Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 5776954, 10 pages
https://doi.org/10.1155/2022/5776954

Research Article

Internet of Medical Things and Edge Computing for Improving Healthcare in Smart Cities

Muna Alrazgan

Department of Software Engineering, College of Computer and Information Sciences, KingSaud University, Riyadh, Saudi Arabia

Correspondence should be addressed to Muna Alrazgan; malrazgan@ksu.edu.sa

Received 30 December 2021; Revised 30 January 2022; Accepted 4 February 2022; Published 3 March 2022

Academic Editor: Naeem Jan

To build prosperous smart cities, adequate infrastructure must be provided. Smart cities contain intelligent things to enhance lives and save people's lives. The Internet of medical things (IoM) and edge computing are part of these things. Healthcare services are essential services that should benefit from the infrastructure of smart cities. Increasing the quality of services (QoS) required increased connectivity and supercomputing. Supercomputing is represented by connecting the IoM with high processing devices close to these healthcare service devices called edge processing. Healthcare application requires low network latencies; therefore, edge computing must be necessary. Edge computing enables reduced latency and energy efficiency, scalability, and bandwidth. In this study, we review the most important algorithms used in the resource allocation management process at the MEC, which are the DPSO, ACO, and basic PSO. Our experiments have proven that the DPSO is the better and appropriate algorithm used in the event of intensive process congestion that needs to be addressed at the edges of the network to reduce time, including operations related to patients' health conditions.

1. Introduction

Internet of medical things (IoM) implements smart nodes that sense the data, interpret, process, and respond within a required time in a network. The capability to embed sensing devices into a natural environment enables the transformation of a smart environment. Nowadays, the smart industry proliferates because of the innovative environment and IoM. IoM is the online integration of smart nodes within a network to share information, communicate, and perform a task. It is a combination of smart modules which do their work by coordinating with each other. The smart nodes are designed to sense, share, interpret results, and perform some actions.

Smart city includes various services such as healthcare, transportation, smart electricity, management, sewerage, and buildings. The data in a smart element are generated through smart services, which is helpful to achieve further actions accordingly in a smart city. Cloud computing is the best method for computing smart cities due to its collaboration and elasticity of all smart nodes [1]. Nonetheless, due to the long distance between the smart devices (SDs) and the sensitivity of the data that require processing in a limited period, the idea of edge computing (EC) close to SDs was proposed. The EC includes several meanings, including the cooperation between the surrounding SDs, exploitation of the 5G networks, and EC in the base stations' (BSs) telecom service providers.

In smart cities, the system is controlled and managed by the smart nodes by providing reliable and effective data, which led to changing the real environment to the automation system. Smart cities have numerous components, such as monitoring of weather, management of waste, smart homes, buildings, management of energy, sewerage, medical services, pollution of air, detection and monitoring of forest fire, structural health, congestion of traffic, radiation levels, smart lightening, intelligent shopping, smart roads, maps of urban noise, and diagnosis of vehicle auto. The smart cities concept is coming into reality by applying different smart approaches to make the environment sustainable and smart [2]. IoM and EC technologies have recently seen ceaseless improvements and have helped provide a smart

environment for healthcare services. With IoM-edge cloud computing integration, the demand for smart healthcare as a ubiquitous module that offers seamless and fast response is considerable. Deep reinforcement learning and artificial intelligence (AI) can provide reasoning behaviour and enhance the making of decision capability.

Besides, SDs and technologies are the monitoring stakeholders of smart cities. However, accessing specialized medical hospitals and practitioners is challenging in the complex environment of a smart city. Occasionally, patients with complex cases need instant care and a fast response to life. Therefore, multipart multimedia signals must be conveyed and computed with minimum latency, and the result produced must be suitably accurate for physicians based on initial analysis. Therefore, a combined smart healthcare module that edge cloud computing (ECC) deals with these issues by utilizing the resources available and technologies in the environment of a smart city is essential.

The healthcare industry is now one of the fastest growing fields with considerable demand. It is not just about providing critical services and important to patients, it also brings significant revenue to the health sector. The healthcare providers compete to offer efficient and inexpensive healthcare services to smart city populations [3]. Revolution of the SDs and cheap smart IoM devices of healthcare play a significant role in achieving these purposes, including monitoring blood sugar by smart wearable devices, body temperature, blood pressure, weight, stress, and people who have had been transplanted. Therefore, the integration of IoM-edge cloud technologies has emerged under research recently. This study presents a smart city healthcare monitoring system powered by edge computing and IoM. This can enhance healthcare facilities and infrastructure and ensure on-time care for patients. This could be extremely helpful, especially during pandemic situations, for improving the health infrastructure. In addition, a solution for providing low latency and scalability is proposed: end-to-end network slices for healthcare services. Furthermore, scheduling patients and allocating resources efficiently are proposed.

The following is an outline of the paper's structure: Section 2 provides a literature review for the present work, an assessment of the design of smart healthcare systems.

2. Related Work

Smart healthcare systems have recently attracted much attention due to their enormous economic and social benefits. Smart healthcare solutions have been proposed, which integrate IoM cloud services [4, 5]. The smart healthcare framework presented in Demirkan [6] provides patients with smart devices to assist them in finding hospitals. Several studies propose frameworks for maintaining electronic health records [7]. IoM applications and devices are set to be adopted by the health sector with the exponential growth of IoT [8]. These applications and equipment may handle individuals' private health information. Internet resources make it possible to access the same globally. Considerations must be given to issues of security and privacy in particular. A security requirement might include threats, vulnerabilities, and so on. Regarding IoT-cloud-based healthcare systems, some issues and challenges must be resolved [8], such as confidentiality, integrity, availability, data freshness, authorization, privacy, and computational limitations. The IoT application market is also receiving enormous attention and IoT services. Users and patients use applications directly instead of services to develop apps. Services are oriented towards developers, whereas applications are directed towards users. Today, various wearable sensing devices are available on the market, and multiple applications could be classified as IoT modernization to facilitate different healthcare solution options. Tests to measure blood glucose levels are known as glucose-level sensing [9]. Diabetes is a metabolic disease that is especially relevant to diabetes management. Sugar levels in the blood are high for an extended period, causing diabetes. Monitoring blood glucose allows you to plan meals and activities and determine when to take medications. It also shows when blood sugar levels fluctuate. On a real-time basis [8], propose a method for noninvasive glucose sensing based on the IoM in healthcare. In order to connect healthcare providers and sensors, IPv6 connectivity is used. Several components are needed to make this work: a mobile phone, computer, processor, and a blood glucose meter.

In recent years, 5G has prompted a renewed interest in multiaccess edge computing, also called MEC. As a processing terminal, base stations or servers use EC. These stations can process their data and trigger the responses by proximity. A central cloud server can be used to store data permanently. The edge-assisted cloud computing system relies on both data producers and consumers. A computing edge and a network edge reduce the latency, increase the transmission rate, and increase the analysis power [10]. Computer edge nodes perform computation, compute, and translate analysis from cloud to edge nodes while networks edge nodes perform calculations and service requests from the cloud [11]. In this way, primary disease diagnosis is performed at the edge, and supervision is performed at the cloud as an advantage of cloud processing [12]. Wireless access points, routers, and bridges are used as ESs; mobile phones and smart devices work together to improve edge computation capabilities [10].

To maintain homogenous consistency and optimize dosage regimens, drug monitoring ensures the concentration of drugs in an individual's blood supply is monitored and analyzed at a fixed, regular interval [13]. Therapeutic drug monitoring (TDM) combines knowledge of pharmaceutics, pharmacokinetics, and pharmacodynamics and is the clinical measurement of chemical parameters with proper medical interpretation for prescribing to patients safely [13, 14]. Based on the relationship between dose and concentration and its therapeutic effect, the system monitors drugs with low therapeutic ranges, pharmacokinetic variability, multiple concentration targets, and causing adverse health effects.

A smart city service and application uses different types of sensor-embedded electronic devices for collecting data crucial to efficiently managing resources and the assets of a

municipality, including government, transportation, healthcare, libraries, schools, water supply, law enforcement, and other community services. An application performs various functions to solve issues concerning local communities such as parking, public safety, lighting, and waste. Several reasons explain how it can benefit from using the edge computing platform: large data quantity, low latency, location awareness [15]. IoT-enabled edge devices could revolutionize healthcare by providing a wide range of health services to the community through many embedded sensors and IoT devices [16].

3. Healthcare Monitoring System Using IoM

There have been significant advancements and innovations in the area of edge healthcare networks. As a result of the literature review, we identified some fundamental requirements and concerns during implementation. The solution we provide is aimed at solving the same problems.

There are no adequate processing powers or memory to store vast quantities of data generated by these devices because they are embedded with sensors. During rainy days, it is crucial to keep track of medical data and alert a doctor as soon as possible if there is an issue called responsiveness. A heart attack or a paralytic stroke is an example of critical circumstance or emergency in which this information is beneficial. In order for a swift treatment to be administered, the doctor needs to know right away. In this case, the total duration of time devoted to data transmission, notification, analysis, and redressing should be minimized. Analysing historical data, diagnosing diseases, and prescribing treatments benefit an intelligent system. In order to take full advantage of a single system, other practitioners must be able to access the data and any essential findings or models generated. Data formats for different IoM devices differ and the hardware they use. Data gathered from IoM devices needs to be understood and transmitted correctly. A device's data must be decoded and sent correctly from the IoM. Short message services or e-mails informing doctors and family members of any urgent cases must be integrated into the system.

Therefore, to solve these issues, MEC technology is proposed. The MEC contains the edge central controller (ECC), EC, and cloud data center (CDC). The EC is deployed between the mobile device and the CDC. Using the ECC, hospitals can see information about the resources available to them. In addition to scheduling the patient, the EC is responsible for ensuring that the patient receives appropriate help as quickly as possible. The ECC monitors different EC that are connected to various hospitals. The ECC receives a request for services from another EC node because they lack sufficient resources. As a result, the ECC determines which nodes connected to the patient can provide the patient with resources. Once it is identified, the patient's request to receive services is routed to a peer node, and the resource flow is diverted to the patient from that node. In a real-time environment, everything occurs at the EC and ECC. This architecture reduces service latency dramatically while the ECC provides resources. One edge controller (ECC) can handle several edge nodes (EC). This makes the architecture scalable. The EC node would ensure connectivity for the patient across any of these nodes, which would significantly increase the system's reliability and responsiveness. Using end-to-end network slicing at the near edge, multiple services can be offered on the same infrastructure. Each node at the edge implements its functions in various slices. Consequently, different core functions can be gathered into QoS slices.

Figure 1 presents a proposed architecture for a healthcare IoM network based on IoM technology that efficiently and quickly serves healthcare agencies. There are three segments to the architecture. Following are the descriptions of the corresponding components:

3.1. IoM-Edge. The main objective of the designed healthcare module is to schedule computation resources and transmission for efficiently controlling and monitoring processes in the IoMEC system. Monitoring healthcare data involves three essential characteristics: medical urgency, energy consumption, and cost. Medical urgency measures the healthcare data's health severity index. Intuitively, blood sugar data and heart rate data are different medical speeds. It is possible to categorize healthcare data based on their urgency. Assuming the healthcare data with a high medical urgency have the same attribute, it should be assigned the highest priority for transmission. A medical record is generally a time-sensitive data set, and monitoring data that are out of date is of little use to healthcare professionals. Body sensors can monitor instant data throughout the day to offer real-time healthcare. However, IoMs are constrained in terms of energy supply and computation resources, which require continuous monitoring. So, the freshness of received healthcare data by gateways is measured using the MoIEC paradigm. When body sensors collect data, they are timestamped.

The system model investigated in Figure 1 contains fourth layers: SDs, EC, ECC, and remote cloud (RC). As shown in Figure 1, each layer possesses specific functionality and communication interactions. To reduce network congestion and improve the network's real-time communication capabilities, we designed and implemented a traffic prediction method based on deep learning to address the ubiquitous and low-latency nature of IoMEC applications.

3.1.1. Smart Devices. It is a sensing layer, illustrated in Figure 1, generating and collecting information about healthcare and behaviour. In this layer, data are sensed contextually in real-time and non-real-time through various wearable devices placed under multiple scenarios, such as soccer grounds, fitness rooms, and unrestricted areas; data sensing is the main functionality of this layer. The ubiquitous IoMEC application is connected to this layer by any healthcare data acquisition and transmission embedded device, differentiating it from existing healthcare frameworks. Our module enables seamless access to edge computing for the ubiquitous IoMEC application based on edge computing. Sensors or microphones on smartphones can

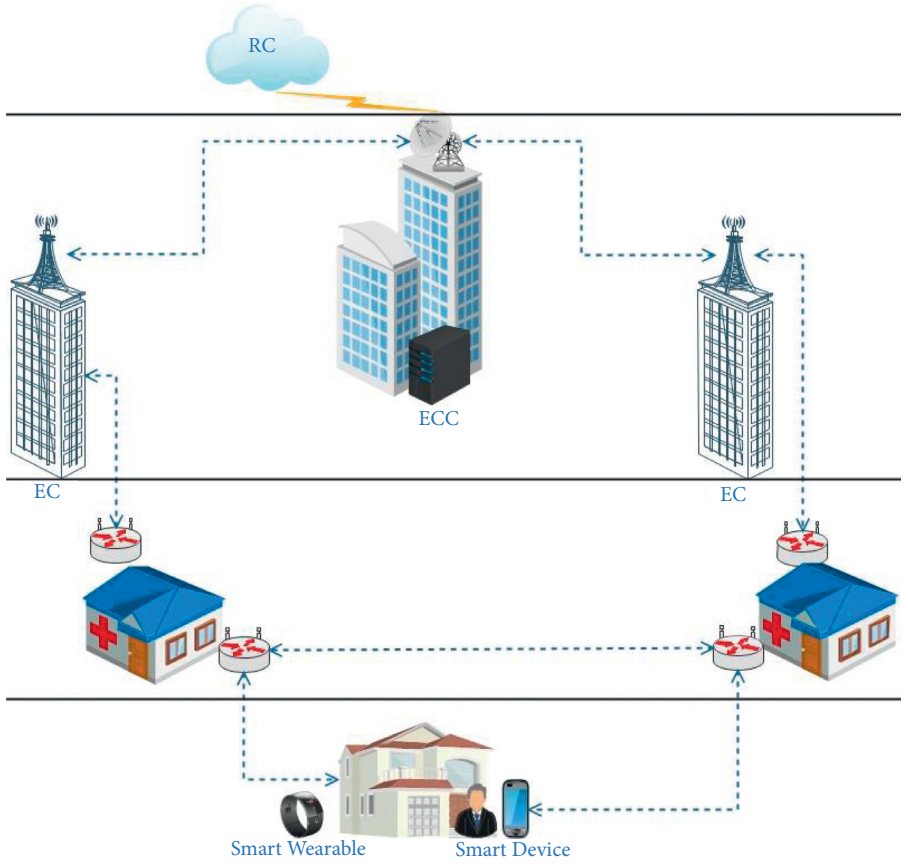

Figure 1: IOM architecture network.

collect health data, such as heart rate or a description of heart sounds. Patients may enjoy smartphone-based sensors, but the number of sensors embedded in the hardware is limited. However, a dedicated medical sensor can generate and process more significant sensor data, providing a more accurate diagnosis. Sensors are used in health monitoring to track heartbeat, oxygen saturation, blood pressure, and glucose levels. In addition, the sensor can detect actions, such as type of activity, steps, and sleep cycles, regarding motion. Internet of Things (IoT) devices are the newest trend in health edge computing.

3.1.2. Edge Computing. Edge computing moves the processing closer to the network's edge, enabling faster response times and more energy-efficient operations. Instead, the data can be mined and processed closer to the users instead of constantly moving the data to the cloud, incurring an energy cost. Health monitoring through edge solutions allows for the prompt arrival of emergency medical help in situations requiring health monitoring. Traditionally, cloud services send large amounts of data, so privacy and security remain major concerns, especially if patient information can be exposed. The edge can distribute information instead of concentrating just one part of the network with important information. In addition, these devices must be easy to use because untrained personnel must still transmit accurate data with these sensors. The health monitoring using 5G m is embedded with the communication base station (BS), and there are central controllers to monitoring offloading tasks.

3.1.3. Edge Central Controller. The edge ECC is a unit used to control and monitor each EC which redirects the intensive tasks to the appropriate resources. The task offloading information is first uploaded to the EC, embedded in the BSs, and was collected to the central edge controller (ECC). Then the controller determines the most efficient scheduling strategy for offloading. In addition, users upload a summary of their tasks. This includes data size ls, execution time, and completion rate.

3.2. System Model. This study proposes a network architecture that integrates a cloud data center (CDC-RAN) with MEC to achieve low latency, cost, and traffic. We assume that the MEC is embedded into the traditional BS. In addition, the BS will be split into EC resources and C-RAN. EC functions provide the resources needed to process the offloaded tasks. In addition, the EC offers external system resource management, operation, maintenance, and environmental monitoring, as shown in Figure 2.

The C-RAN function is to receive the tasks, schedule them, and manage resources to serve the most significant amount of the offloaded tasks, or transfer them to the adjacent EC and direct them to the available resources. The

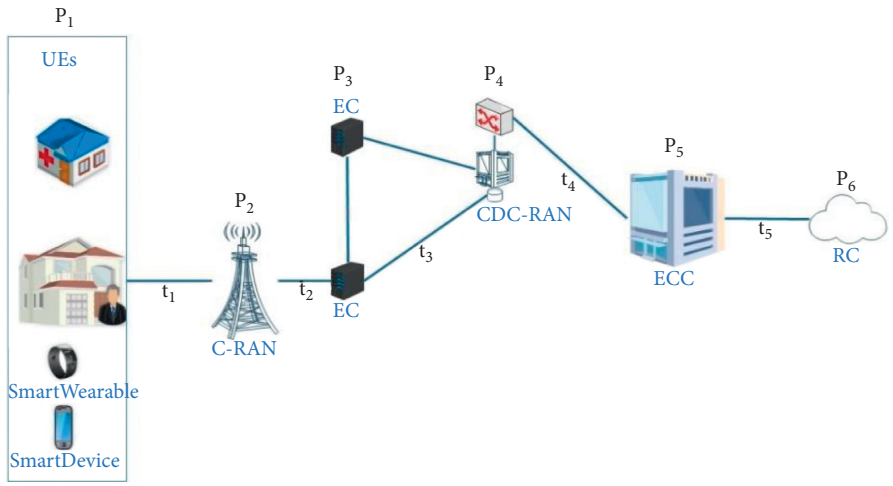

Figure 2: System Model diagram.

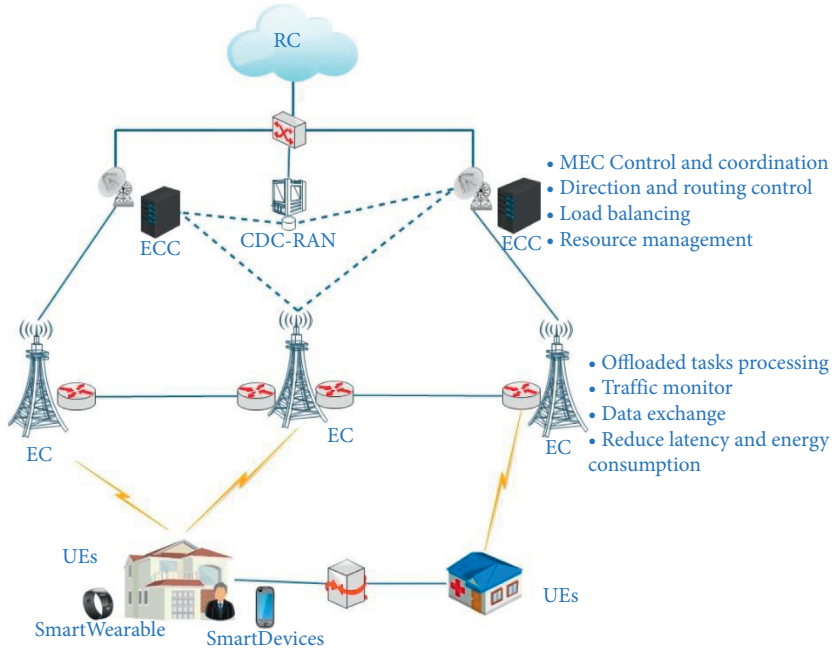

Figure 3: Network architecture.

ECC is the monitoring of the environment and management of system resources. ECCs are responsible for connecting the individual components of a network and controlling the path the data take to reach the target destination. A suitable algorithm can be used to balance the load of the network components. ECC must also consider each CDC-RAN pool's type and quantity of data requirements when optimizing its cache mechanism. Cache management and coordination are essential functions of EC. Despite the possibility of connecting ECCs, not all ECCs can communicate due to distance restrictions. Figure 1 shows the detailed network architecture. This architecture allows UEs to obtain data directly from ECs, reducing transmission delays. It can dramatically reduce data transmission when many UEs send the same data request because there is no need to access the RC to obtain data.

Our scenario is detailed in Figure 3. MEC networks have task offloading systems where a single user is equipped with a base station to execute some offloaded tasks N with M nearby ECs. Therefore, the task is divided into nth subtasks that can be allocated to nearby ECs. Practically, the computational capability of each *ECs* is maybe different, and we use $\complement_m$ to denote the computational capability of the mth EC, where $\complement_m$ can be measured by the number of CPU cycles required for each subtask for a particular bit. Furthermore, the ECs may have different prices for the user, so we use ρ_x as a measure of how much it costs to compute 1M bits within each EC. Furthermore, we identify the upload link and download link bits for the nth subtask with U_n and D_n.

We can determine whether the nth subtask has been offloaded or not by using x_{nm}:

$$x_{\mathrm{nm}} = \begin{cases} 0, & \text{subtask offloaded to the EC}_m \\ 1, & \text{subtask nonoffloaded to the EC}_m \end{cases}, \quad (1)$$

where $X = \{x_{\mathrm{nm}}, 1 \le n \le N, 0 \le m \le,$ represent the offloading strategy matrix.

In order to save computation resources, it is recommended that each subtask be computed only once. When the subtasks local processing computational latency and the energy consumption for the nth subtask are given by

$$t_l = \frac{U_n}{\complement_n}, \qquad e_l = p_l * \quad \text{where } n = 0, \quad (2)$$

where $\complement_n$ local computational capability and p_l is regional computational power.

On the contrary, if the offloading decision is made to upload to the nearby CE $= P_2$ with $x_{nm} = 1$, the upload and download transmission times are given by

$$t_{P2}^{U} = \frac{U_{P2}}{L_s^U}, \qquad t_{P2}^{D} = \frac{D_{P2}}{L_s^D}, \quad (3)$$

where the t_{P2}^{U} and t_{P2}^{D} are the upload link and download link of transmission time, respectively. The L_s^U and L_s^D denote the transmission data rate of the uplink and downlink in point P2, respectively. In addition, the computational time for the nth subtask executed at nearby the ECm is given by

$$T = \frac{U_{P2}}{\complement_{P2}}, \quad (4)$$

where the $\complement_{P2}$ is the computational capability of the EC$P2$. The energy consumption at the ECs is ignored in.

The ECs are directly connected to the power supply in general. From the t_{P2}^{U} and t_{P2}^{D}, the transmission energy is given by

$$e_{P2}^{\mathrm{nm}} = p_U \frac{U_n}{\complement_{P2}} + p_D \frac{U_n}{\complement_{P2}}, \quad (5)$$

where the p_U and p_D are the uploading and downloading computational power and the $\complement_{P3}$ is the capability of the P2.

When the nearby CEs (P1) are unable to process the offloaded task for any reason, the ECC (P5) monitors the ecosystem using the CDC-RAN (P4), the appropriate adjacent EC to process the offloaded tasks. The upload and download transmission times are given by

$$t_{P3}^{U} = \frac{\alpha_U}{L_s^U}\left(t_{P2}^{U} + U_{P3} + U_{P4} + U_{P5}\right), \qquad t_{P3}^{D} = \frac{\alpha_D}{L_s^D}\left(t_{P2}^{D} + D_{P3} + D_{P4} + D_{P5}\right), \quad (6)$$

where the α_U and α_D are the uplink and downlink transmission weight from the P2 to the P3, respectively. The U_{P3}, U_{P4}, and U_{P5} are the upload link bits for the nth subtask with each point (P3, P4, and P5), respectively. The D_{P3}, D_{P4}, and D_{P5} are the download link bits for the nth subtask with each point (P3, P4, and P5), respectively. In addition, the download link bits for the nth subtask with each point (P3, P4, and P5), respectively.

From the t_{P3}^{U} and t_{P3}^{D}, the transmission energy is given by

$$e_{P3}^{\mathrm{nm}} = p_U \frac{t_{P3}^{U}}{\complement_{P3}} + p_D \frac{t_{P3}^{D}}{\complement_{P3}}. \quad (7)$$

Therefore, the computing cost of the offloaded subtask in by the EC is given by

$$\mathscr{C}_{\mathrm{nm}} = \rho_m U^n. \quad (8)$$

We can determine whether the nth subtask has been computed in the nearby EC or by the adjacent CE using x_{nm}:

$$\mathrm{Z}_{\mathrm{nm}} = \begin{cases} 1, & \text{subtask computed in the nearby EC}_m \\ 0, & \text{subtask computed in the adjacent EC}_m \end{cases}. \quad (9)$$

Furthermore, we can calculate the latency of the ECs required to compute the offloaded subtask, given by

$$L_m = \sum_{n \in N} x_{\mathrm{nm}}\left(\mathrm{Z}_{\mathrm{nm}}\left(t_{P2}^{U} + t_{P2}^{D}\right) + \mathrm{Z}_{\mathrm{nm}}\left(t_{P3}^{U} + t_{P3}^{D}\right)\right), \quad \text{where the m} > 0. \quad (10)$$

The L_m represents the transmission and computation latency of the subtasks offloaded to nearby ECs.

The total system cost $\mathscr{C}_{\mathrm{total}}$ from the ECs is given by

$$\mathscr{C}_{\mathrm{total}} = \sum_{m=1}^{M} \sum_{n=1}^{N} \mu_{\mathrm{nm}}. \quad (11)$$

The total energy (E) consumption of the system consists of computation energy and transmission.

$$E = \sum_{n=0}^{N} \sum_{m}^{M} e_{P2}^{\mathrm{nm}} + e_{P3}^{\mathrm{nm}}. \quad (12)$$

Therefore, we formulate the optimization problem to measure the system performance from the three-performance metrics: latency L_m, total energy E, and the total cost $\mathscr{C}_{\mathrm{total}}$. The optimization problem is formulated by

$$\beta = \sigma_1 L_m + \sigma_2 E + (1 - \sigma_1 - \sigma_2)\mathscr{C}_{\mathrm{total}}. \quad (13)$$

where σ_1, σ_1 denote the weight factors of the latency, energy consumption, and the cost price, respectively. These factors play a more critical role in the system for each performance metric.

3.3. IoM Empowered by MEC Based on DPSO. In this section, we will describe a performance metric to compare the effectiveness of MEC networks based on latency, energy consumption, and cost. In addition, we present the offloading module design based on dynamic particle swarm optimization (DPSO) for the MEC network.

First, we used an array signal processing at the user, which was mentioned by Guo et al. [17] given by

$$\mathcal{S}_m^U = B\,\mathrm{Log}_2\left(1 + \rho^U \sum_{k=1}^{K} |h|^2\right), \tag{14}$$

where B is the bandwidth between the user and CEs and and h denotes the channel parameter between the user's base station and the CEs.. The transmission between the user and the CEs is optimized after signal processing is performed at the user. This study employs the DPSO scheme to obtain an efficient offloading strategy.

The ultimate objective is to minimize the latency, cost, and energy consumption. In this problem, we have a mixed integer nonlinear optimization problem that is NP-hard. A PSO algorithm based on heuristics is proposed to solve the approximate optimal solution of this problem, which can be very difficult to solve as the amount of data increases exponentially. PSO algorithms usually employ stochastic or greedy strategies for generating their initial solutions. The PSO algorithm is a heuristic algorithm and a bionic optimization algorithm based on the swarm intelligence that simulates the behaviour of swarm [18]. Intelligent swarms are systems that are capable of achieving a level of intelligence that is really beyond the capabilities of any one of their units. In order to apply DPSO to the problem of task assignment in MEC networks, we modify the optimization objective to reduce the cost towards an optimal path rather than obtaining a shortest path. To solve optimization problems, DPSO uses a simplified model of social behaviour in conjunction with a cooperative and intelligent approach. Then the swarm will make a corresponding decision based on all choices' social behaviour and finally select the best allocation strategy for the task offloading in MEC networks.

3.4. DPSO Algorithm. Under a dynamic environment, basic PSO algorithm can fall into local optimum easily. As the global environment changes, it is difficult for organizations to optimize their operations. Using DPSO, enable the particle or population to sense the external environment in order to be able to adapt their position according to changes in the external environment. In addition, a response feedback mechanism will allow more effective adaptation to the dynamic environment based on the detected changes. With the MEC model, a task migration algorithm called DPSO is proposed for a given set of tasks. Considering the real-time state of the system, the algorithm optimizes latency, energy consumption, and cost of the system. First, it keeps track of particles for how long their personal best positions pBest are not updated. A second consideration is that it also counts iteration because the group's global best ranking (gBest) has not improved. After a predefined number of iterations, particles pBest and groups gBest that have not improved for a while are restructured temporarily for a few iterations, taking into account the particle's best position in a previously selected iteration and the particle's current personal best in a randomly selected iteration respectively. With DPSO, address two of the core causes for stagnation related to PSO. Therefore, the steps that comprise the DPSO algorithm are mentioned as follows:

Evaluating an initial population: the DPSO Algorithm 1 starts with an initial random population matrix like many evolutionary algorithms. In the population, each element is referred to as a node. The DPSO consists of a finite number of nodes that randomly choose the initial value. Here, the edge layer is consisting of a C-RAN, EC nodes, and ECC coordinator nodes. The n-dimensional search space is initialized by randomly selecting the position and velocity of each particle.

A controller can use these vectors to find the best node. Local optimal positions can be found using the location vector, and optimal global positions (X^M) are found using the velocity vector (V^M).

The Fitness Function: pBest, and gBest computation of each particle: The initial population is evaluated using the fitness function. As the problem is a three-objective optimization, the fitness function should consider these objectives. The first objective is to minimize the total latency in the edge network indicated by L_m using equation (13). The second objective is to reduce the energy consumption by equation (15). The third objective is to minimize the cost using equation (14). These objectives are defined as minimization. Ultimately, fitness is calculated as an optimization problem mentioned in equation (16).

Location and Velocity Updates: An optimization problem has as many parameters as the function to optimize. Throughout the system's history, the best positions of all nodes are stored, and nodes choose their next move based on this information. All nodes are moved into the next n-dimensional space to locate the general optimal point as each iteration proceeds. According to the best absolute and local solutions, the nodes update their velocities and positions. In this case, the ECC nodes read the EC nodes' availability tables and publish those that are the best to the controller. For this reason, we update the velocity by the following equation:

$$\begin{aligned} V_N^M = {} & \omega V_N^M(t-1) + c_1 R_1\left(\mathrm{pBest}_N^M - x_N^M(t-1)\right) \\ & + c_2 R_2\left(\mathrm{gBest}_N^M - x_N^M(t-1)\right), \end{aligned} \tag{15}$$

where $\omega \in [0.9\,\mathrm{to}\,0.4]$ is a inertia weight between [0, 5], pBest_N position obtained by nth particle, gBest_N global best position obtained by swarm (group best), R_1 and R_2 random numbers have uniform distributions in the range of 1 to 2, and (c_1, c_2) acceleration coefficients with the constant value (1.4944). In addition, we update the position vectors of particles as follow:

$$x_N^M = x_N^M(t-1) + V_N^M(t). \tag{16}$$

4. Simulation Results

In this section, we show some simulation results which support the proposed studies. We consider that the task is 250 Mb in size, and it has been randomly divided into five subtasks. We used the default parameters of the experiment setting in the [18]. The CPU frequency is set to $1200 * 10^8$ (unit: cycles/s). For different types of mobile devices,

(1) Initialize the particles position initialization (X_M) and particles velocity initialization (V_M).
(2) Initialize pBest = 0 and gBest = 0.
(3) Compute the fitness of each particle and compare this value with the optimal historical value, X_m.
(4) Find the best particle in the group, compare it with the best particle in the group's history and pick the better particle.
(5) Update the velocity and position of the particles using equations (15) and (16), respectively.
(6) Repeat Steps 2 through 4 until you have reached the iterations maximum number or the minimum error.

Algorithm 1: Dynamic PSO.

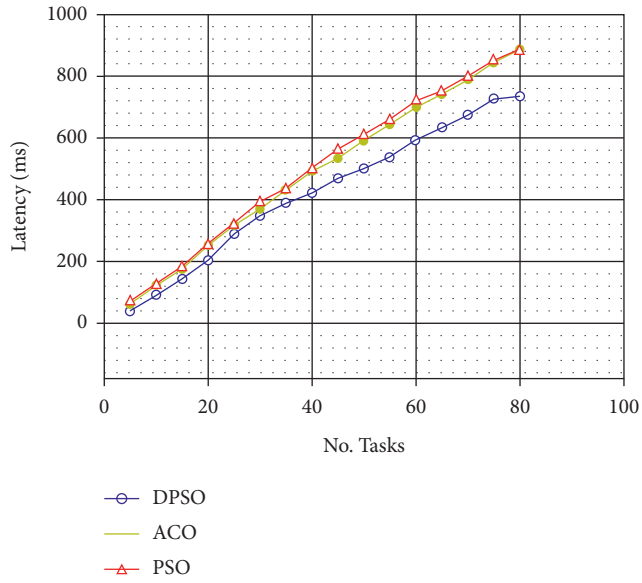

Figure 4: The latency of response per the tasks.

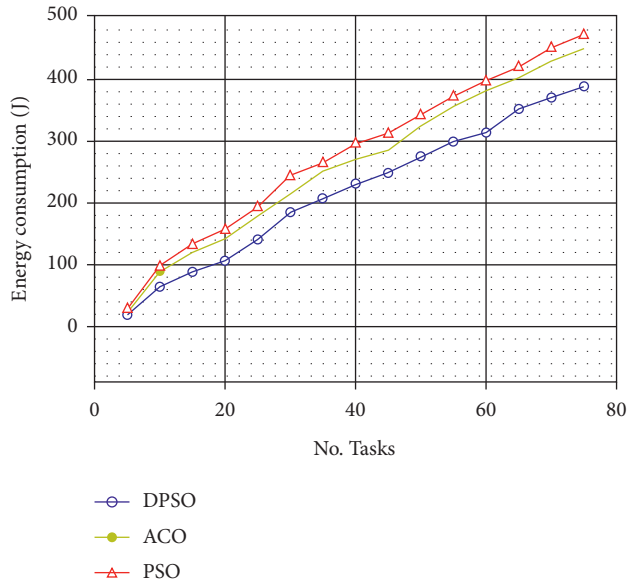

Figure 5: The energy consumption per the tasks.

multiple interdependent task maps are constructed, and the final result represents the average of multiple experimental results. As a result of our experiments in this paper, we can verify that the proposed algorithm is accurate and reliable for the 15 interdependent tasks. CloudSim tool is used to verify the DPSO algorithm and optimize the parameters in the algorithm. As shown in the following figures, the optimal fitness function value varies with the number of iterations.

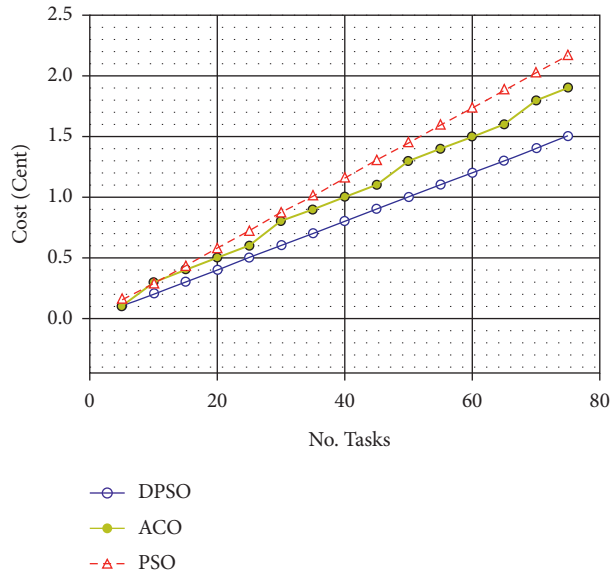

Figure 6: The computation cost per the tasks.

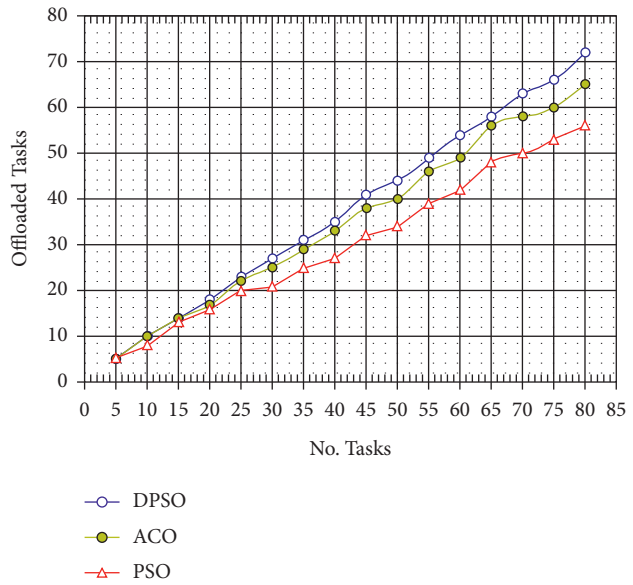

Figure 7: The number of tasks that have been uploaded based on the offloading decision.

In Figure 4, three algorithms were compared. We assumed that the system depends on the offloading decision using the three algorithms—DPSO, ACO, and standard PSO—to calculate the latency time to the computation of offloaded tasks. We started to upload five tasks as a package, and in each iteration, the system offloads a larger number of the previous offload with five tasks until a total of 80 tasks. We calculate the offloading time for each package.

We found that the DPSO algorithm is superior to ACO and basic PSO, which showed better results. The results prove that the DPSO algorithm is the most suitable. In addition, the DPSO appropriates to the smart cities' environment in terms of the difference in the number of tasks to be calculated from time to time, especially peak time.

Figure 5 shows energy consumption against the number of tasks in the taskgraph. We notice an improvement in the process of energy consumption in the DPSO by 15% over the ACO algorithm, and 19% over the basic PSO. This indicates that the DPSO depends on the current state of the environment the optimal exploitation of resources.

In Figure 6, we show the cost of computation against the offloaded tasks. It appears that an increase in the VM that is created in each server of the ECs has more effect on the computation cost or the performance of the algorithms used for comparison. Therefore, the DPSO is based on creating VMs with fewer numbers and optimizing them, and this is reflected in the lower cost compared to other algorithms. Here, we note the low cost of computing offloaded tasks, even if slightly when using the DPSO.

In Figure 7, we note that the DPSO is suitable in the offloading decision process, as it works on analysing the current state of the resources, task and network environment, and making the appropriate decision by calculating the time taken to compute the task in locally or at the ECs. The DPSO algorithm showed superiority over the other algorithms by the number of offloaded tasks assigned to the ECs and remote processing.

5. Conclusion

In this paper, we studied the offloading strategies for the MEC networks and resource allocation, where some intensive computational tasks are to be computed either locally or by the nearby ECs in the MEC. Computation price, energy consumption, and latency have all been linearly combined to calculate the system cost. The smart offloading strategy was designed using the DPSO algorithm based on the system cost. Healthcare services are essential services that should benefit from the infrastructure of smart cities. Increasing the quality of services (QoS) required increased connectivity and supercomputing. Supercomputing is represented by connecting the IoM with high processing devices close to these healthcare services devices, called edge processing. Healthcare application requires low network latencies; therefore, edge computing must be necessary. Edge computing enables reduced latency and energy efficiency, scalability, and bandwidth. In this study, we review the most important algorithms used in the resource allocation management process at the MEC, which are the DPSO, ACO, and basic PSO. Our experiments has proven that the DPSO is the better and appropriate algorithm used in the event of intensive process congestion that needs to be addressed at the edges of the network to reduce time, including operations related to patients' health conditions.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

There are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the Researchers Supporting Project number (grant no. RSP-2021/206), King Saud University, Riyadh, Saudi Arabia.

References

[1] K. Kumar and Y.-H. Yung-Hsiang Lu, "Cloud computing for mobile users: can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[2] P. Sotres, J. R. Santana, L. Sanchez, J. Lanza, and L. Munoz, "Practical lessons from the deployment and management of a smart city internet-of-things infrastructure: the smartsantander testbed case," *IEEE Access*, vol. 5, pp. 14309–14322, 2017.

[3] S. Gera, M. Mridul, and S. Sharma, "IoT based automated health care monitoring system for smart city," in *Proceedings of the 2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 364–368, IEEE, Erode, India, 8 April 2021.

[4] T. M. Alfaqih and J. J. I. J. o. C. A. Al-Muhtadi, "Internet of things security based on devices architecture," vol. 133, no. 15, pp. 19–23, 2016.

[5] T. M. Alfaqih and M. M. J. I. J. o. C. S. Hassan, "Engineering GIS Cloud: integration between cloud things and geographic information systems (GIS) opportunities and challenges," vol. 3, no. 5, pp. 360–365, 2016.

[6] H. Demirkan, "A smart healthcare systems framework," *IT Professional*, vol. 15, no. 5, pp. 38–45, 2013.

[7] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "UbeHealth: a personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities," *IEEE Access*, vol. 6, pp. 32258–32285, 2018.

[8] K. Jaiswal and V. Anand, "A survey on IoT-based healthcare system: potential applications, issues, and challenges," in *Advances in Biomedical Engineering and Technology*, pp. 459–471, Springer, New York, US, 2021.

[9] R. S. Istepanian, S. Hu, N. Y. Philip, and A. Sungoor, "The potential of Internet of m-health Things "m-IoT" for non-invasive glucose level sensing," in *Proceedings of the 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5264–5266, IEEE, Boston, MA, USA, 30 August 2011.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[11] R. K. Barik, H. Dubey, C. Misra et al., P. Mishra, H. Das, and K. Mankodiya, Fog assisted cloud computing in era of big data and internet-of-things: systems, architectures, and applications," in *Cloud Computing for Optimization: Foundations, Applications, and Challenges*, pp. 367–394, Springer, New York, US, 2018.

[12] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: a taxonomy, survey and future directions," in *Internet of Everything*, pp. 103–130, Springer, New York, US, 2018.

[13] J.-S. Kang and M.-H. Lee, "Overview of therapeutic drug monitoring," *The Korean Journal of Internal Medicine*, vol. 24, no. 1, p. 1, 2009.

[14] D. J. Touw, C. Neef, A. H. Thomson, and A. A. Vinks, "Cost-effectiveness of therapeutic drug monitoring," *Therapeutic Drug Monitoring*, vol. 27, no. 1, pp. 10–17, 2005.

[15] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[16] V. K. Rathi, N. K. Rajput, S. Mishra et al., "An edge AI-enabled IoT healthcare monitoring system for smart cities," *Computers & Electrical Engineering*, vol. 96, Article ID 107524, 2021.

[17] Y. Guo, Z. Zhao, R. Zhao et al., "Intelligent offloading strategy design for relaying mobile edge computing networks," *IEEE Access*, vol. 8, pp. 35127–35135, 2020.

[18] T. Alfakih, M. M. Hassan, and M. J. I. A. Al-Razgan, "Multi-objective accelerated particle swarm optimization with dynamic programing technique for resource allocation in mobile edge computing," *IEEE Access*, vol. 9, 2021.