. These matrices are as follows:

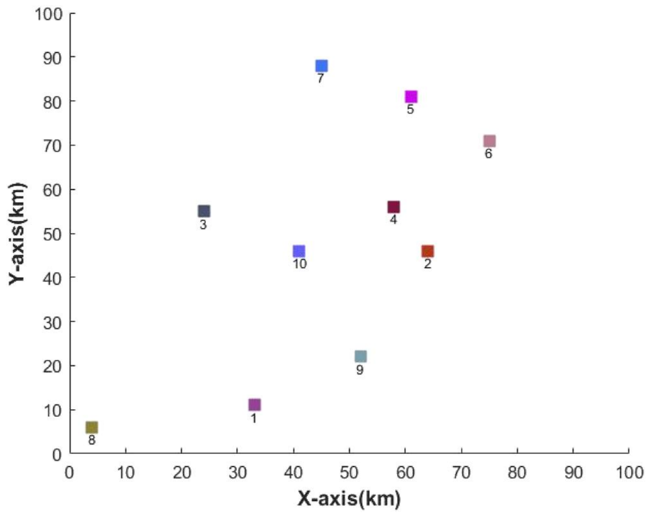

FIGURE 2 10 disconnected points (MCs) on a map.

TABLE 4 Sorted 10 points (Ms) based on the number of links per point, how close the point is to all others, and the historical loads per point.

Points	Number of links	Distance to all	Historical loads
4	7	307.3874	711
10	5	303.4536	211
2	5	329.8995	459
9	4	383.7702	253
5	4	409.7863	602
6	4	420.9528	718
7	4	449.8674	847
3	3	373.1516	393
1	3	456.99	424
8	1	615.2826	879

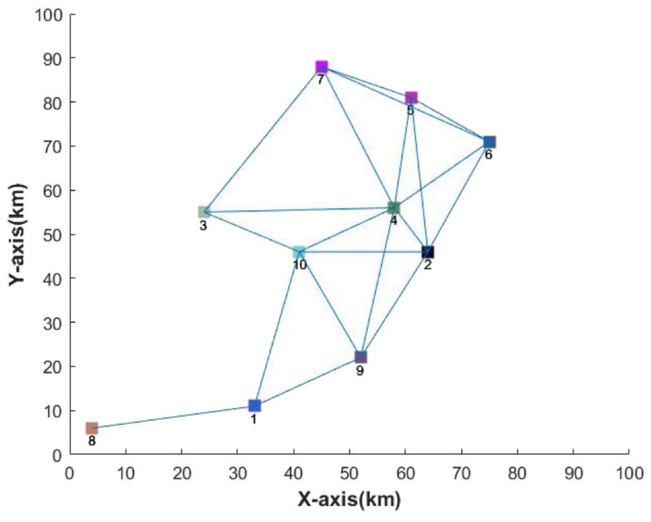

FIGURE 3 A virtual network of 10 connected points (MCs) based on distances between neighbours ($D_{\max}$ = 40 km).

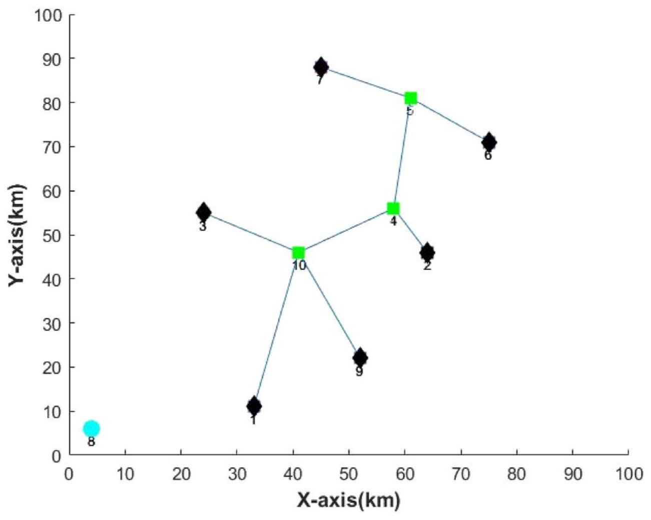

FIGURE 4 A network of 10 points after choosing the first, second, and third best points (MCs): Green points = Best points, Black points = connected points and Cyan = Not-connected points.

- Best = [4 10 5]
- Connected = [2 9 6 7 3 1]
- NotConnected = [8]

The subsequent step involves determining which points should be selected to be the best ones. The algorithm does not impose any limitation on the number of these points. The primary objective is to achieve full network connectivity while minimising costs and adhering to all constraints. The remaining unconnected points need to be checked for the possibility of connecting them or not. If the distance between an unconnected point and any connected points is less or equal to the maximum distance (constrain 1), then this point can be connected. Subsequently, the algorithm selects the closest Connected point to it and includes this Connected point among the Other-Best points. Conversely, if this condition is not met, it is isolated from the group and marked as a never-connected point. Figure 5 illustrates that point 8 is connected to point 1 which is selected as an Other-Best point.

After completing this step, the matrices have been updated as follows:

- Best = [4 10 5]
- Other_Best = [1]
- Connected = [2 9 6 7 3]
- NotConnected = Ø
- Never_Connected = Ø

With the current configuration, all points are connected, and no points remain unconnected. This indicates that complete coverage has been achieved in the network, fulfilling the objective of the algorithm.

Upon observing Figure 5, it becomes evident that there is a noticeable difference between the network configuration before and after the algorithm was applied. The algorithm was able to successfully select 4 points out of 10 to achieve

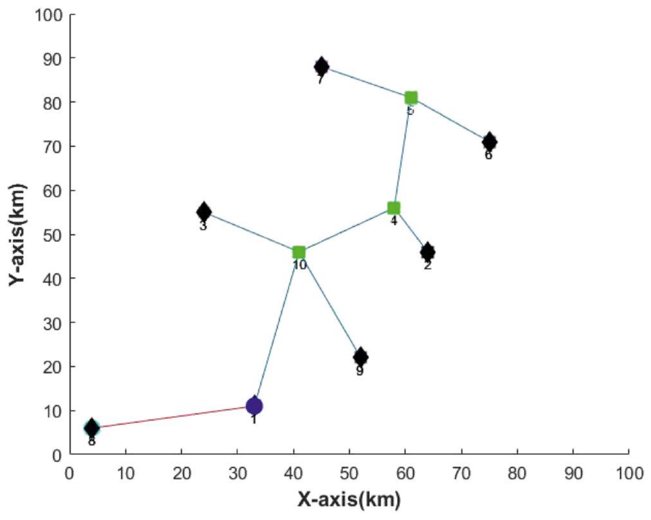

FIGURE 5 The final network of 10 points: Green points = Best points, Blue pints = Other best points, Black points = connected points.

complete network connectivity. Each point is now within one hop distance from the nearest Best point and with not more the maximum distance constraint. Through this process, we have achieved several goals, including reducing cost and delay. Instead of deploying servers at all 10 points, they are now strategically placed at only four points, leading to a significant reduction in the number of connections required. Additionally, the algorithm has successfully reduced latency to the greatest extent possible by considering the distance and number of hops required to reach the servers. By strategically placing the servers at selected points, the algorithm ensures that data transmission distances are minimised, resulting in reduced latency. This optimisation of network infrastructure contributes to improved overall performance and responsiveness of the system.

- *Scenarios 2 and 3 (20 and 30 points respectively)*

To demonstrate the effectiveness of the proposed algorithm, Figures 6 and 7 show two different scenarios involving the selection of the Best points to position edge-servers among a pool of 20 and 30 points respectively. Figures 6 and 7 b present the final shape of the networks of 20 and 30 points after applying the algorithm. The initial steps of the algorithm involve the identification of the Best three points and assessing the network's connectivity. Subsequently, the algorithm proceeds to select additional Other-Best points to attain complete network connectivity, while adhering to predefined constraints.

The crucial point that needs to be emphasised, and can be clearly seen when examining the results, is that the selection of the best points and determination of their number depends on the shape of the network and the proximity of the points to each other. There is no fixed method or specific number of points that must be chosen in all cases. For instance, in the 20-point networks, seven points were selected as the best points, while the 30-point network only required five points to be chosen as the best for full connectivity (see Figures 6 and 7). Furthermore, this algorithm ensures minimal delay in accessing services. Users can directly connect to an MC with edge-servers or through an MC that is one hop away from the edge-servers. By applying this algorithm, we have achieved several benefits, including a significant reduction in cost while maintaining low latency. The proposed algorithm ensures that the delay, which was a key achievement of the HMAN [5] system, is preserved. Figure 8 illustrates the comparison of latency before and after implementing the algorithm, demonstrating that the algorithm successfully maintains the low delay achieved by the HMAN system. This preservation of low delay ensures efficient access to services for users. As a result, the suggested algorithm not only reduces costs but also minimises latency, enhancing the overall performance and efficiency of the network.

Table 5 provides a comparison that highlights the cost difference' between the HMAN system [5] before and after the application of the proposed algorithm. It is worth noting that these percentages may vary across different networks, as they are influenced by the number of points selected as the Best points. This selection process is dependent on the shape of the

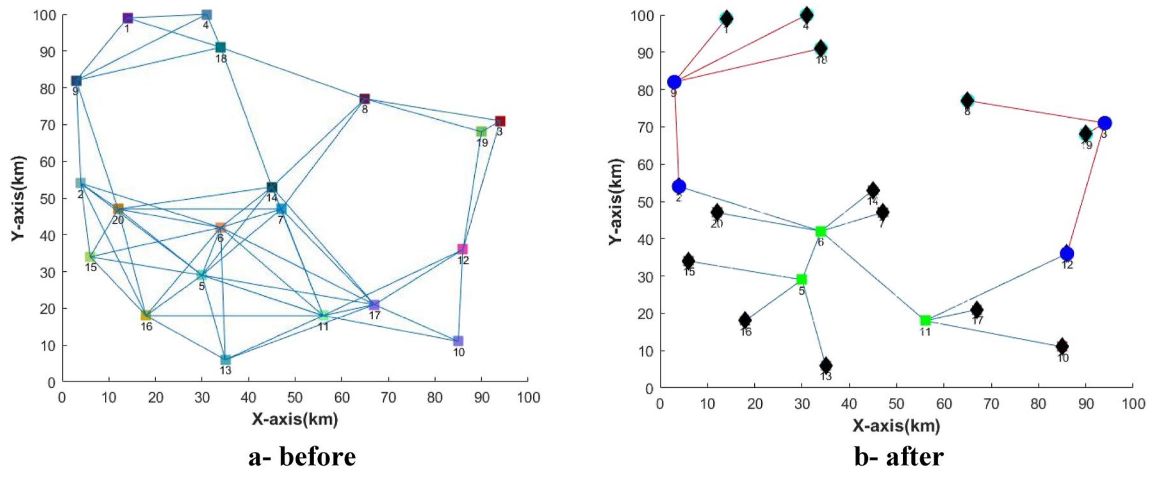

FIGURE 6 A network of 20 points before (a) and after (b) choosing the best points (MCs): Green points = Best points, Blue pints = Other best points, Black points = connected points.

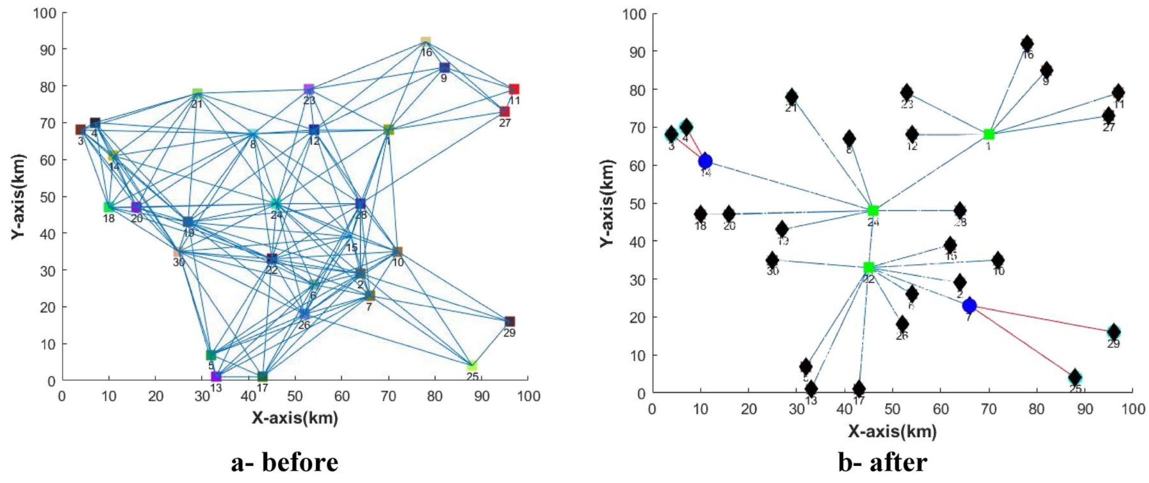

FIGURE 7 A network of 30 points before (a) and after (b) choosing the best points (MCs): Green points = Best points, Blue pints = Other best points, Black points = connected points.

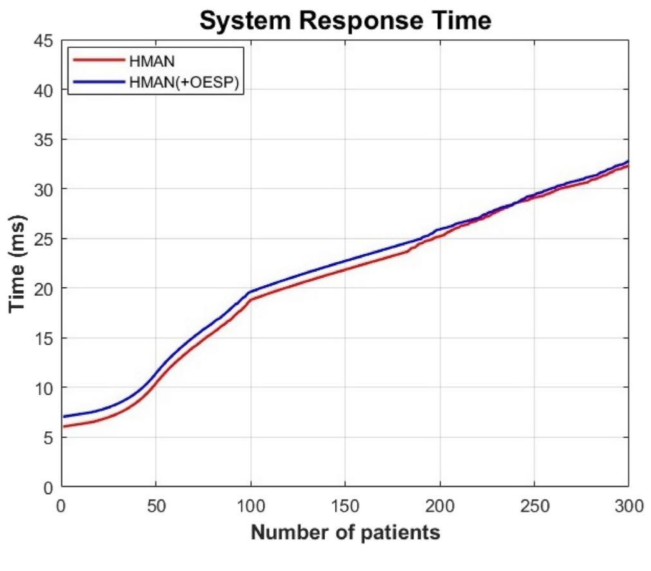

FIGURE 8 A comparison of latency before and after implementing optimal edge-servers placement (OESP) algorithm on HMAN architecture.

network on which the algorithm is to be applied, as explained earlier. However, in all cases, there is a significant improvement in cost while preserving the previously achieved benefits outlined in ref. [5].

In short, this study introduces a novel algorithm for edge server placement in health monitoring frameworks. By considering the shape of networks and proximity of nodes, the OESP algorithm overcomes limitations observed in existing literature reviews. Unlike previous approaches, such as refs. [27–29], that rely on network size for edge server selection, it offers a more robust and tailored solution. In addition, one significant advantage of this study is the careful selection of sites, ensuring that they are located within one hop from the connected sites. This strategic placement of edge servers minimises latency and contributes to the overall effectiveness of the proposed algorithm. By reducing the distance and number of hops required to reach the servers, the latency is kept to a minimum, resulting in improved performance and a seamless user experience. The findings of this research

TABLE 5 A comparison between HMAN and optimal edge-servers placement (OESP) cost.

noMCs	No. of edge-server nodes in HMAN	No. of edge-server nodes in OESP	Deployment cost
3	3	3	0
5	5	3	−40%
10	10	3–5	−(50–70)%
15	15	5–7	−(53–66)%
20	20	5–9	−(55–75)%
30	30	5–12	−(60–83)%

contribute to advancing the field of edge computing in healthcare systems, opening avenues for further exploration and optimisation in this domain.

7 | CONCLUSION

The primary motivation of s-health is to contribute to reducing hospitalisation rates, while providing affordable telehealth services to remote patients. By integrating s-health with edge/fog computing, additional benefits, such as reduced delay and power consumption, network bandwidth savings, as well as improved security and data privacy can be achieved. However, a key challenge lies in determining the optimal placement of edge-servers in a cost-effective manner while ensuring full coverage for all patients with minimal latency. In this paper, two algorithms have been proposed with the aim of providing an efficient priority offloading/processing mechanism and solving the edge-server placement problem. The simulation results have shown that the two proposed algorithms are highly promising. The priority mechanism algorithm successfully classified patients based on the severity of their disease and prioritised their services accordingly. On the other hand, the Optimal Edge-Server Placement (OESP) algorithm effectively identified optimal locations for deploying edge-servers, achieving objectives such as cost reduction with minimal delay. Although the proposed algorithms showcased promising results in improving the efficiency and effectiveness of edge-server placement and priority offloading/processing, further research is needed to address areas such as load balancing and resource allocation for fully optimising network performance. In summary, the combination of s-health, edge/fog computing, and the proposed algorithms offers a comprehensive solution for delivering cost-effective and efficient telehealth services. This research opens up new avenues for improving healthcare accessibility, reducing costs, and enhancing patient care through advanced technologies and intelligent algorithms.

AUTHOR CONTRIBUTIONS

Ahmed M. Jasim: Conceptualisation; data curation; formal analysis; methodology; resources; software; validation; visualisation; writing – original draft preparation. **Hamed Al-Raweshidy**: Funding Acquisition; investigation; project administration; resources; supervision; validation; writing – review & editing.

ACKNOWLEDGEMENTS

Brunel University London.

CONFLICT OF INTEREST STATEMENT

The authors confirm that there is no conflict of interest related to this work.

DATA AVAILABILITY STATEMENT

Data is available on request from the authors.

ORCID

Ahmed M. Jasim https://orcid.org/0000-0001-9276-577X

REFERENCES

1. Yang, Z., Liang, B., Ji, W.: An intelligent end–edge–cloud architecture for visual IoT-assisted healthcare systems. IEEE Internet Things J. 8(23), 16779–16786 (2021). https://doi.org/10.1109/JIOT.2021.3052778
2. Wu, F., et al.: Edge-based hybrid system implementation for long-range safety and healthcare IoT applications. IEEE Internet Things J. 8(12), 9970–9980 (2021). https://doi.org/10.1109/JIOT.2021.3050445
3. Gutierrez-Torre, A., et al.: Automatic distributed deep learning using resource-constrained edge devices. IEEE Internet Things J. 9(16), 15018–15029 (2022). https://doi.org/10.1109/JIOT.2021.3098973
4. Abdellatif, A.A., et al.: Edge computing for smart health, "Context-Aware approaches, opportunities, and challenges. In: IEEE Network, vol. 33, pp. 196–203 (2019). https://doi.org/10.1109/MNET.2019.1800083
5. Jasim, A.M., Al-Raweshidy, H.: Towards a cooperative hierarchical healthcare architecture using the HMAN offloading scenarios and SRT calculation algorithm. In: IET Netw, pp. 1–18 (2022). https://doi.org/10.1049/ntw2.12064
6. Wu, Q., et al.: Fedhome: cloud-edge based personalized federated learning for in-home health monitoring. IEEE Trans. Mobile Comput. 21(8), 2818–2832 (2020). https://doi.org/10.1109/tmc.2020.3045266
7. Singh, A., Chatterjee, K.: Edge computing based secure health monitoring framework for electronic healthcare system. Cluster Comput. 26(2), 1–16 (2022). https://doi.org/10.1007/s10586-022-03717-w
8. Rahman, M.A., Hossain, M.S.: An internet-of-medical-things-enabled edge computing framework for tackling covid-19. IEEE Internet Things J. 8(21), 15847–15854 (2021). https://doi.org/10.1109/jiot.2021.3051080
9. Alwan, O.S., Prahald Rao, K.: 'Dedicated real-time monitoring system for health care using ZigBee. " Healthcare Technol. Lett. 4(4), 142–144 (2017). https://doi.org/10.1049/htl.2017.0030
10. Aceto, G., Persico, V., Pescapé, A.: The role of information and communication technologies in healthcare: taxonomies, perspectives, and challenges. J. Netw. Comput. Appl. 107, 125–154 (2018). https://doi.org/10.1016/j.jnca.2018.02.008
11. Pham, M., et al.: Delivering home healthcare through a cloud-based smart home environment (CoSHE). Future Generat. Comput. Syst. 81, 129–140 (2018). https://doi.org/10.1016/j.future.2017.10.040
12. Uddin, M.Z.: A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system. J. Parallel Distr. Comput. 123, 46–53 (2019). https://doi.org/10.1016/j.jpdc.2018.08.010
13. Abdellatif, A.A., et al.: Edge computing for smart health: context-aware approaches, opportunities, and challenges. IEEE Netw. 33(3), 196–203 (2019). https://doi.org/10.1109/mnet.2019.1800083
14. Al-Anbagi, H.N., Vertat, I.: Collaborative network of ground stations with a virtual platform to perform diversity combining. In: 2022 International Conference on Applied Electronics (AE), pp. 1–6. Pilsen, Czech Republic (2022). https://doi.org/10.1109/AE54730.2022.9920037

15. Yan, H., et al.: Edge server deployment for health monitoring with reinforcement learning in internet of medical things. In: IEEE Transactions on Computational Social Systems, pp. 1–11 (2022). https://doi.org/10.1109/tcss.2022.3161996
16. Jia, M., Cao, J., Liang, W.: Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Trans. Cloud Comput. 5(4), 725–737 (2017). https://doi.org/10.1109/tcc.2015.2449834
17. Zhao, L., et al.: Optimal placement of cloudlets for access delay minimization in SDN-based Internet of Things networks. IEEE Internet Things J. 5(2), 1334–1344 (2018). https://doi.org/10.1109/jiot.2018.2811808
18. Xu, Z., et al.: Efficient algorithms for capacitated cloudlet placements. IEEE Trans. Parallel Distr. Syst. 27(10), 2866–2880 (2016). https://doi.org/10.1109/TPDS.2015.2510638
19. Fan, Q., Ansari, N.: Cost Aware cloudlet Placement for big data processing at the edge. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6 (2017). https://doi.org/10.1109/ICC.2017.7996722
20. Meng, J., et al.: Cloudlet placement and minimum-delay routing in cloudlet computing. In: 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), pp. 297–304 (2017). https://doi.org/10.1109/BIGCOM.2017.58
21. Yao, H., et al.: Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. Concurrency Comput. Pract. Ex. 29(16), e3975 (2017). https://doi.org/10.1002/cpe.3975
22. Santoyo-González, A., Cervelló-Pastor, C.: Network-aware placement optimization for edge computing infrastructure under 5G. IEEE Access 8, 56015–56028 (2020). https://doi.org/10.1109/ACCESS.2020.2982241
23. Li, D., et al.: Towards optimal system deployment for edge computing: a preliminary study. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–6 (2020). https://doi.org/10.1109/ICCCN49398.2020.9209754
24. Li, Y., Wang, S.: An energy-aware edge server placement algorithm in mobile edge computing. In: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 66–73 (2018). https://doi.org/10.1109/EDGE.2018.00016
25. Lähderanta, T., et al.: Edge computing server placement with capacitated location allocation. J. Parallel Distr. Comput. 2021(153), 130–149 (2021). https://doi.org/10.1016/j.jpdc.2021.03.007
26. Lovén, L., et al.: Scaling up an edge server deployment. In: 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 1–7 (2020). https://doi.org/10.1109/PerComWorkshops48775.2020.9156204
27. Bhatta, D., Mashayekhy, L.: A bifactor approximation algorithm for cloudlet placement in edge computing. IEEE Trans. Parallel Distr. Syst. 33(8), 1787–1798 (2022). https://doi.org/10.1109/TPDS.2021.3126256
28. Wang, Z., Gao, F., Jin, X.: Optimal deployment of cloudlets based on cost and latency in Internet of Things networks. Wireless Network 26(8), 6077–6093 (2020). https://doi.org/10.1007/s11276-020-02418-9
29. Zeng, F., et al.: Cost-effective edge server placement in wireless metropolitan area networks. Sensors 19(1), 32 (2018). https://doi.org/10.3390/s19010032

How to cite this article: Jasim, A.M., Al-Raweshidy, H.: Optimal intelligent edge-servers placement in the healthcare field. IET Netw. 1–15 (2023). https://doi.org/10.1049/ntw2.12097