Hybrid Workload Enabled and Secure Healthcare Monitoring Sensing Framework in Distributed Fog-Cloud Network

Lakhan, A., Mastoi, Q., Dootio, M. A., Alqahtani, F., Alzahrani, I. R., Baothman, F., Shah, S. Y., Shah, S. A., Anjum, N., Abbasi, Q. H. & Khokhar, M. S.

Article

Hybrid Workload Enabled and Secure Healthcare Monitoring Sensing Framework in Distributed Fog-Cloud Network

Abdullah Lakhan [1], Qurat-ul-ain Mastoi [2,*], Mazhar Ali Dootio [1], Fehaid Alqahtani [3], Ibrahim R. Alzahrani [4], Fatmah Baothman [5], Syed Yaseen Shah [6], Syed Aziz Shah [7], Nadeem Anjum [8], Qammer Hussain Abbasi [9] and Muhammad Saddam Khokhar [1]

1   Research Lab of AI and Information Security, Benazir Bhutto Shaheed University Lyari, Karachi 74660, Pakistan; abdullahrazalakhan@gmail.com (A.L.); mazharaliabro@bbsul.edu.pk (M.A.D.); saddam_khokhar@hotmail.com (M.S.K.)
2   Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia
3   Department of Computer Science, King Fahad Naval Academy, Al Jubail 35512, Saudi Arabia; F-alqahtani@rsnf.gov.sa
4   College of Computer Science and Engineering, University of Hafr Al Batin, Al Jamiah, Hafar Al Batin 39524, Saudi Arabia; ialzahrani@uhb.edu.sa
5   Faculty of Computing and Information Technology, King Abdul Aziz University, Jeddah 21431, Saudi Arabia; fbaothman@kau.edu.sa
6   School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK; syedyaseen.shah@gcu.ac.uk
7   Research Center for Intelligent Healthcare, Coventry University, Coventry CV1 5RW, UK; Syed.shah@coventry.ac.uk
8   Department of Computer Science, Capital University of Science and Technology, Islamabad 44000, Pakistan; Nadeem.anjum@cust.edu.pk
9   James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, UK; Qammer.Abbasi@glasgow.ac.uk
*   Correspondence: quratulain.mastoi@siswa.um.edu.my

Abstract: The Internet of Medical Things (IoMT) workflow applications have been rapidly growing in practice. These internet-based applications can run on the distributed healthcare sensing system, which combines mobile computing, edge computing and cloud computing. Offloading and scheduling are the required methods in the distributed network. However, a security issue exists and it is hard to run different types of tasks (e.g., security, delay-sensitive, and delay-tolerant tasks) of IoMT applications on heterogeneous computing nodes. This work proposes a new healthcare architecture for workflow applications based on heterogeneous computing nodes layers: an application layer, management layer, and resource layer. The goal is to minimize the makespan of all applications. Based on these layers, the work proposes a secure offloading-efficient task scheduling (SEOS) algorithm framework, which includes the deadline division method, task sequencing rules, homomorphic security scheme, initial scheduling, and the variable neighbourhood searching method. The performance evaluation results show that the proposed plans outperform all existing baseline approaches for healthcare applications in terms of makespan.

Keywords: ethereum security; privacy; smart contract; rules; distributed

1. Introduction

Nowadays, the usage of medical devices based on the Internet of Medical Things (IoMT) network to deal with healthcare issues has been growing progressively [1]. The IoMT is a network that is composed of medical sensors, wireless technology and distributed cloud computing technologies [2]. Therefore, the combination of IoMT and healthcare devices can improve the quality of human life and provide better care services and create a more cost-effective system [3]. Recently, many IoMT-based applications have been developed

by different shareholders to deal with other diseases, such as E-healthcare sensor-based applications and E-real-time-healthcare applications [4]. These healthcare applications have different classes, such as workflow or fine-grained classes, while providing services to the users [5]. The IoMT workflow is a complete process where all tasks are dependent on each other. At the same time, fine-grained-based applications have only one independent task process. For instance, a heartbeat task or ECG monitoring task [6].

As mentioned above, the IoMT is a distributed network that combines different technologies such as medical devices, wireless technology, and cloud computing. These healthcare applications are distributed and implemented in different places (e.g., fog cloud nodes). However, they execute onto geographically distributed network [7]. One of the main concerns is resource-constraint issues in devices (e.g., limited storage, small CPU, and short bandwidth utilization) and cannot execute applications locally [8]. Offloading is a way that migrates compute-intensive tasks of applications to distributed computing nodes for further execution via wireless technologies (e.g., WiFi, LTE, and others). Cloud computing is a crucial paradigm in the IoMT network, which allows devices to offload their compute-intensive and rich-resource computing for execution [9]. However, each application has different kinds of tasks, e.g., a set of security tasks, delay-sensitive tasks, and delay-tolerant tasks). Therefore, except for delay-tolerant tasks, the rest of them cannot be used on the remote cloud due to long latency and security risks. Fog computing is an extension of cloud computing which brings remote capabilities of the cloud at the edge of wireless network which is in proximity to users devices [10]. However, all delay-sensitive tasks can perform fog computing, but security tasks have security risks due to open wireless networks. Generally, one of the best solutions is offloading a set of security risk tasks that often perform at the local device and sending their data to the cloud for further execution [11]. The current cloud computing model offers services based on demand with some time limitations, for instance, weekly, monthly, yearly, with a set of constraints. Every time, the users do not use cloud services. However, services are still paid for due to the rent of services in advance [1]. Serverless computing is a new cloud cost model in which the cloud provider provides machine resources on-demand and manages the servers on their customers' behalf. Serverless computing does not store resources in volatile memory; instead, it performs the computation in brief bursts, with the results saved to a disc. There are no computer resources given to an application when it is not in use. The cost of an application is determined by the number of resources it consumes. It may be a type of utility computing. In the sense that cloud service providers still employ servers to provide code for developers, "serverless" is a misnomer. On the other hand, developers of serverless applications are not concerned with the container, VM, or physical server capacity planning, configuration, management, maintenance, fault tolerance, or scaling. In the literature, many studies [2,3,5,7,8,11,12] proposed different systems, architectures, and heuristics-based IoMT network to solve the offloading and scheduling problem of workflow and fine-grained applications. These studies improved devices' battery lives, offloading cost and the obtained Service Level Agreement (SLA) and Quality of Service (QoS) applications during their problem formulation. They optimized the energy, makespan and cost of applications. However, many challenges still exist which are needed to be overcome in the IoMT network. (i) All existing efforts only consider either a workflow application or fine-grained application at any one time. However, modern devices can provide support to both simultaneously, but existing IoMT systems cannot. (ii) Current studies only focus on offloading efforts; they widely ignore the performance of remote computing nodes such as fog and cloud, which greatly impacted users' performances and cost. For instance, existing fog and cloud are offered on-demand as-you-pay and services are informed on virtual machines. This resource-provisioning model is very costly for fine-grained tasks where they need only usage-based services instead of monthly and yearly contract-based services. (iii) Existing resource allocation policies of theses studies [2,3,7,9,10,12–14] in the IoMT network only make offloading decisions at the user level and based on the threshold. However, optimal scheduling and failure aware mechanisms at the distributed computing

node level are widely ignored. Due to this, the failure ratio of tasks and deadline of applications are missed over a wide range. This paper proposes a novel scheduling system for mixing fine-grained and workflow IoMT tasks in distributed and virtual machine-based mobile edge cloud networks to cope with the issues mentioned earlier. This work considers both workflow and fine-grained tasks simultaneously, the proposed serverless functions and a virtual machine aware service environment in a distributed mobile edge cloud network. The function as a service (FaaS) is a cost-efficient model for fine-grained tasks which pay for the execution of tasks rather than provisioning for monthly or yearly for not using tasks. For the workflow tasks, the virtual machine-based solution has been proposed in the system. Each application is divided into three types of tasks: security tasks, delay-sensitive tasks, and delay-tolerant tasks. The proposed IoMT system consists of different paradigms such as mobile computing, fog computing, and cloud computing based on task types. The study's goal is to minimize both makespans of applications and the cost of the system during problem formulation. In summary, this paper makes the following contributions to solve the scheduling problem.

- Initially, the study devises the mathematical model of hybrid work of IoMT applications with many objectives. The hybrid workloads consist of fine-grained and workflows tasks, and the multi-objective functions are makespan, cost, and energy consumption. Each objective function has different weights to optimize the IoMT for each application;
- The study devises three-phase level scheduling methods such as deadline-efficient, cost-efficient and energy-efficient ones in the IoMT system to optimize the overall system in the network;
- To maintain the security requirements of fine-grained and workflow workloads, the fully homomorphism Encryption (FHE)-enabled security is suggested to ensure the security in IoMT for all applications;
- To optimize all objectives together, the study devises the deep graph convolutional network-enabled weighting scheme to boost and optimize the study's overall nonlinear objective functions in different convolutional networks.

The sections are organized as follows. Section 2 presents existing studies related to the considered problem. Section 3 defines all steps of the problem formulation. Section 4 outlines the proposed methods and their solutions. Section 5 illustrates the performance evaluation of techniques on different workflow benchmarks. Section 6 presents the conclusion of the paper.

2. Related Work

For many years, Internet of Medical Things (IoMT) frameworks or systems have gained significant traction in various medical sectors. Many different sorts of healthcare workloads are taken into account in the IoMT to tackle various scheduling and offloading issues. Workflows, applications, services, and fine-grained and coarse-grained models are examples of workloads. The IoMT consists of different heterogeneous computing nodes where a connection between computers or computer programmes is known as an application programming interface (API). It is a form of software interface that provides a service to other programmes. An API specification is a document or standard that defines how to create such a connection or interface. An API is implemented or exposed by a computer system that meets this standard. The term API can be used to refer to either the specification or the implementation. As a result, as indicated in Table 1, there are different multi-objective techniques for each workload, each with its own set of restrictions and security requirements.

Table 1. Existing offloading methods.

Study	Workload	Constraint	Method	Environment	Security
[1]	Workflow	Deadline	Weighting	Remote Cloud VMs	RSA
[2]	Coarse-Grained	Resource	Constraint	Edge VMs	DES
[3]	Independent	Budget	Goal-Programming	Cloudlet VMs	CRC32
[4]	Fine-Grained	QoS	Min-Max	Edge VMs	DES
[5]	Workflow	Deadline	VEGA	Fog VMs	RSA
[6–8]	Workflow	Deadline	GA	Remote Cloud VMs	RSA
[9–14]	Coarse-Grained	Lateness	PSO	Fog Container	RSA
[15–20]	Coarse-Grained	Cost	Ant-Colony	Fog Container	MD5
[21]	Application	Deadline	Weighting	Remote Cloud VMs	Athentication
[22]	Services	Deadline	Evolutionary	Remote Cloud VMs	Authorization
[23]	API	Deadline	GA	Fog VMs	RBS
[24,25]	Model	Cost	Min-Max	Open Fog Cloud VMs	3-DES
[26]	Workflow	Tardiness	MOEA	Edg-Cloud VMs	SHA-256
[27]	Workflow	Deadline	NSGA-II	Edge-Cloud VMs	SHA-32
Proposed	Workflow, Fine-Grained	Deadline, Cost, Energy	DGCN	Cloud VM, Fog Functions	FHE

In the study of [1], the workflow application, deadline constraint, weighting method, and remote Cloud VMs along with the RSA security mechanism aware IoMT system are suggested. The objective is to offload healthcare data with their deadlines on the cloud servers. The study of [2] suggested IoMT-based coarse-grained healthcare workloads, with resource constraints and the programming aware constraint method on latency optimal edge nodes. The study implemented a DES security mechanism for offloaded workloads in the system. The goal is to minimize end to end latency. The study of [3] suggested IoMT based on the independent healthcare workload, budget objective, goal-programming multi-objective method, latency optimal cloudlet deployed virtual machines and CRC32 security mechanism. The aim is to minimize end to end latency. Refs. [4,5] suggested an IoMT system based on workflow that is fine-grained and quality of service (QoS) aware, as well as a min-max multi-objective method and distributed edge implemented virtual machines and DES and RSA security for healthcare applications. The goal is to minimize resource consumption and energy and delay the objectives of the study. The vector evaluated genetic algorithm (VEGA) allows dealing with multiple objectives. However, the min-max algorithm only achieved good results with the single constraints in nondominant solutions on the Pareto frontier.

The studies [6–8] devised dynamic and secure IoMT systems based on different primitives such as workflow applications, deadlines, Genetic Algorithm (GA) on virtual machines (VMs), which enable cloud data centers, and RSA-based networks. The purpose of these studies is to gain dynamic results for healthcare applications in distributed cloud data centers. The studies [9–14] proposed IoMT with a tuple of implementations, such as coarse-grained workload, and optimized the objectives' lateness and energy with particle swarm optimization schemes in distributed RSA-enabled fog virtual machines. Particle swarm optimization (PSO) is a computational method for solving problems by iteratively improving a potential solution against a set of quality criteria. The message digest (MD5) enabled the secure distributed cloudlets and fog node aware IoMT systems suggested by [15–20]. The workload considered to be coarse-grained is solved by a multi-objective approach and ant-colony is done so with a dynamic approach. The objective is to minimize service cost, latency and delay of applications

in the IoMT system. The deadline, resource and lateness are considered during offloading and resource allocation in the system.

The applications, services, application programming interface (API) and model-based workload have been implemented in [21–27]. The virtual machines, container and serverless aware resources are offered during workload execution in the system. The min-max, Multi-objective Evolutionary Genetic Algorithm (MOGA) and NSGA-II-enabled multi-objective-based techniques suggested to solve the healthcare problems in distributed fog cloud nodes. The goal is to optimize different objectives with nondominance and dominance schemes with the Pareto frontier tool for different healthcare workloads. These studies considered the single constraint during decision in IoMT. The deep convolutional neuron network-enabled healthcare system is suggested in [28–31]. The goal is to handle multiple objectives such as energy, makespan, and cost of coarse-grained applications in the distributed IoT fog cloud network in the system. These studies suggested a dynamic heuristic based on reinforcement learning where the considered workload is only coarse-grained in the system for offloading and resource allocations.

To the best of our knowledge, a hybrid workload-enabled and secure healthcare monitoring sensing framework in a distributed fog cloud network has not been studied yet. The considered problem and system in the present study differ from existing works [1,2,18,22,28–31] in the following way. The proposed work considers the hybrid workloads such as workflow and the fine-grained model and the proposed mathematical model, whereas the study devises the functions and virtual machine aware fog cloud network which was not considered in the existing works. The main reason for this is that the research focuses on the cost-efficient scheduling and resource-optimal allocations of workload in the distributed fog cloud network. Therefore, in the considered problem, the study has three different conflicting objectives: makespan, lateness, and energy consumption with cost, deadline, and lateness constraints in the IoMT system. The existing multi-objective approaches cannot be applied to hybrid workloads in the IoMT because all existing objectives require a lot of decision time and resources to find optimal solutions for all objectives in IoMT. Therefore, the study considers the deep graph-based convolutional network-enabled algorithm framework to solve the supposed problem in the IoMT.

3. Proposed Architecture

The study proposes a new secure mobile edge cloud architecture to run IoMT workflow applications in a distributed environment. The proposed architecture consists of three main layers, the application layer, management layer and resource layer, as shown in Figure 1. The IoMT workflow applications layer consists of multiple applications where each application is composed of three different types of functions. The nodes, such as the blue node, show security tasks, the light node shows delay-sensitive tasks, and the red node displays delay-tolerant functions. The architecture initially takes the inputs of all applications into the management layer. The workflow tasks are annotated as the design time in different types, such as security tasks, delay-sensitive tasks, and delay-tolerant tasks. The execution time and energy consumption are anticipated in advance before scheduling tasks to any node by exploiting the energy profiler and workload execution profiler at the design time of applications. These mechanisms of application partitioning and the time estimation are already published in our previous work [7]. Therefore, this work only focuses on scheduling, not application partitioning and offloading in the current model.
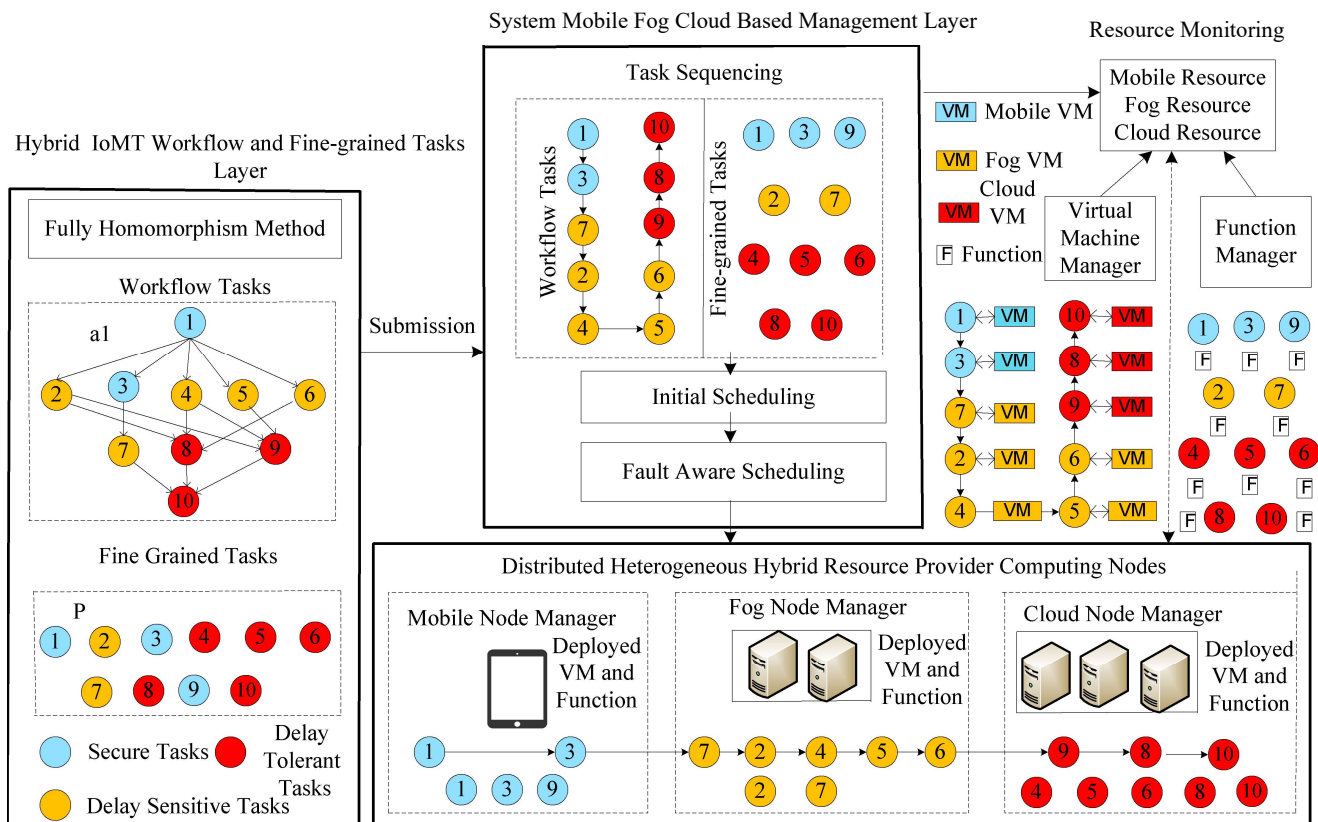
Figure 1. Hybrid workload-enabled and secure healthcare monitoring sensing framework in distributed fog cloud network.

The IoMT agent is an administrator in the management layer that processes the requested IoMT workflow applications $P = \{a1, \ldots, aP\}$. The IoMT agent consists of the following components: deadline division, task sequencing, homomorphic security scheme, initial scheduling and Variable neighborhood Searching (VNS). The deadline division divides the deadline d_a of all IoMT applications into task deadlines based on their execution time on different computing nodes. All the tasks are sorted based on their requirements by the sequencing rules. The rules are earliest due date and cost of resources. We assigned the priority to each task in the following way: The homomorphic security method encrypts and decrypts security tasks locally on the devices. The Denial of Service (DoS) and surfing profiling handles and identifies an attack in the network. This way, we can save resource and time before offloading and scheduling in the system. Initial scheduling maps all tasks based on sorting and security requirements onto heterogeneous mobile edge cloud computing efficiently. After that, a VNS-based searching method improves the initial solution from candidate solutions.

The resource layer consists of mobile computing, edge computing and cloud computing. Resource-constrained mobile computing only executes security annotated tasks with private keys. Furthermore, delay-sensitive tasks are carried out using edge computing which locates the edge of the network with ultra-low latency. Finally, all delay-tolerant tasks are carried out using cloud computing.

Table 2 describes the notation of the mathematical model of the considered problem.

Table 2. Mathematical notation of workflow and fine-grained tasks.

Notation	Description
P	The number of IoMT Workflow Applications
a	*ath* IoMT workflow application of A
d_a	The deadline of *ath* application
V	Total number of tasks
Va	Total number of tasks of application a
i	*ith* task of V
T_i^e	The execution time of a task i
T_i^c	The communication time of a task i
w_i	The workload of a task i
d_i	The deadline of a task i
S	Set of security tasks
L	Set of delay-sensitive tasks
DR	Set of delay-tolerant tasks
M	The heterogenous computing nodes (e.g., mobile edge cloud)
j	The *jth* computing node
ζ_j	The processing speed of computing node j
R	The resource of computing nodes
r	The resource particular r
N	Total number of task
T	Set of all tasks
t_i	The *tth* task of T
M	Number of virtual machines
v_j	*jth* virtual machines in a server
ζ_j	Speed of *jth* virtual machine
p_j	Power of *jth* VM
s_i	Size of each task
ET_i	Execution time of task t_i
fm	Faster machine
sm	Slower machine
$\overline{ET_i}$	Processing time of task t_i
$E_{i,j}^{fm}$	Task t_i execution time on the faster machine
$E_{i,j}^{sm}$	Task t_i execution time on the slower machine
$FT_{j,k}$	k^{th} task finish time on VM v_j
F_i	Finish time of t_i
F_i^{fm}	Task finish time on the faster machine
F_i^{sm}	Task finish time on the slower machine
$\overline{F_i}$	Task t_i estimate finish time
d_i	Deadline of task t_i
$E_{i,j}$	Consumed energy of task t_i on VM v_j
Ωv_j	Total execution time of a set of tasks on VM v_j
T_i^{slack}	Task t_i slack time

This study considered two types of application workloads with different processes such as fine-grained tasks and workflow tasks.

3.1. Fine-Grained Tasks

The healthcare fine-grained tasks have their data, deadlines, and required CPU per execution during the process. Each fine-grained task is isolated; it needs a separate function to run its operation. The fine-grained task shown in Figure 2 has three types of tasks: secure tasks, delay-sensitive tasks, and delay-tolerant tasks.
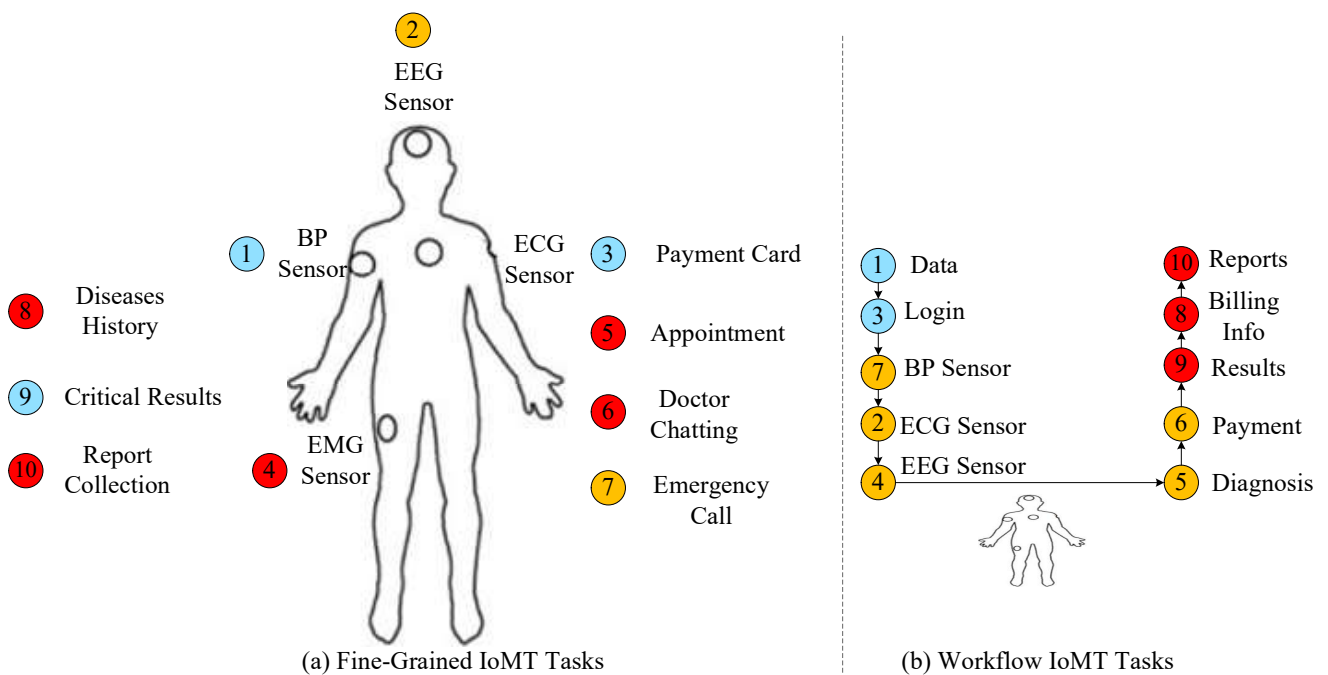
Figure 2. Hybrid IoMT workload of tasks.

The study makes the Problem Formulation of Workflow Tasks in the following way.

The paper investigates P number IoMT workflows applications, i.e., $\{a1, a2, \ldots, aP\}$. The directed acyclic graph, i.e., $a(V, E)$ illustrates constraint rules of applications, where i illustrates a particular task, and $e(i, j) \in E$ represents the communication nodes among different tasks. There are certain rules in IoMT tasks: (i) a task i should finish before starting task j. Furthermore, some tasks use original data and some of them have generated data. The notation w_i is an original datum of a particular task, and $w_{i,z}$ generated data of precedence during execution. Each IoMT application a categorized tasks into three lists: (i) security list $S = \{s_i = 1, \ldots, S \in Va\}$, delay-sensitive tasks (ii) $L = \{l_i = 1, \ldots, L \in Va\}$, and delay-tolerant tasks $DR = \{dr_i = 1, \ldots, DR \in Va\}$.

The present paper discusses the scenario of real-life healthcare IoMT applications, as shown in Figure 3. In the system model, there are three types of task lists: (1) secure tasks, i.e., $S = \{s_i = 1, \ldots, S \in Va\}$, which must be encrypted and decrypted locally by IoMT devices; (2) delay-sensitive tasks, i.e., $L = \{l_i = 1, \ldots, L \in Va\}$, which must be performed on edge nodes because of their latency requirements; (3) delay-tolerant tasks, i.e., $DR = \{dr_i = 1, \ldots, DR \in Va\}$, which are offloaded to the public cloud for execution. There are three layers in the system: the edge layer, where all organizations such as hospitals, clinics, and any medical centre use different IoMT devices to run IoMT applications. These applications secure data locally on their own devices, and delay-sensitive tasks are offloaded via an access point (e.g., wifi) to the edge layer for execution. Furthermore, via the internet, all applications offloaded their tasks to the public cloud. The categorized tasks we already defined in detail above.
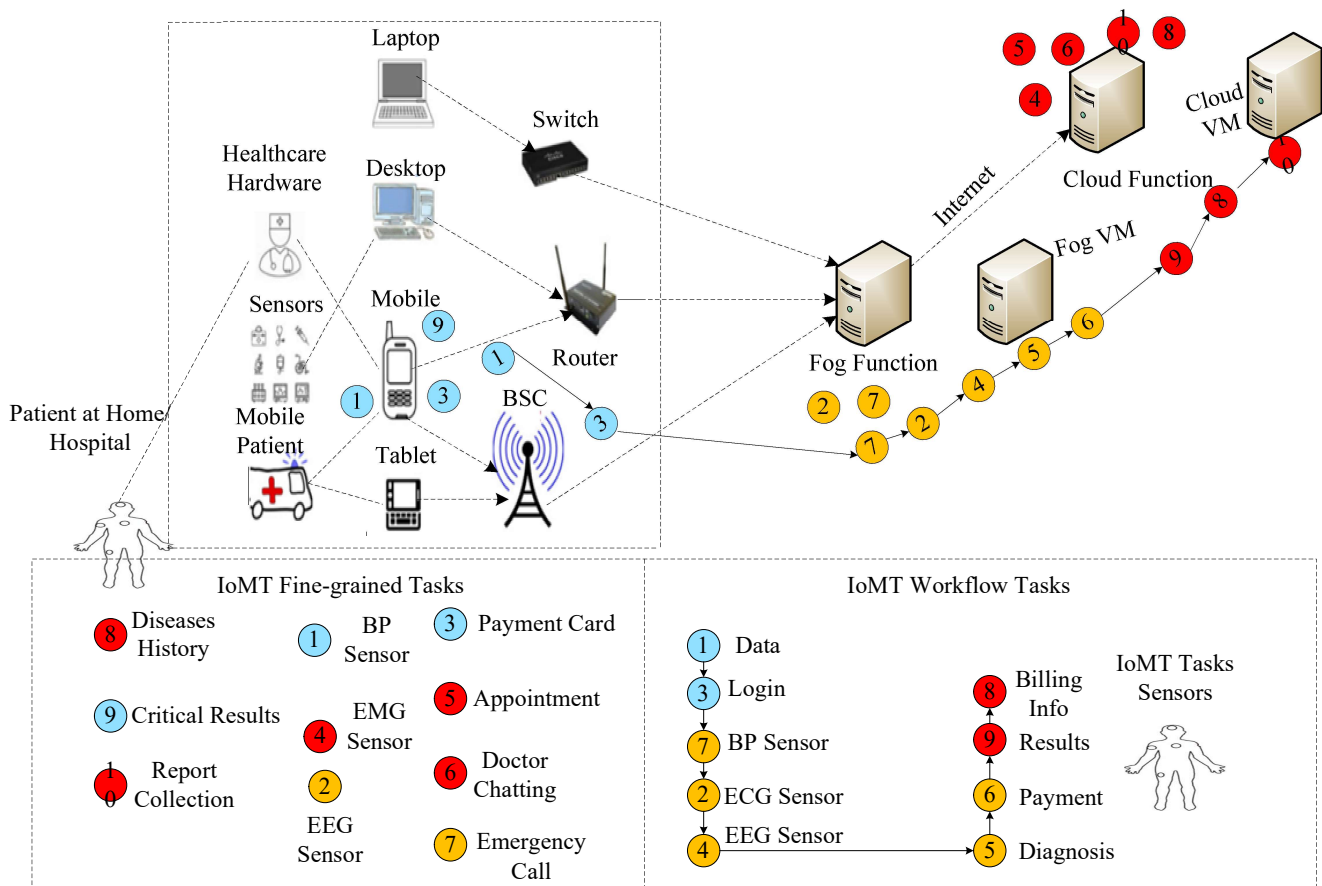
Figure 3. Hybrid IoMT tasks aware system.

The work takes M types of computing nodes, i.e., $M = \{j = 1, \ldots, jM\}$. They all are distinct by their features (e.g., speed and storage). Each computing node j has fixed homogenous virtual machines $\{VM = 1, \ldots, M\}$. We denote the speed factor of all computing nodes in the following way, i.e., $\zeta_j\{\zeta_j 1, \ldots, \zeta_j\}$, whereas the work shows the computing nodes resources in the following way, e.g., $R = \{r = 1, \ldots, rR\}$. We determine the execution time of a particular task in the following way.

$$T_i^e = \begin{cases} \frac{w_i}{\zeta_{j1 \times VM1}}, & y_i = 0, \\ \frac{w_i}{\zeta_{j2 \times VM2}}, & y_i = 1, \\ \frac{w_i}{\zeta_{j3 \times VM3}}, & y_i = 2. \end{cases} \tag{1}$$

In Equation (1), the vector $y_i = 0$ means the execution of a task in the local machine, e.g., $y_i = 1$. It implies the performance of a task on the edge and $y_i = 2$ executions of a task in the cloud.

IoMT workflow applications have relationship and communication time requirements due to transferring of data between them.

$$e_{\{i,j\}}^c = \begin{cases} Local, & z_i = 0, \\ LAN, & z_i = 1, \\ WAN, & z_i = 2. \end{cases} \tag{2}$$

Equation (2) determines the communication time between constraint tasks in workflow while sharing their data for execution.

$$T_i^c = \left(\frac{w_i}{B_w} \right) \times e_{\{i,j\}}^c. \tag{3}$$

If two tasks i and j are being carried out in the local machine, there is no communication time between them, i.e., $z_i = 0$. If two tasks i and j are running on an edge LAN network, then there is a fixed communication time between them, i.e., $z_i = 1$. Finally, if two tasks i and j are being carried out on a cloud WAN network, then there is fixed communication time between them, i.e., $z_i = 2$. The constraint (3) calculates the communication time between tasks i, j. We determine the finish time of a general task in the following way.

$$FT_i = \max_{v_j \in pred(v_i)} T_i^e + T_i^c. \tag{4}$$

Equation (4) calculates the finish time of a task.

We obtained the makespan of IoMT workflow applications as

$$MW = \sum_{a=1}^{P} \sum_{s_i=1}^{S \in Va} \sum_{l_i=1}^{L \in Va} \sum_{dr_i=1}^{DR \in Va} \sum_{ei,j}^{E} \sum_{j=1}^{M} \{FT_i(exit)\}, \tag{5}$$

MW denotes the makespan of all IoMT workflow applications in Equation (5).

The paper presents the problem mathematically as follows:

$$\min MW \tag{6}$$

which is subject to

Equation (6) shows the objective function of each application.

$$\sum_{a=1}^{P} \sum_{s_i=1}^{S \in Va} \sum_{l_i=1}^{L \in Va} \sum_{dr_i=1}^{DR \in Va} \sum_{ei,j}^{E} \{FT_i(exit)\} \leq \sum_{a=1}^{P}$$
$$d_a, \quad \forall\{a = 1, \dots P\}. \tag{7}$$

Equation (7) denotes the deadlines for completion of tasks of all applications.

$$\sum_{a=1}^{P} \sum_{s_i=1}^{S \in Va} \sum_{l_i=1}^{L \in Va} \sum_{dr_i=1}^{DR \in Va} w_i \leq \sum_{j=1}^{M} r \in R. \tag{8}$$

Constraint (8) shows all the requested workloads of applications that must not exceed the limits of resources during execution.

$$\sum_{i=1 \in Va}^{P} x_i = 0, 1, 2, \quad \forall\{j = 1, \dots, M\}, \tag{9}$$

Constraints ((9), (10), and (11)) show that each task to be assigned to one node, and each node can execute one task at a time when it is successfully assigned to any particular node.

$$\sum_{j=1}^{M} x_i = 0, 1, 2, \quad \forall \sum_{a=1}^{P} \{i = 1, \dots, Va\}, \tag{10}$$

$$x_i = \{0, 1, 2\}. \tag{11}$$

3.2. Problem Formulation of Fine-Grained Tasks

This study considers T number of fine-grained tasks, i.e., $\{t = 1, \ldots, T\}$. Each task has a workload, e.g., W_t and t_d deadline. The number of fog cloud functions is represented by $F = \{f1, f2, \ldots F\}$. Each function has a memory size of fm. The execution time of fine-grained tasks is determined in the following way.

$$\tau_T = \frac{t_w}{\zeta fm} \times j. \tag{12}$$

Equation (12) calculates the execution time of a task on the function in node j Therefore, the execution cost of all tasks is determined in the following way.

$$C = \sum_{t=1}^{T} \sum_{j=1}^{M} \sum_{f=1}^{F} \frac{t_w}{\zeta fm} \times \tau_t. \tag{13}$$

The functions can run on only computing nodes such as $j1$ to M. Therefore, the cost of the function is to be determined by the memory size and execution time as determined in Equation (13).

3.3. Energy Consumption Computing Nodes

This study determines the energy consumption due to virtual machines and particular function nodes. Therefore, j_w is the energy consumption per watt of node j to run virtual machines and functions. The power consumption of nodes is determined in the following way.

$$E = \sum_{t=1}^{T} \sum_{v=1}^{V} \sum_{j=1}^{M} \sum_{f=1}^{F} \tau_t \times j_w + T_i^e \times j_w. \tag{14}$$

Equation (14) determines the energy consumption due to both the workflow and fine-grained tasks in the computing nodes.

The study examines multi-objective problems such as energy, makespan, and lateness of both workflow and fine-grained jobs based on the suggested mathematical formula. As a result, multi-objective optimization is a subsection of multiple criteria decision making that deals with mathematical optimization problems that necessitate simultaneous optimizations of many objective functions. The Pareto frontier is used to construct the multi-objective problem. The Pareto frontier is an optimal technique for solving the problem restrictions since the study has conflicting aims in the suggested system with various resources. No one solution simultaneously optimizes each objective for a nontrivial multi-objective optimization problem. The objective functions are incompatible in this instance, and there are a (potentially infinite) number of Pareto optimal solutions. If there is no improvement in a single function value without deteriorating some of the other objective values, the key is nondominated, Pareto optimum, Pareto-efficient, or noninferior. All Pareto optimum solutions are deemed equally desirable without any additional subjective preference information. Many existing multi-objective optimization techniques from many problems suggested formulating and solving them. The goal could be to locate a representative group of Pareto optimal solutions, quantify the trade-offs in achieving several objectives, or identify a single solution that satisfies a human decision maker's subjective preferences (DM).

$$\min Z = MW + C + E. \tag{15}$$

Equation (15) shows the objective functions of both workflow and fine-grained tasks.

4. Proposed Security-Efficient Optimal Solution (SEOS) Algorithm Framework

This work considers the IoMT workflow applications, where each application has three types of tasks: security tasks, delay-sensitive tasks, and delay-tolerant tasks. We analyze the heterogeneous computing nodes (e.g., mobile node, edge node and remote

cloud node) that are distinct by their speeds and resources. The advised problem is secure offloading and scheduling for IoMT workflow applications in heterogeneous computing nodes. This section proposes the Security-Efficient Offloading and Scheduling (SEOS) algorithm framework, which consists of different components to solve the considered problem. Initially, we divide the applications into task deadlines. In the second part, we sort all tasks into topological order based on the proposed three sequence rules. The third-party offloading-based homomorphic encryption method encrypts and decrypts security tasks locally on the local devices. Due to precedence constraint requirements, the cypher-text data of tasks are offloaded to the edge cloud for delay-sensitive tasks. The edge node applies computation on cypher-text instead of converting it into plaintext. The final part is local searching-based task scheduling, where all tasks are scheduled in different computing nodes. We explain the SEOS framework steps in the following algorithm, Algorithm 1.

Algorithm 1: SEOS.

Input : $P = \{a = 1, \ldots, aP\}, M = \{j = 1, \ldots, jM\}, \{f = 1, \ldots, F\},$
$\qquad \{t = 1, \ldots, T\};$
Output: min Z;

1 **begin**
2   **foreach** *(a to P & t = 1 to T)* **do**
3   Call Deadline Division Methods for workflows applications;
4   Prioritizing of Workflow and Fine-grained;
5   Call Secure Homomorphic Offloading Method;
6   Call Scheduling Method;
7   Calculate $Z \leftarrow a \leftarrow j$ based on Equation (6);
8   Call Search $Z \leftarrow Z^*$;
9   Call DGCN Scheme;
10   **return** Z^*;
11 End-Loop;

All steps of Algorithm 1 are detailed as follows:

- Initially, the algorithm takes the input of all IoMT applications and computing nodes;
- Divide the deadline of all applications into task deadlines;
- Sort all tasks based on the proposed topological sequences rules;
- Use the security method to encrypt and decrypt tasks locally;
- Use the scheduling method to search and schedule all tasks with the optimal makespan.

4.1. Deadline Division

The deadline division is a way to divide the application deadline into task deadlines; this way, we can achieve the quality of tasks based on their deadlines. For example, we split the applications into the following form.

$$ratio = \sum_{a=1}^{A} \frac{d_a}{Z}, \tag{16}$$

$$\sum_{a=1}^{A} \sum_{i=1}^{Va} T_i^{'e} = T_i^{e} \times ratio, \tag{17}$$

$$T_i^{'c} = T_i^{c} \times ratio, \tag{18}$$

$$d_i = \min(\{i_{dj}\}) - T_i^{'e}(i_j) - T_i'(c)$$
$$\forall i = 1 \in Va \exists j \in successor(i). \tag{19}$$

Initially, we obtained the ratio of all applications based on Equation (16), which determines the division of the deadline of each application with the makespan of the application. This way, we assigned the deadline to each task based on the execution time and communication based on executions ((17)–(19)).

Algorithm 2 divides the deadline of all applications into task deadlines to obtain the optimal makespan of each application onto heterogeneous computing nodes.

Algorithm 2: Deadline division method.

> **Input** : $Z, P = \{a = 1, \ldots, aP\}, d_a \in A;$
> **1 begin**
> **2**   **foreach** *(a = 1 ∈ P)* **do**
> **3** $ratio = \frac{d_a}{Z} \in a;$
> **4** $T_i^{'e} = T_i^e \times ratio;$
> **5** $T_i^{'c} = T_i^c \times ratio;$
> **6** $d_i = \min(\{i_{dj}\}) - T_i^{'e}(i_j) - T_i'(c);$
>
> **7 End-Loop;**

4.2. Task Sequencing

In this section, we introduce task sequencing rules based on the following methods. Earliest Due Date (EDD) is exploited to order the tasks in a deadline manner. Each task is prioritized via Equations (20) and (21). Smallest Process First (SPF) is exploited when the smallest processing task is assigned the highest rank and scheduled before the longest process task. Smallest Slack Time First (SSTF) method shows the remaining time between finish time and the actual deadline should be smaller when a task i is scheduled on the same paradigm. We assigned the priority to each task in the following way:

$$Priority(v_i) = data_i + \max_{v_j \in succ(v_i)} Rank(v_j), \tag{20}$$

$$Priority(v_i) = data_i, v_i \in V. \tag{21}$$

We assume that the w_i is equal to whether ordinal data or generated of the task i during priority assignment. Both Equations (20) and (21) define the priority of all tasks from entry the task i to exit V by considering all predecessors and successors of the given application. Initially, we sort the topological priority of tasks in the following way.

- All workflow tasks sort out by descending order by their deadlines;
- All fine-grained tasks sort out by their deadlines.

We tried all sequences during initial task scheduling until submitted tasks are satisfied by the given requirements.

4.3. Security Aware Offloading Method

The Homomorphic Encryption [32] is a tool that allows computation on encrypted data of tasks. In this way, data of tasks remain private and confidential during offloading and scheduling at heterogenous mobile edge cloud networks. FHE encourages security-sensitive applications to work with sensitive data in untrusted environments. The geographically distributed computation and heterogeneous mobile edge cloud networking; secure communication is a good indication related to the applications.

When the data transfers to the cloud, we use standard encryption methods to secure the operations and the data storage. Our basic concept was to encrypt the data to the cloud provider before sending it back. However, at every transaction, the last one has to decrypt data. Therefore, the client will need to provide the server (cloud provider) with the private key to decrypt data before executing the required calculations, affecting the confidentiality and privacy of data stored in the cloud.

The secure homomorphic Algorithm 3 takes as input a list of security tasks *S*, which is annotated at the time of design. Algorithm 3 has the following steps. Firstly, it encrypts all security tasks of all applications locally and offloads them for assistance to external computing nodes. Secondly, the computing nodes apply the ciphertext instead of plaintext and then return them to the corresponding end-user devices. Finally, it will decrypt the results of encrypted tasks locally on the computers.

Algorithm 3: Secure homomorphic offloading.

Input : All security tasks $(w_i, Si = \{i = 1, \ldots, S \in Va\})$;
Output: $DoS = 0$

1 **begin**
2   **foreach** *(i = 1 S)* **do**
3      $DoS \leftarrow 1$ Denial of Service Attack present;
4      $p \leftarrow$ large integer;
5      $q \leftarrow$ large integer;
6      $p \neq q$;
7      $n \leftarrow p \times q$;
8      $\phi(n) = (p - 1) \times (q - 1)$;
9   **if** *(DoS = 0)* **then**
10   Choose an integer ϵ when $1 < \epsilon\phi(n)$;
11   **if** *($T_i^e + T_i^c \leq d_i$)* **then**
12   Offloader Engine ;
13          $e_c(w_i \leftarrow s_i) = s_i^b \bmod n$;
14   **else if** *($e_c(w_i \leftarrow s_i)$! empty)* **then**
15   Decrypted all tasks;
16          $d_c(i_y) = i_y^a \bmod n$;
17   End inner loop;
18   **else**
19   DoS=1;
20   Waiting for offloading;
21 End Main;

In this paper, we suggest implementing a system for performing operations on encrypted data without decrypting them, which will produce the same results after calculations as if we were directly operating on the raw data. Homomorphic encryption systems are exploited to execute encrypted data operations without understanding the private key (without decryption); the client is the sole owner of the secret key.

In the considered problem, the study considered the homomorphic encryption in the following condition. If Enc (a) and Enc (b) are used to estimate Enc (function (a, b)), where function can be: +, x, and outwardly practicing the private key. Moreover, additive homomorphic encryption considers raw data additions is the Pailler.

$$\sum_{v=1} V \sum_{a=1}^{P} D_{ci}(E_{ci}(n) \times E_{ci}(m)) = n \times m \; OR \; Enc(s_i \oplus s_j) =$$

$$Enc(s_i) \oplus Enc(s_j).$$

$$\sum_{t=1}^{T} D_{ct}(E_{ct}(n) \times E_{ct}(m)) = n \times m \; OR \; Enc(s_i \oplus s_t) =$$

$$Enc(s_t) \oplus Enc(s_t).$$

(22)

Equation (22) is called additive homomorphic encryption.

$$\sum_{v=1}^{P} V \sum_{a=1}^{P} D_{ci}(E_{ci}(n) \times E_{ci}(m)) = n + m \ OR \ Enc(s_x \oplus s_y) =$$

$$Enc(t_x) \oplus Enc(t_y).$$

$$\sum_{t=1} TD_{ct}(E_{ct}(n) \times E_{ct}(m)) = n + m \ OR \ Enc(s_x \oplus s_y) = \tag{23}$$

$$Enc(t_x) \oplus Enc(t_y).$$

In contrast, Equation (23) is multiplicative homomorphic encryption. An algorithm is fully homomorphic if both properties are satisfied simultaneously.

For the multiplicative homomorphic encryption, let us assume that $n = pq$, where p and q integer primes. Then, we choose a and b keys such that $ab = 1$, i.e., $(mod \ \phi(n))$. In contrast, b and n represent the public key, and p, q, and a denote the private key. We encrypt sensitive tasks in the following way.

$$e_c(si_x) = \sum_{v=1}^{V} \sum_{a=1}^{P} si_x^{ib} \ mod \ n.$$

$$e_c(si_x) = \sum_{t=1}^{T} si_x^{b} \ mod \ n. \tag{24}$$

$$d_c(si_y) = \sum_{i=1}^{N} si_y^{a} \ mod \ n. \tag{25}$$

Equations (24) and (25) show the encryption and decryption of a task. We suppose that s_i and s_j are plaintexts of task i and j; then, we denote as follows

$$e_c(s_i) \ e_c(s_j) = s_i^{b} \ s_j^{b} \ mod \ n = (s_i, s_j)^{b} mod \ n = e_c(s_i, s_j). \tag{26}$$

We define the FHE security scheme in Algorithm 3 as follows:

- Let us assume the algorithm takes the s_i and s_j inputs as the security tasks, and they require encryption locally on the IoT devices;
- p and q are long integers exploited during the encryption round. n is cross multiplication during block-switching performed from lines 2 to 6;
- ϵ is a small positive number employed for variation in the ordinal 64-bit block of encryption. At the same time, gdc and mod functions perform the fully homomorphic operation;
- The algorithm performs encryption on security tasks from lines 7 to 15. The list was added after all were encrypted after applying the security mechanism. The offloader engine is a method used inside devices which offloads the ciphertext of tasks to the system for further computations. Once the calculation was practiced on ciphertext, and the result was sent back to the devices, and they all decrypted on devices with their private keys;
- DoS is the profiling that identifies denial of service in the system; if it is 1 it means there is a risk of attack else, otherwise it will remain zero.

4.4. Initial Task Scheduling

The initial scheduling is not the final scheduling of all tasks, and they can reschedule heterogeneous mobile edge cloud networks (e.g., heterogeneous computing nodes). The initial scheduling depends upon the deadline division component, task sequencing and security scheme. We propose the iterative scheduling algorithm, Algorithm 4, which shows the process of scheduling tasks under their requirements.

Algorithm 4 performs the scheduling in the following way:

- Initially, the algorithm conducts deadline division, which shows the deadline of each task;
- All tasks schedule based on given sequences by sequence rules methods;
- All local tasks are encrypted and decrypted by the homomorphic security method and executed locally in the devices;
- The delay-sensitive tasks are scheduled at the edge node; this is necessary for all nodes, and the requested workload must be less than their resources during processing;.
- All delay-tolerant tasks are to be scheduled at the public cloud for execution;
- The algorithm iteratively allocates all tasks to heterogeneous computing nodes and calculates the makespan of each IoMT workflow application at initial scheduling.

Algorithm 4: Deadline-efficient scheduling.

    Input : $Z, P = \{a = 1, \ldots, aP\}, d_a \in P, t = 1, T, f = 1, F$;

1 **begin**

2   **foreach** *(j = 1 in M)* **do**

3   Call Deadline Algorithm 2;

4   call ordering and prioritizing of tasks;

5   **foreach** *(s_i in S $\in$ Va)* **do**

6   **if** *($s_i \leq d_i$ & $w_i \leq r \in j$)* **then**

7   Calculate execution time of local tasks based on Equation (4);

8   Call FHE Scheme;

9   Optimize objection based on Equation (6);

10   calculate the objective function in the following way;

11                 $Z \leftarrow v_i \leftarrow j$ & $t \leftarrow f$;

12                 $Z \leftarrow s_i \leftarrow r$;

13   **foreach** *(l_i in L $\in$ Va)* **do**

14   **if** *($l_i \leq d_i$ & $w_i \leq r \in j$)* **then**

15   Calculate execution time of delay-sensitive tasks based on Equation (4);

16   Optimize objection based on Equation (6);

17                 $Z \leftarrow l_i \leftarrow r$;

18   **foreach** *(dr_i in DR $\in$ Va)* **do**

19   **if** *($dr_i \leq d_i$ & $w_i \leq r \in j$)* **then**

20   Calculate execution time of local tasks based on Equation (4);

21   Optimize objection based on Equation (6);

22                 $Z \leftarrow dr_i \leftarrow r$;

23 End-Loop;

4.5. Searching Optimal Solution-Based VNS

The variable neighborhood Search (VNS) solves the initial scheduling when tasks are distributed and allocated to different computing networks. It traverses distant neighborhoods of the current obligatory solution, i.e., Z, and proceeds from beyond the new key if any improvement is made. Algorithm 5 is a global search iterative algorithm that improves the current solution with the new one via variable temperatures. If the temperature decreases, the makespan of applications reduces the initial schedule with the new key.

Algorithm 5 has the following steps to reach the optimal solution:

- The algorithm takes the initial cost of each application with the initial solution C;
- The temperature tmp is a variable whose initial value = 100; it reduces to near zero, as tmp minimizes the cost of each application minimizes;
- The set of candidate solutions, i.e., N, and C' is a new solution with available costs compared with the initial solution C;

- The Boltzmann constant, i.e., $rand(0,1) \leq e^{\frac{\Delta}{tmp}}$ is an acceptance method; it allows one to replace the original solution with a new one with the minimum exponential rate and temperature tmp. The rate of change in Δtmp temperature could be minimized or increased depending upon the situation;
- If the solution reached the maximum level, no furthermore improvement is made, then the algorithm accepts C^* as a final solution.

Algorithm 5: Cost-efficient VNS searching.

Input : C;
Output: C^*;
1 $C \leftarrow i = 1 \in Va \in P, t = 1 \in T$;
2 $f(C) \leftarrow$ Initial Solution ;
3 C^* Optimal solution;
4 α cooling parameter;
5 $tmp \leftarrow$ tempreture = 1000$\sim$500 ;
6 $iter \leftarrow 0$;
7 $max \leftarrow 10$ Maximum iterations;
8 **begin**
9 　 **while** *(tmp > 0)* **do**
10 　　 **while** *(iter ≤ max)* **do**
11 　　　 $C' \leftarrow$ randomly select neighbor solution $C' \in N(C)$;
12 　　　 $\Delta \leftarrow \frac{f(C')-f(C)}{tmp}$;
13 　　　 **if** *(Δ ≤ 0)* **then**
14 　　　　 $C \leftarrow C'$;
15 　　　　 **if** *(f(C') ≤ f(C))* **then**
16 　　　　　 Swap current solution with new one;
17 　　　　　 $C \leftarrow C'$
18 　　　 **else if** *(rand(0,1) ≤ $e^{\frac{\Delta}{tmp}}$)* **then**
19 　　　　 $C \leftarrow C'$;
20 　　　 **else**
21 　　　　 Stay with current solution;
22 　　　 $C^* \leftarrow C$;
23 　　　 $iter \leftarrow iter + 1$;
24 　　　 $tmp \leftarrow tmp \times (1 - \alpha)$;
25 　 End Conditions;
26 　 **return** C^*;
27 End Loop;

4.6. Energy-Efficient Scheduling

All the nodes are ordered according to the power consumption in the network. In the first step, both fine-grained and workflow applications are scheduled based on their deadlines. In the second step, all the tasks are rescheduled based on their execution costs. Finally, in the third scheduling, all nodes are rescheduled according to their power consumption to minimize their power consumption without violence or service quality applications. Algorithm 6 reschedules all tasks based on computing nodes' energy. In contrast, it is no matter if either the energy of node j is consumed due to virtual machines or functions for executing the workflow and fine-grained workload. Algorithm 6 ensures the energy-efficient scheduling without violating the deadline and cost of applications in the system.

Algorithm 6: Energy-efficient scheduling.

Input : $\{j = 1, \ldots, M\}$;

1 **begin**
2   **foreach** *(j = 1 in M)* **do**
3   Call Algorithm 4 verify the deadline of applications;
4   Call Algorithm 5 proves the cost of applications;
5   Apply Dynamic Voltage Frequency Scaling method to re-arrange the node according to their power consumption;
6   Calculate the power consumption of nodes based on Equation (14);
7   Schedule all workloads based on Algorithms 4 and 5;
8   End of Assignment until all workloads checked respect nodes energy consumption;

4.7. Multi-Objective Deep Graph Convolutional Network-Based Scheme

These days, for graph-structured aware applications, the usage of deep convolutional networks have become extremely popular. As a result, multi-objective decisions based on heterogeneous resources and parameters of applications can be made efficiently. However, early neural networks could only be implemented with regular or Euclidean data, even though many data in the actual world have non-Euclidean graph structures. The nonregularity of data structures has driven recent advances in graph neural networks. As a result, graph neural networks have developed different variations in recent years, with Graph Convolutional Networks (GCNs) being one of them. GCNs are also one of the most fundamental graph neural networks variations.

The study devises the weighting multi-objective nondominant schemes based on the deep graph convolutional network. Algorithm 7 shows the process of the proposed method with different steps. The algorithm has three layers: the input layer, deep convolutional layer, and output layer. According to the given scenario in Figure 4, the algorithm performs the following operations.

Algorithm 7: Multi-objective weighting scheme based on a deep graph convolutional network.

Input : $\{f = 1, \ldots, F, VM\} \leftarrow \{j = 1, \ldots, M\}, \{a = 1 \in P, t = 1 \in T\}$;

1 **begin**
2    $weight = \{0.1, 0.2, \ldots, 0.9\}$;
3   H^{l+1} deep convolutional layer;
4   **foreach** *(j = 1 in M)* **do**
5   The input model takes by model as the graph ;
6   The variable features x_j for individual node j;
7   Each deep convolutional network layer is the nonlinear function;
8      $H^{l+1} = Z \leftarrow MW + E + C$;
9   Calculate the workloads and functions optimization based on Algorithms 4–6;
10      $Z \leftarrow MW + E + C = \{H^{l+1} \times Weight, \ldots H^{l+N} \times Weight\}$;
11   **if** *($H^{l+1} = Z \leftarrow MW + E + C$)>0.6* **then**
12   Calculate the weight sum of all objectives should be optimal than existing weight in different convolutional layers;
13        $Z^* \leftarrow MW + E + C = \{MW.H^{l+1} \times Weight + E.H^{l+N} \times Weight + C.H^{l+N} \times Weight\}$;
14   End of Inner Optimization;
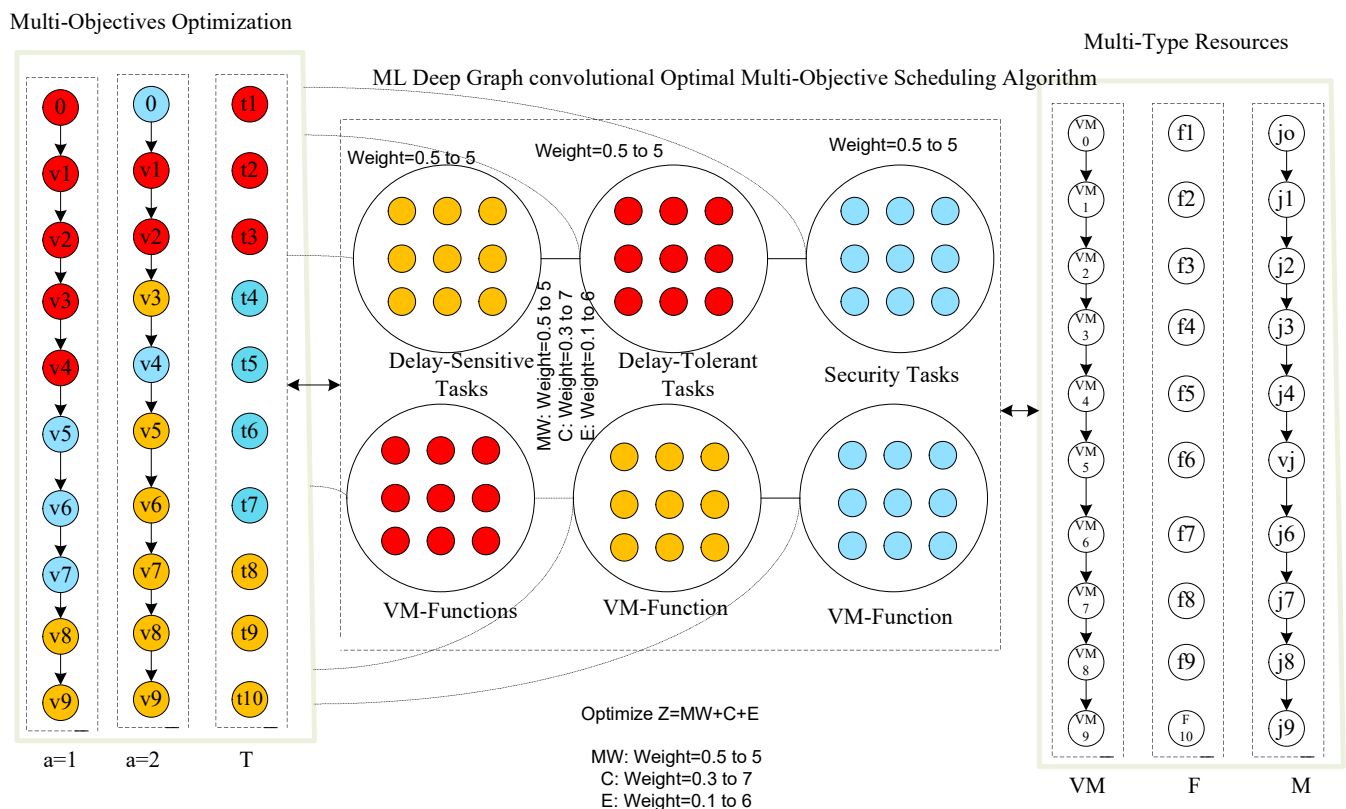15   End of sum optimization ;
16 End of main;

Figure 4. Multi-objective deep graph convolutional network-based scheme.

- In the first step, the workload of all applications after initial scheduling will be considered an input;
- All objectives have their weights concerning workloads and resources;
- The resources are virtual machines and functions which are assigned based on their cost function;
- The deep convolutional network chooses the best optimal weight of all objectives and sum them together. If the optimal weight is greater than the existing one, the multi-objective weight of all objectives is optimal, e.g., Z^*;
- All types of tasks such as delay-sensitive, delay-tolerant and security ones and their quality of service must be satisfied as defined in Figure 4.
- Every 10 min, the multi-objective tasks will call to optimize each objective function based on the available weights in the network;
- If the algorithm finds no further improvement, it will terminate the network with no further improvement in the system.

5. Performance Evaluation

This section shows the efficiency and effectiveness of the proposed work via the simulation results. Somehow, the simulation results are the same as a real-practice experiment in practice. The performance evaluation part consists of many sub-parts such as parameter setting, system implementation, component calibrations and result discussion. The paper explains sub-parts in detail to ensure an easy understanding of the experiment.

5.1. Parameter Settings

This subsection shows the experimental setup of the program configuration, languages and computing nodes as shown in Table 3. All parameters are included in the implementation part, such as programs and algorithms, in the JAVA, Python and YAML languages. There are three computing nodes configured for the proposed architecture. For instance, mobile node (e.g., HTC G17 and Samsung 1997), edge node (e.g., Intel 5 laptop,

AndroidX86 runtime), and cloud node (e.g., AndroidX86 Amazon). We repeated all experiments 50 times with different parameters. Table 3 describes the simulation parameters of the experiment.

Table 3. Simulation parameters.

Simulation Parameters	Values
Languages	JAVA, Python, YAML
Simulation Time	6 h
Experiment Repetition	50 times
Mobile devices	HTC G17 and Samsung 1997
Edge Cloud	Intel 5 laptop, AndroidX86
Public Cloud	AndroidX86 Amazon t2.medium
$z_i = 1$	10–30 ms
$z_i = 2$	100 ms
t	1000~500

Furthermore, we extended the computing nodes resource specification into a different table, Table 4. The main goal of this is to offer the computing capability and resource availability of each node in the system. There are three types of resources: a likewise mobile node, edge node and cloud node. All nodes are distinct by their speeds and resource specifications. All resources of different computing nodes are fixed, and they cannot scale up and scale during runtime in the implemented system.

Table 4. Heterogenous node resource specification.

Resource Type	Storage (GB)	Core	Speed (MIPS)
Public Cloud	20,000	1	10,000
Edge Cloud	50,000	1	5000
Mobile cloud	100	1	1000

5.2. Component Calibration

There are three main layers in the proposed architecture, as shown Figure 1. However, the application layer and system layer components are included in the calibration to evaluate the performances of the entire system. The features are secure offloading, task sequencing, and task scheduling. In addition, the Relative Percentage Deviation (RPD) was adopted to measure the performances of the components, as mentioned earlier, to run many types of IoMT workflow tasks in the system. The RPD measures in the following way:

$$RPD\% = \frac{Z - Z^*}{Z^*} \times 100\%. \tag{27}$$

Equation (27) shows the overall performances of all applications using distributed computing (e.g., mobile, edge and cloud nodes). The Z is the initial scheduling in the system; however, due to roaming features of applications, the initial solution of scheduling could be replaced with optimal scheduling Z^* during the searching for space in the solution. As we mentioned above, all answers are achieved via candidate solutions during global searching with limited iterations during the process. The $RPD\%$ is the difference between the initial and best solutions during the entire process.

5.3. Iomt Workflow Tasks and Fine-Grained Tasks

The study implemented both types of workloads such as workflow and fine-grained in the simulation configuration file.

Figure 5 shows the interfaces of the system with the results of workflow dag tasks graph during execution in the system.
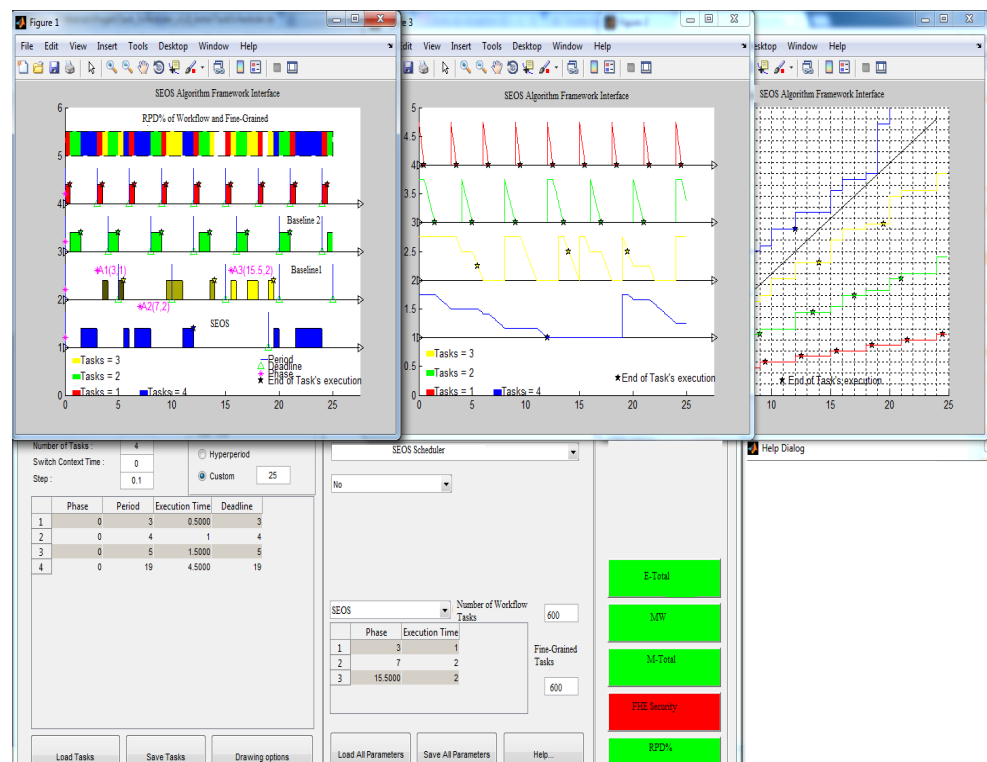
Figure 5. Workflow Interfaces of DAG Graph of different IoMT tasks.

All tasks are workflows; some have original data, and some share their data for processing. All tasks are constrained by their predecessors and successors in the system.

5.4. Workflow Tasks Generator

In this paper, we consider only three types of tasks. All workflow applications are real IoMT applications, which are open source and available at GitHub: https://github.com/OpenIoMeT/Iomet-wiki accessed on 1 July 2021. Initially, we analyzed all applications in DAG graphics with different types of tasks. The initial application is annotating notations (e.g., all types of tasks annotated at the design time). After that, we converted the IoMT workflow into a DAG graph, where blue nodes are security tasks (e.g., local tasks), light yellow nodes are edge tasks, and red nodes are remote tasks, and they have their execution time and communication time (e.g., ms and kb) due to precedence constraints.

5.5. Discussion of Results

This subsection compares the results of IoMT workflow tasks with the proposed framework with its components and existing offloading and scheduling frameworks. The discussion of component results starts with the following subsections.

5.6. Secure Offloading Performance

After the deadline division for each task, the security aware offloading applies security to the list of security tasks locally at the devices. We implemented fully homomorphic encryption and decryption methods that convert plaintext of security tasks into ciphertext in the application layer. Then, the offloader engine offloads those tasks to the system to be carried out further. The other performance means the ciphertext data of tasks are the inputs of different tasks in the system. Therefore, it is necessary, and we measured the accounts of the offloading method into two environments. The first environment is stable where there is no risk of hacking or Denial of Service (DOS) attacks; another environment is unstable where some chances of DoS exist in the network during offloading. In this case, we compared our proposed secure offloading schemes with the existing best security aware

offloading schemes, i.e., baseline 1 and baseline 2. In baseline 1, an RSA-based encryption method is implemented, which offloads tasks with encrypted data to the server, and then the server decrypts tasks with the key and performs computations. After the calculation again, the server encrypted tasks and sent them back to the devices, and then devices interpreted all tasks in the original form. This entire process is risky, and we can trust the untrusted cloud, and it is not good practice to leave essential data on the server.

Figure 6a,b show that the proposed component (e.g., secure offloading) of the SEOS framework outperforms in any environment compared to the existing secure offloading techniques concerning resources and performance. The main reason behind this is that all existing baseline approaches only consider the security and require resources; however, the proposed secure offloading method encrypted and decrypted all tasks based on their deadlines and availability of resources. Furthermore, before offloading to any nodes, we anticipated the available network which was either secure or not in the system. Our approach can stabilize and be unstable because we care about resource utilization, tasks' QoS and network stability before sending data to the surrogate edge or remote servers.
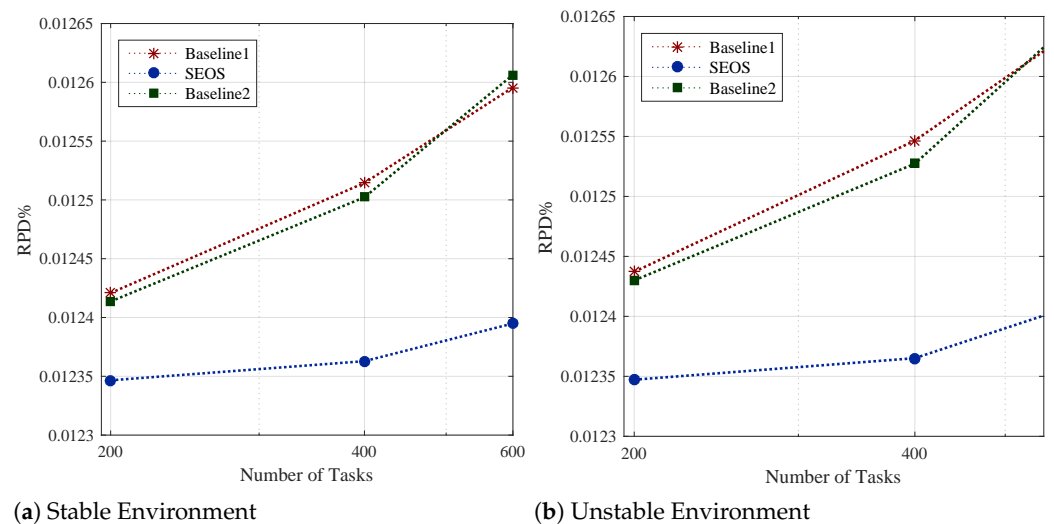

(**a**) Stable Environment   (**b**) Unstable Environment

Figure 6. Security aware performances.

A denial of service (DoS) outbreak happens whenever verifiable applications can not access their edge nodes or remote nodes resources for further execution due to either a cyber attack or network attack in the system. These nodes may be concerned by any attack and not able to respond. A denial of service attack may harm both resources and time even though tasks are encrypted. With this consideration, the proposed secure offloading method, including encryption decryption and deadline, detects and anticipates any attack before offloading via network monitoring and surfing profiling at the local device. It may save our resources and time during offloading in all kinds of environments. Therefore, Figure 7a–d show that the component of SEOS outperforms in terms of resource utilization and the deadlines of tasks, and identifies DoS in advance, in contrast to all existing approaches which considered only encryption and decryption and resources without deadlines and availability of DoS attack.
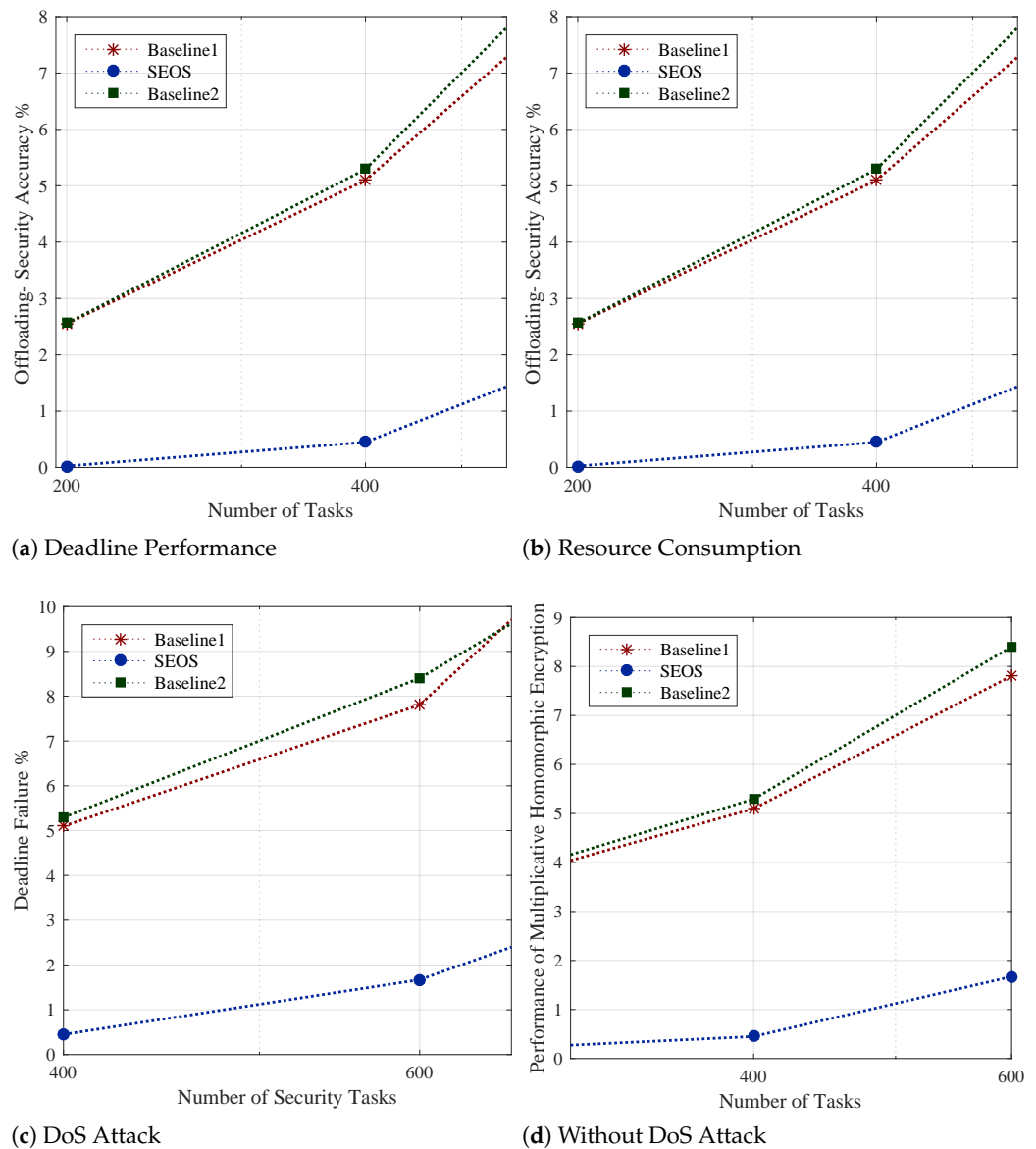
(**a**) Deadline Performance

(**b**) Resource Consumption

(**c**) DoS Attack

(**d**) Without DoS Attack

Figure 7. Security aware performances.

Figure 8a,b show that the proposed task sequence rules adopt initial sorting and dynamic sorting to maintain the deadline of tasks for the runtime. Therefore, it is necessary to execute all tasks under their deadlines with a minimum loss of generosity.
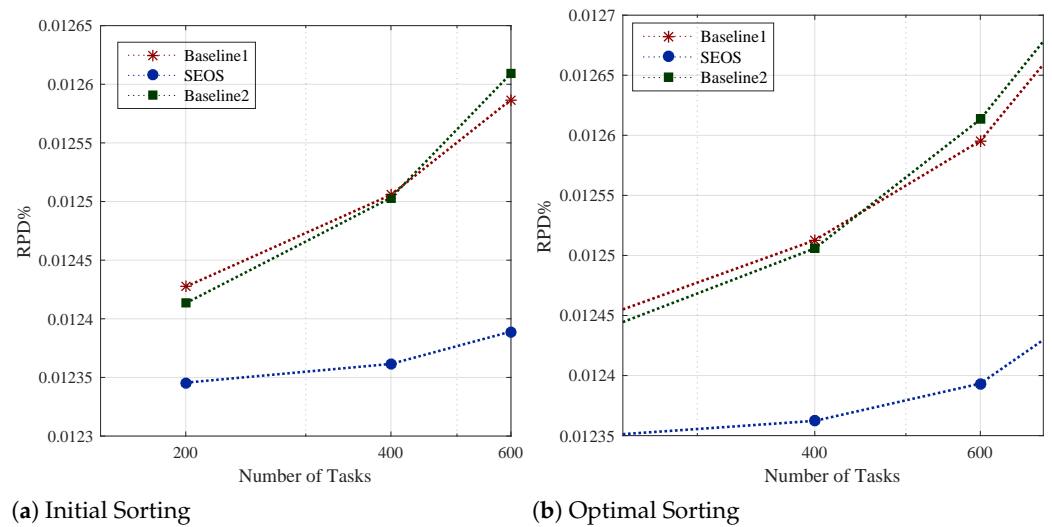
(**a**) Initial Sorting   (**b**) Optimal Sorting

Figure 8. Deadline aware performances.

5.7. Task Scheduling

Based on security-efficient offloading, sorting with different rules, task scheduling is the final phase where all tasks must be completed with precedence and deadline constraints. We set four flows of IoMT tasks with different numbers for scheduling. These tasks have different types, as we discussed above. The goal of the study is to minimize the makespan of all applications. We consider the four various applications with a different number of tasks. Each application has three different types of tasks and deadlines with constraint rules. Somehow, a few tasks are executed in parallel order, and few tasks are performed in the sequencing order; it depends upon the application order. We implemented Heterogeneous Earliest Finish Time (HEFT) and genetic algorithm (GA) as the baseline 1 framework, and Dynamic Heterogeneous Earliest Finish Time (DHEFT) and particle Swarm Optimization framework as baseline 2. These frameworks are widely investigated for traditional and mobile workflow applications in the literature. These frameworks offer different components to run mobile workflow applications in additional steps, such as task sequencing and scheduling. We ran all applications with other frameworks (e.g., SEOS, baseline 1 and baseline 2), the results of all applications with their objectives can be seen in Figure 9a–d. Each application has different requirements, such as security, latency, and resources to run its tasks. However, the SEOS outperforms all existing frameworks in terms of all makespans and the needs of all applications.
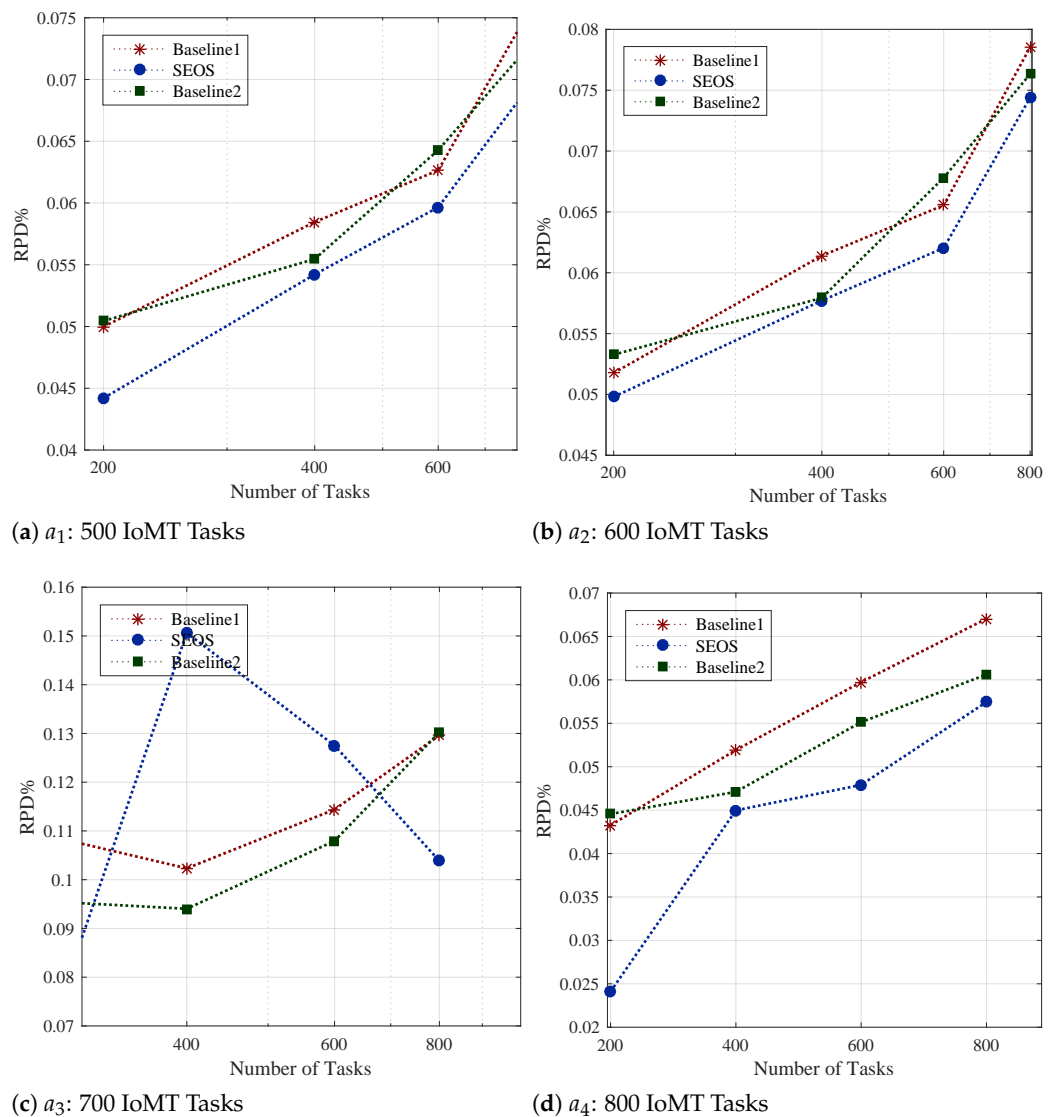
(**a**) a_1: 500 IoMT Tasks

(**b**) a_2: 600 IoMT Tasks

(**c**) a_3: 700 IoMT Tasks

(**d**) a_4: 800 IoMT Tasks

Figure 9. Makespan aware performances of all applications.

The main reason for this is that all existing algorithm frameworks have some races in the encryption and decryption format. They consume many resources and time to run different types of tasks (Figure 10): (i) Encryption of all tasks locally with the sharing key and offloading to the surrogate server for further execution. The server decrypts all tasks with a shared key and applies computation on plaintext instead of ciphertext. After the calculation, the server again encrypts tasks into ciphertext and send back their results. Furthermore, local devices decrypt the result into plaintext with the key. This way, the authentication, time and resources are challenging and uses at extending level. (ii) All existing studies partition the application into different types of tasks at the runtime based on various parameters (e.g., deadline, availability of resources, network contexts). However, due to the dynamic environment and load balancing situation in computing, these techniques benefit from lower running time and waste of resources. (iii) The loss of deadline and failure ratio of tasks in the system becoming very high. Therefore, the proposed SEOS partitioned the application at the design level to security, latency and resource requirements of all applications efficiently and ran them in the heterogeneous computing node during execution.
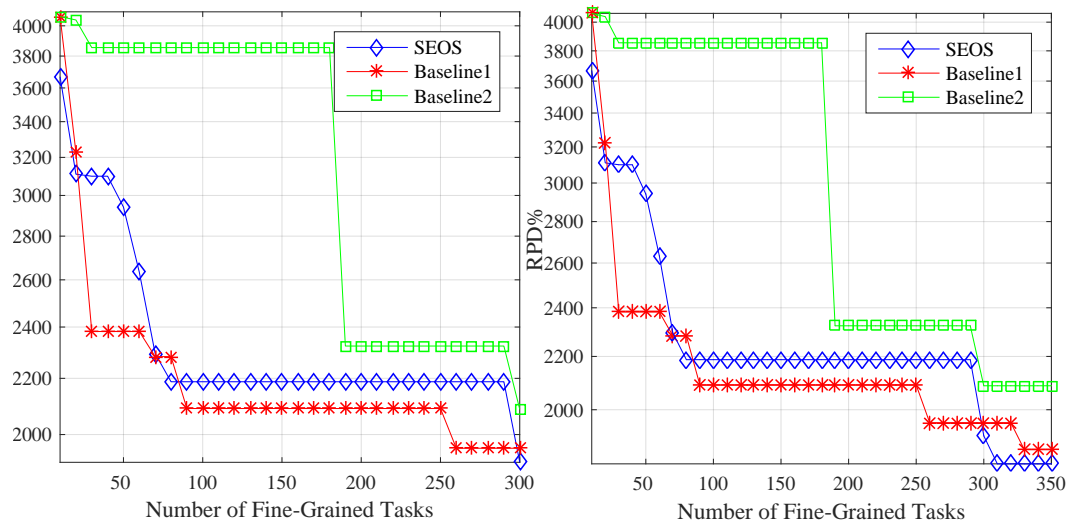
(**a**) a_1: 200 to 300 Fine-Grained Tasks.
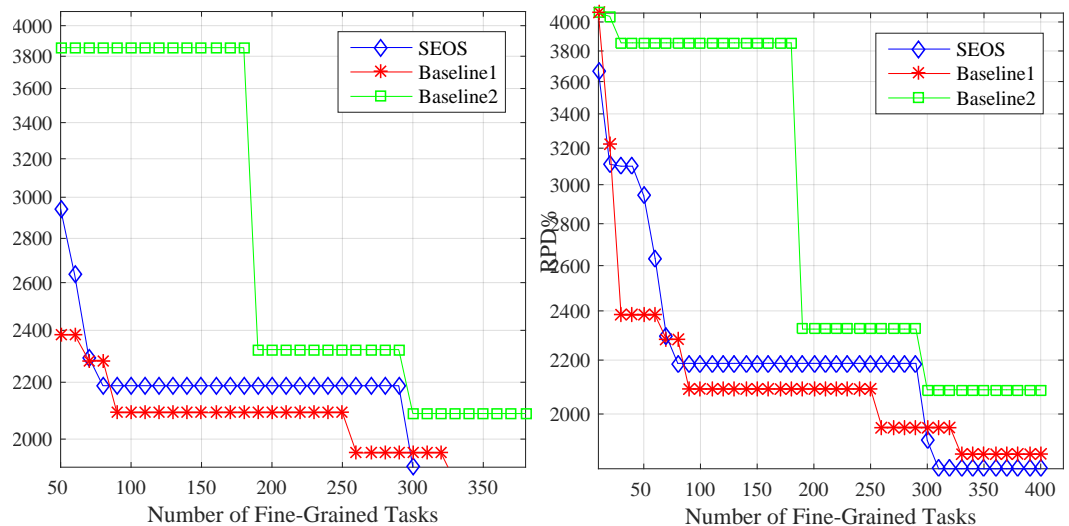
(**b**) a_2: 400 to 600 Fine-Grained Tasks.

Figure 10. Fine-grained tasks.

6. Conclusions

This work proposed a new healthcare architecture based on workflow applications based on heterogeneous computing nodes, consisting of different layers: an application layer, management layer, and resource layer. The goal is to minimize the makespan of all applications. Based on these layers, the work proposed the secure offloading-efficient task scheduling (SEOS) algorithm framework, which includes the deadline division method, task sequencing rules, homomorphic security scheme, initial scheduling, and variable neighborhood searching method. The performance evaluation results show that the proposed plans outperform all existing baseline approaches for healthcare applications in terms of makespan. The discussion of the results showed that the proposed idea and SEOS framework outperformed all IoMT applications' existing methods in heterogeneous computing nodes. The discussion of results and comparison has been made via different components based on HSD and ANOVA famous techniques. However, there are few things to be improved in the future.

This work did not consider the mobility aware offloading and scheduling for IoMT workflow in a heterogeneous computing node environment. The runtime uncertainty in the network contexts, load balancing, failure of tasks situation will be future work of our

study. We will design deep reinforcement learning architecture and framework, which will include policy, Q-deep learning, and different methods.

References

1. Ying Wah, T.; Gopal Raj, R.; Lakhan, A. A novel cost-efficient framework for critical heartbeat task scheduling using the Internet of medical things in a fog cloud system. *Sensors* **2020**, *20*, 441.
2. Lakhan, A.; Mohammed, M.A.; Rashid, A.N.; Kadry, S.; Panityakul, T.; Abdulkareem, K.H.; Thinnukool, O. Smart-Contract Aware Ethereum and Client-Fog-Cloud Healthcare System. *Sensors* **2021**, *21*, 4093. [CrossRef]
3. Hussain, M.; Wei, L.F.; Lakhan, A.; Wali, S.; Ali, S.; Hussain, A. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100517.
4. Liu, C.F.; Bennis, M.; Debbah, M.; Poor, H.V. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Trans. Commun.* **2019**, *67*, 4132–4150. [CrossRef]
5. Lakhan, A.; Li, X. Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks. *Computing* **2019**, *102*, 105–139. [CrossRef]
6. Marin, R.C.; Gherghina-Pestrea, A.; Timisica, A.F.R.; Ciobanu, R.I.; Dobre, C. Device to Device Collaboration for Mobile Clouds in Drop Computing. In Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 11–15 March 2019; pp. 298–303.
7. Lakhan, A.; Dootio, M.A.; Groenli, T.M.; Sodhro, A.H.; Khokhar, M.S. Multi-Layer Latency Aware Workload Assignment of E-Transport IoT Applications in Mobile Sensors Cloudlet Cloud Networks. *Electronics* **2021**, *10*, 1719. [CrossRef]
8. Memon, M.S.; Lakhan, A.; Mohammed, M.A.; Qabulio, M.; Al-Turjman, F.; Abdulkareem, K.H. Machine learning-data mining integrated approach for premature ventricular contraction prediction. *Neural Comput. Appl.* **2021**. [CrossRef]
9. Lakhan, A.; Sajnani, D.K.; Tahir, M.; Aamir, M.; Lodhi, R. Delay sensitive application partitioning and task scheduling in mobile edge cloud prototyping. In Proceedings of the International Conference on 5G for Ubiquitous Connectivity, Nanjing, China, 4–5 December 2018; Springer: Cham, Switzerland, 2018; pp. 59–80.
10. Lakhan, A.; Li, X. Content aware task scheduling framework for mobile workflow applications in heterogeneous Mobile-Edge-Cloud paradigms: CATSA framework. In Proceedings of the 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 16–18 December 2019; pp. 242–249.
11. Guo, S.; Xiao, B.; Yang, Y.; Yang, Y. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In Proceedings of the IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
12. Waseem, M.; Lakhan, A.; Jamali, I.A. Data security of mobile cloud computing on cloud server. *Open Access Libr. J.* **2016**, *3*, 1–11. [CrossRef]
13. Khoso, F.H.; Arain, A.A.; Lakhan, A.; Kehar, A.; Nizamani, S.Z. Proposing a Novel IoT Framework by Identifying Security and Privacy Issues in Fog Cloud Services Network. *Int. J.* **2021**, *9*, 592–596.
14. Lakhan, A.; Ahmad, M.; Bilal, M.; Jolfaei, A.; Mehmood, R.M. Mobility Aware Blockchain Enabled Offloading and Scheduling in Vehicular Fog Cloud Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 4212–4223. [CrossRef]
15. Guo, S.; Liu, J.; Yang, Y.; Xiao, B.; Li, Z. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. *IEEE Trans. Mob. Comput.* **2018**, *18*, 319–333. [CrossRef]
16. Tang, C.; Hao, M.; Wei, X.; Chen, W. Energy-aware task scheduling in mobile cloud computing. *Distrib. Parallel Databases* **2018**, *36*, 529–553. [CrossRef]

17. Wang, T.; Wei, X.; Tang, C.; Fan, J. Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. *Peer- Netw. Appl.* **2018**, *11*, 793–807. [CrossRef]

18. Tang, C.; Xiao, S.; Wei, X.; Hao, M.; Chen, W. Energy Efficient and Deadline Satisfied Task Scheduling in Mobile Cloud Computing. In Proceedings of the 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Shanghai, China, 15–17 January 2018; pp. 198–205.

19. Peng, H.; Wen, W.S.; Tseng, M.L.; Li, L.L. Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment. *Appl. Soft Comput.* **2019**, *80*, 534–545. [CrossRef]

20. Zhou, B.; Dastjerdi, A.V.; Calheiros, R.N.; Buyya, R. An online algorithm for task offloading in heterogeneous mobile clouds. *ACM Trans. Internet Technol. (TOIT)* **2018**, *18*, 23. [CrossRef]

21. Liu, L.; Fan, Q.; Buyya, R. A deadline-constrained multi-objective task scheduling algorithm in mobile cloud environments. *IEEE Access* **2018**, *6*, 52982–52996. [CrossRef]

22. Schäfer, D.; Edinger, J.; Eckrich, J.; Breitbach, M.; Becker, C. Hybrid task scheduling for mobile devices in edge and cloud environments. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Athens, Greece, 19–23 March 2018; pp. 669–674.

23. Chen, Z.; Hu, J.; Min, G.; Chen, X. Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization. *Concurr. Comput. Pract. Exp.* **2019**, *33*, e5413. [CrossRef]

24. Xu, R.; Wang, Y.; Cheng, Y.; Zhu, Y.; Xie, Y.; Sani, A.S.; Yuan, D. Improved Particle Swarm Optimization Based Workflow Scheduling in Cloud-Fog Environment. In Proceedings of the International Conference on Business Process Management, Sydney, Australia, 9–14 September 2018; Springer: Cham, Switzerland, 2018; pp. 337–347.

25. Zhang, J.; Qi, L.; Yuan, Y.; Xu, X.; Dou, W. A Workflow Scheduling Method for Cloudlet Management in Mobile Cloud. In Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 8–12 October 2018; pp. 932–937.

26. Lakhan, A.; Xiaoping, L. Energy Aware Dynamic Workflow Application Partitioning and Task Scheduling in Heterogeneous Mobile Cloud Network. In Proceedings of the 2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB), Fuzhou, China, 15–17 November 2018; pp. 1–8.

27. Zhang, G.; Cheng, S.; Shu, J.; Hu, Q.; Zheng, W. Accelerating breadth-first graph search on a single server by dynamic edge trimming. *J. Parallel Distrib. Comput.* **2018**, *120*, 383–394. [CrossRef]

28. Chai, R.; Song, X.; Chen, Q. Joint Task Offloading, CNN Layer Scheduling, and Resource Allocation in Cooperative Computing System. *IEEE Syst. J.* **2020**, *14*, 5350–5361. [CrossRef]

29. Nagarajan, S.M.; Deverajan, G.G.; Chatterjee, P.; Alnumay, W.; Ghosh, U. Effective task scheduling algorithm with deep learning for Internet of Health Things (IoHT) in sustainable smart cities. *Sustain. Cities Soc.* **2021**, *71*, 102945. [CrossRef]

30. Firouzi, F.; Farahani, B.; Marinšek, A. The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Inf. Syst.* **2021**, 101840. [CrossRef]

31. Ding, D.; Wang, Z.; Han, Q.L.; Wei, G. Neural-network-based output-feedback control under round-robin scheduling protocols. *IEEE Trans. Cybern.* **2018**, *49*, 2372–2384. [CrossRef] [PubMed]

32. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.