IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 6, 2019, accepted September 13, 2019, date of publication September 20, 2019, date of current version October 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942611

Towards the Task-Level Optimal Orchestration Mechanism in Multi-Device Multi-Task Architecture for Mission-Critical IoT Applications

SHABIR AHMAD, AZIMBEK KHUDOYBERDIEV, AND DOHYEUN KIM
Computer Engineering Department, Jeju National University, Jeju 63243, South Korea

Corresponding author: Dohyeun Kim (kimdh@jejunu.ac.kr)

This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this research was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (No. 2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent Technology for adaptive and disposable IoT).

ABSTRACT Internet of Things is advancing, and the augmented role of architecture in automating processes is at its vanguard. Mission-critical applications are becoming a vital category of future IoT applications, and due to the advancements in the 5G, the design of mission-critical application overcome a big hurdle. Mission-critical applications must be reliable, and the output should be known in advance. Therefore, to model such application, architecture is considered the cornerstone. One of the major requirements is the flexibility of the operation and the adaptability to new devices. In this paper, an optimal orchestration mechanism is proposed to automate the processes in a conventional multi-device and multi-task mission-critical architecture for flexible and scalable operations. The central goal of this paper is to threefold; first, to model tasks in such a way to maximize the flexibility in the operation plane. Secondly, to design a strongly correlated pair which has maximum relation and thus the chance to hit the task on the devices will be potentially maximized and also the idle time among operations is minimal. Lastly, to register devices in a network which is optimal for the group of this device in terms of correlation. We propose a multi-layer particle swarm optimization for each of the optimization objectives. Results show that the operation plan is flexible and with scaling up the problem size, the orchestration is still graceful and within the requirements of mission-critical applications. The performance of multi-level particle swarm optimization is compared with conventional single-level particle swarm optimization and it has been learned that the later is not only slower but also less accurate.

INDEX TERMS Internet of Things, task modeling, mission-critical IoT systems, task mapping, task deployment, service orchestration, task orchestration.

I. INTRODUCTION

Internet of things (IoT) is an established technology now, and over the past few years, the number of connected devices has been increased exponentially [1]–[3]. The recent advances in wireless sensor technology (WSN), real-time IoT, and 5G communication technology makes the job of mission-critical applications' designer more comfortable than it ever is. Mission-critical applications [4] are extensively used in smart healthcare, smart navigation systems, and robotics, to name a few. However, in the past, some of the requirements of mission-critical systems were making hurdles due to non-deterministic delays of networks and thus the surety which is needed in some special cases such as hard real-time missions could not be implemented as a result [5]–[7]. With recent breakthrough in 5G technologies, many research studies are surfaced which ensure safe profiles for network communication delays [8], [9].

Mission-critical applications are being revamped to fit in the paradigm of the IoT. A significant shift in the design of these applications is the power to control them remotely. In typical mission-critical application, the communication

The associate editor coordinating the review of this manuscript and approving it for publication was Zhong Fan.

TABLE 1. General Characteristics of Mission-critical IoT systems.

Attribute	Description
Dependability	These applications should be trusted and tested. The more the users trust the system, the more it will be dependent on it.
Safety	The system must be safe to use, and this must be ensured in the design phase.
Security	In mission-critical IoT systems, different interactions are being made with heterogeneous devices; thus, it must be ensured to have a proper firewall to protect from malicious attacks.
Real-time systems	If the correctness of a system not only depends on the accuracy of the results but also on the timely response.
Heterogeneity	One of the crucial attributes of mission-critical IoT systems to enable the connection of the different nature and design of physical things.
Scalability	Mission-critical application can achieve the size of hundreds of distributed things, and thus one of the design attributes has to be the easy provisioning of new devices, i.e., the scalability.
Context-awareness	The system must adapt to the changing context, and thus, the process must be made flexible to change.
Reusability	The system can be made in such a way that the components of the system could be reused.

was done on machine to machine (M2M) level, and due to the invariant network delays, the application layer protocol such as HTTP was not trusted which made the remote control impossible. In addition to the remote control, there are some other crucial requirements of mission-critical IoT applications which have been summarized in table 1. These characteristics demand a distinctive look at the design and architecture of the application. In recent studies, the design considerations of the mission-critical system are highlighted. For instance, in [4] the authors propose a model-driven approach as a go-to approach considering the attributes whereas in [10], the do-it-yourself (DIY) approach is proposed to be the recommended way for the design of these vital categories of IoT applications. Nevertheless, the architecture of the mission-critical applications still holds a pivotal role in their advancement in the right direction.

The concept of orchestration [11], [12] is often guided with the discussion of architecture. The processes and the data flow from one component of the system into the other systems is often known as orchestration if it involves some degree of autonomy. However, only autonomy is not enough when the requirements of the applications are changing and truly dynamic in nature, e.g., in the case of mission-critical applications. Having said that, autonomy must be accomplished in an optimized way. For instance, considering the scalability, real-time systems, and context-awareness characteristics of mission-critical applications, one of the optimized ways of orchestration is to maximize the flexibility, optimal task assignments, and scalability. A mere orchestration will automate this aspect without caring much on the other possible solutions which may be the better from the ones currently being orchestrated. Therefore, considering mission-critical applications, the optimized orchestration is of paramount importance.

In this paper, we propose a novel architecture for mission-critical IoT applications which approaches the orchestration mechanism in an optimized way to for flexible and scalable operation plan. Owing to the fact that characteristics of the mission-critical application must be considered in the design of the architecture, the paper considers the context-awareness and scalability characteristics to maximize the scalability and flexibility of the operation plans. The flexibility is the measure of the distance of the critical path time of the operations. For instance, if there is 50 hours time leading to the critical path, it would mean that another operation can be added if the time falls with 50 hours. Thus maximizing this time can maximize the flexibility in the operation plan. Another part of the optimized orchestration is the allocation of tasks in such a way to lead to the lowest idle time and fastest response. If the devices in the operation plan are highly correlated, it will be more likely that the request made will target the related devices in the group. One final orchestration is the optimal design of devices within a network. It will lead to the lowest response time as this will maximize the intra-network request to minimize the inter-network requests. For this, multi-purpose optimization, we introduce a multi-level particle swarm optimization. Each level addresses one objective and provides the optimal results to the next level.

The contribution of this paper is as follows:

- Implements an optimized orchestration mechanism to allow more flexible and scalable operation plan.
- Introduces surplus time optimization, which is the quantitative measure of flexibility and scalability.
- Introduces device correlation utility to measure the relevancy of devices within a group of operation plan and thus decreases the idle time.
- Implements network commutative correlation index, which measures the correlation of devices inside a network and thus boosts the response time.
- Finally, implements a multi-level particle swarm optimization to optimize the various stages of orchestration.

The remainder of the paper is structured as follows. Section 2 presents the literature review and highlights the relevant contributions in mission-critical IoT architectures and optimization. Section 3 illustrates the system model for the architecture which designs optimized orchestration mechanism. Section 4 overviews the mathematical formulation of the system and derives the objective functions for each level of the optimization. Section 5 portrays the design of the architecture and multi-level particle swarm optimization. Section 6 discusses the implementation environment and hardware being used in this paper. Section 7 discusses the results and evaluates the performance of the system with a single-level particle swarm optimization. Section 8 concludes the paper and highlights the future directions.

II. RELATED WORK

With the current revolution in the IoT, the number of connected devices is in the order of billions. With this evolution, the shift is moved from conventional consumer-based applications towards mission-critical applications [10]. In consumer-based applications such as smart home, reliability is not of significant importance whereas in mission-critical applications such as remote surgery, reliability and deterministic behaviour of the application must be known before the deployment in real environment [13]. Many efforts have been made in conventional mission-critical applications and machine-to-machine communication, and as a consequence, very sophisticated solutions are around for quite some time [14], [15]. The problem is that in mission-critical IoT application, the communication is not only machine-to-machine but also user-to-machine and machine-to-user. In other words, the communication has to be performed on the application layer as well as on the device-to-device layer as in case of machine-to-machine communication. The mission-critical IoT application is the upgraded form of conventional mission-critical application with allowing communication at application layer [4], [16].

Nevertheless, the requirements of these application demand determinism and guarantee before deploying in a real environment. In previous IoT-based mission-critical research studies, the focus was towards effective communication to achieve ultra-low latency and also the deterministic profile of communication medium. These studies are still fundamental in a sense that without a reliable communication medium, allowing mission-critical IoT communication is considered a big hurdle. With the recent advancements in 5G technology and network slicing, the communication delay is ultra-low and a safe maximum bound can be ensured [6], [16]. Communication is the single fundamental requirement of mission-critical applications, but there are other requirements that also equally contribute to these applications. As mentioned in the Introduction section, some of these requirements are safety, scalability, adaptability and real-time compliance. Therefore, there is a clear gap in the literature to address some of these requirements in addition to the communication networks optimization.

The concept of orchestration is used in designing optimal service provisioning in IoT application in general. This concept often comes around after the popularity of service-oriented architecture and has been used in many studies [17]–[19]. In earlier approaches, the orchestration was considered as a single centralized executable process running on cloud. However, with the advancement in the fog technologies, the orchestration became more of a distributed nature. Nonetheless, it remains the fundamental process in traditional as well as modern IoT application [20]. The work done so far on the orchestration are focused on automatic provisioning of services, and there is a recent shift towards task-level orchestration [10]. Orchestration at the task level will make the application more flexible, and the service which is composed of tasks will be more dynamic and smarter [10]. Moreover, it will be more lightweight, which is the need of nowadays fog computing architecture. Considering the requirements of mission-critical application orchestration at task level suits best to and therefore used in this paper.

In mission-critical IoT applications, task orchestration refers to the dynamic generation of tasks, mapping them on the virtual objects and scheduling the time of the allocation optimally. Task management in IoT and Wireless Sensor Networks have attracted some literature studies; however, all of the studies focus on one thing or another and consequently, there is no major work which considers task orchestration. For task orchestration modelling of tasks is very vital. Similarly, in modelling the load of the system, the resources associated with the application can also be taken into consideration, and consequently, the modelling should be smarter given the limited characteristics of IoT resources [5]. The studies in [21]–[23] deals with the allocation of tasks in such a way that the resources are utilized in an optimized manner and also the load are balanced on all edge nodes. *IoT_ProSe* [24] framework extends this further to consider the challenge of task allocation in mobile nodes. Task mapping, on the other hand, is somewhat unfamiliar in the IoT domain but a variety of efforts have been made in WSN and distributed systems. The main challenge in the mapping is to reduce the time of mapping and inter-process communication and static traversal of resources for tasks which consume time and CPU clocks as well [25], [26]. Mapping can do this well before time, and the constrained power will not be consumed in a work that can potentially be performed before deployment.

Another challenge in orchestration is the optimal selection of processes. The mere automation of processes sometimes leads to poor results; therefore, the selection of the optimal route is necessary. As the role of orchestration is many-fold, therefore it is a typical case of a multi-objective problem. When there are more than a single objective for an optimization problem, and some of the work can be parallelized, the recommended approach is to use multi-level optimization. It achieves better fitness than single-level optimization and at the same time is a more efficient approach. There are a variety of instances in the literature in which multi-level optimization is preferred over a single-level optimization for a specific problem space [27]–[29]. For instance, in [27], the authors propose a hybrid of genetic algorithm and PSO for a supply chain model using a bi-level optimizer. Similarly, Garg and Sharma [30] back the same idea and propose the solution of a multi-objective problem using fuzzy nonlinear programming to achieve a more adaptable and flexible solution. In contrast to a hybrid approach, there are some instances as well where a multi-objective problem is solved with a single algorithm by deploying it either recursively or in a pipeline architecture in more than single-level [31], [32]. Yeh [31] proposed a two-stage discrete particle swarm optimization for optimal redundancy allocation in the series system. Similar efforts have been recently made to propose

optimization in multi-level and found to have to achieve better results than mono-level optimization [33]–[35].

III. SYSTEM MODEL

In this section of the paper, a detailed system modelling of the proposed work has been illustrated. As described in earlier sections, one of the prime goals of the mission-critical applications is the design and architecture of these applications should be in compliance with their requirements. For instance, some of the requirements are low latency for real-time communication and scalability to support networks at large-scale. Therefore, architecture must consider these requirements. In this paper, the focus is to design an architecture which is robust, efficient and flexible. It has been outlined in recent studies that task-level design and orchestration is a commendable approach, unlike traditional service-level approach, which is a not so-flexible approach. Consequently, this paper follows these recent suggestions and design the architecture, whose main design block is a task. In task-level architecture, tasks are generated based on the mission requirements, and afterwards, the tasks are mapped on the devices and scheduled to find the order which meets the real-time requirements. Finally, the tasks are allocated in the designated footprint decided by the scheduler.

In this, the task-level orchestration is used in an optimal way to maximize the flexibility in the scheduling, tasks mapping and device registry for the optimal network. Figure 1 shows the overall block model of the proposed approach. The proposed approach follows a top-down methodology. The mission requirements are supplied by end-users. The application parses the requirements and generates tasks using natural language processing techniques. These tasks are supplied to the next layer in which the tasks are modified for optimal flexibility, the detail of which will be illustrated in the subsequent section. Once the tasks are optimized, the next layer deals with deploying these tasks in such a manner which leads to the best scheduling plan. For this, the device pairing is performed in an optimal way to maximize the correlation. In the next layer, the devices are registered in such a network which leads to maximum network utilization and maximum efficiency. Once these 3-layer optimization performed, the tasks are then allocated and monitored. In the next section, the mathematical formulation is discussed, and the detailed methodology of multi-layer optimization is illustrated.

IV. MATHEMATICAL FORMULATION OF MULTI-LAYER OBJECTIVE FUNCTIONS

In this work, we use a multi-layer optimization scheme to optimize orchestration processes. As described, the architecture revolves round of tasks. Consider we have a set of tasks which are performing some mission-critical operations on edge nodes. The list of symbols used in this paper is described in table 2.

Consider a set of tasks τ having $\tau_1, \tau_2, \tau_3..\tau_n$ responsible for mission operations on physical devices $D = D_1, D_2, D_3, ...D_k$. A micro-tasks $\mu\tau$ is a sub-task which is part of the operation performed by a task. For instance, if report-temperature is taken as one instance of a task, then get-temperature and send-temperature are two distinct micro-tasks. Consequently, a task τ_i has micro-tasks $\mu\tau_1, \mu\tau_2, \mu\tau_3, ...\mu\tau_n$. A task can be allocated on one virtual object VO_i or on a set of virtual objects $VO_1, VO_2, VO_3, ...VO_k$. However, a micro-task must be allocated on only a single virtual object. A micro-task is called device operation when allocated on a physical device. In other words, device operation is micro-task in execution, i.e., $\mu\tau_{ij} = D_iO_r$. Finally, the operation of a task must be executed in ascending order. For instance, $\mu\tau_{11}$ must be executed before $\mu\tau_{12}$. We define the following constraints before digging in detail of optimization on each layer.

TABLE 2. Summary of symbols and notation used in different algorithms.

Symbol	Description
τ_i	ith task
$\mu\tau_{ij}$	Micro task j which belongs to task τ_i
VO_i	ith Virtual object which is the virtual representation of physical devices
D_i	ith Device
D_iO_r	The rth operation performed by ith device
$\mu\tau_{ij}d$	Duration of $\mu\tau_{ij}$
$\mu\tau_{ij}st$	Start time of $\mu\tau_{ij}$
$\mu\tau_{ij}spt$	Surplus time (The time till which the micro-task can be delayed) of $\mu\tau_{ij}$
T_{idle}	Idle time of scheduling footprint
$\mu\tau_{ij}wcft$	Worst-case finish time of $\mu\tau_{ij}$
$\mu\tau_{ij}ft$	Finish time of $\mu\tau_{ij}$
ONS	Optimal Network Selection objective
$\phi(D_1, D_2..D_k)$	Device Correlation among D_1 to D_n
CCI	Commutative Correlation Index of the network of devices
O_p	Operation plan of set of devices

$$\mu\tau_{ij} => (hasAttributes)\{d_{ij}, wcft_{ij}, st_{ij}, ft_{ij}, spt_{ij}\}$$
$$\tau_i => (requires)\{\mu\tau_{i,1}, \mu\tau_{i,2}, \mu\tau_{i,3}, ...\mu\tau_{i,k}\}$$
$$D_m => (performs)\{D_mO_0, D_mO_1, D_mO_2, ...D_mO_r\}$$
$$O_p => (definedBy)\{D_1, D_2, T_{idle}, Corr\}$$

Similarly, a micro-task is equivalent to device operation or simply operation when it is in execution state.

$$\mu\tau_{00} = D_mO_0 = O_o$$

Based on the above-mentioned assumptions, the first job of the architecture is the optimal design of micro-tasks. For this, the proposed architecture computes a function which incentivizes the rearrangement of micro-tasks in a way which would lead to maximum surplus time. The part of this optimization is to assign start time and finish time in such a way to increase the surplus time spt. The surplus time function is given by:

$$SPTu = \sum_{i=0}^{n} spt_i \tag{1}$$

where $SPTu$ is the utility of surplus time in a given tasks arrangement. The surplus time, as described earlier, is the

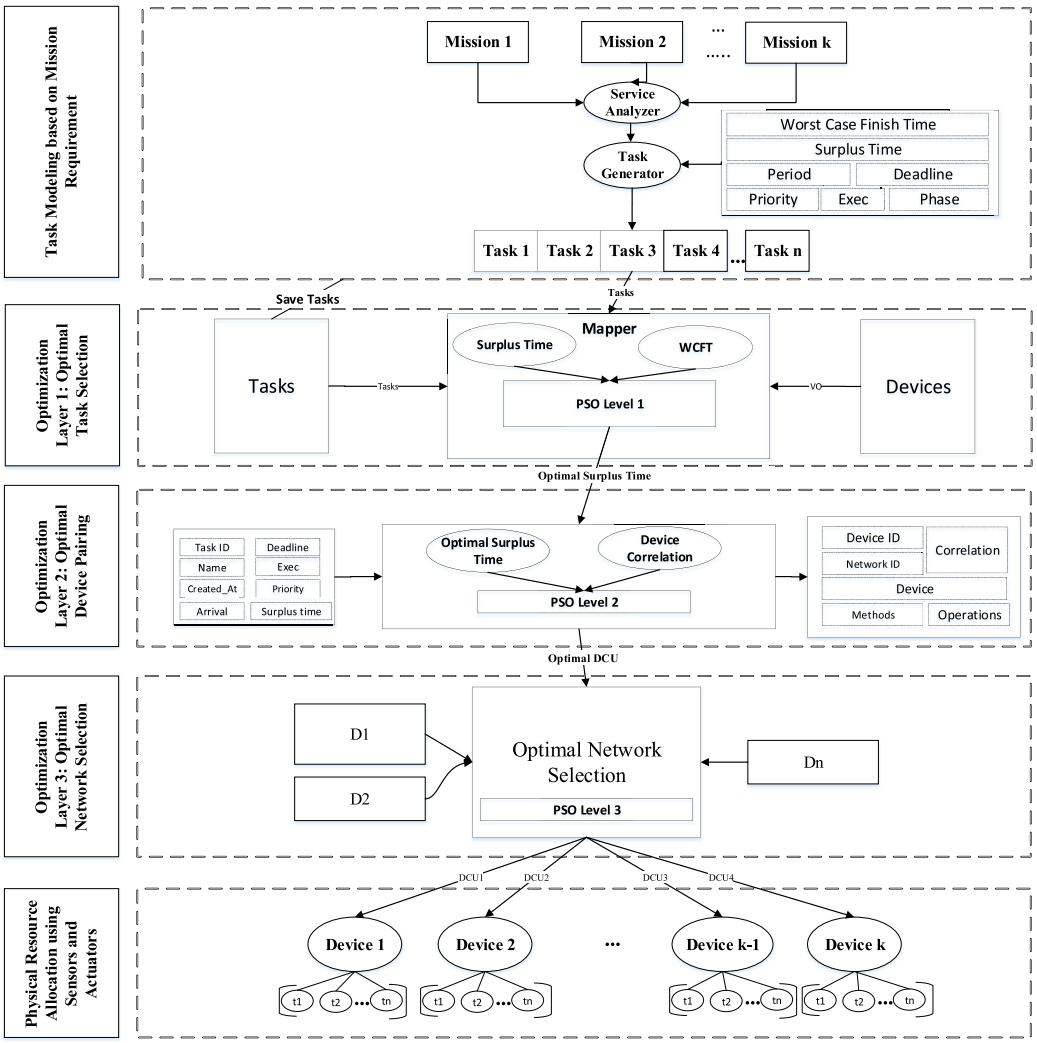


FIGURE 1. System Block Representation.

time till which a particular operation can be delayed. The *spt* of a micro-task $\mu\tau_{ij}$ is given by

$$spt_{ij} = wcft_{ij} - d_{ij} \tag{2}$$

where spt_{ij}, $wcft_{ij}$ and d_{ij} are the surplus time, worst-case finish time and duration of $\mu\tau_{ij}$ respectively. The final time is the instant at which the execution of a particular micro-tasks ends and is given by

$$ft_{ij} = st_{ij} + d_{ij} \tag{3}$$

where ft_{ij}, st_{ij} and d_{ij} are the final time, start time and duration of $\mu\tau_{ij}$ respectively. To assign a start time, the optimizer will make sure the tasks must be executed in the order they are created. For instance, if a task "Record Temperature" has two micro-tasks "getTemp" and "reportTemp", the start time of the reportTemp must be greater than or equal to the final time of getTemp, i.e., $st_{ij} \geq ft_{ij-1}$.

To illustrate this optimization mechanism, we take the example of temperature, humidity and particulate matter sensors as indoor quality monitoring devices. Table 3 overviews the list of devices connected with different IoT agents and the tasks they provide.

TABLE 3. Overview of the list of micro-tasks and their initial attributes.

Device ID	Provides	Agent	Network ID	Symbol
BME280	Temp, Humid, Pressure	Raspberry	192.168.12.1	D1
SM21021	PM10, PM2.5	Arduino	192.168.12.2	D2
BMP280	Temp, Humid, Pressure	Raspberry	192.168.12.2	D3

Based on the devices, and IoT service description the tasks and micro-tasks generated are shown in table 4 alongside their initial attributes. The duration *d* and worst-case finish time *wcft* are the constraint attributes which must be considered whereas other attributes can be assigned in such a way to lead to maximum surplus time *spt*.

Sample problem: For instance, a task τ_1 is defined as $\mu\tau_{10}$ and $\mu\tau_{11}$ and task τ_2 is defined as $\mu\tau_{20}$ and $\mu\tau_{21}$.

TABLE 4. Overview of the list of micro-tasks and their initial attributes.

$\mu\tau$ ID	d	wcft	st	ft	spt	τ ID	$\mu\tau$ Name
1.0	1	9	0	2	7	1	Get Temperature
1.1	1	10	2	4	6	1	Report Temperature
2.0	2	12	3	5	1	2	Get Humidity
2.1	4	11	2	6	4	2	Report Humidity
3.0	6	14	1	7	7	3	Get PM2.5
3.1	3	10	3	6	4	3	Report PM2.5

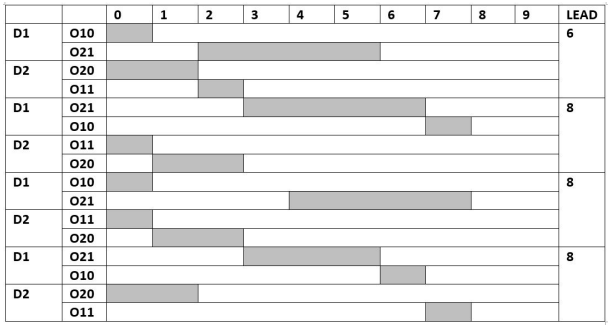


FIGURE 2. Gant-chart of the different arrangement of micro-tasks in a given operation plan.

The values taken for *wcft* and *d* is taken from table 4. Similarly, an operation plan O_p1 for instance is equivalent of device 1 D_1 and operation plan O_p1 is defined as the equivalent of device 2 D_2. Each respective operation plans has defined as follows:

$$D_1 \in \{D_1O_0, D_2O_1\}$$

or simply

$$D_1 \in \{O_{10}, O_{21}\}$$

Similarly, operation plan 2 has the following plan.

$$D_1 \in \{D_1O_1, D_2O_0\}$$

or simply

$$D_1 \in \{O_{11}, O_{20}\}$$

For the above-mentioned sample problem, the different operation plans with the different micro-tasks arrangement are listed in table 5. For operation plan of D_1 and D_2, O_{11} and O_{20} lead to highest surplus time of 18 for D_2 whereas O_{10} and O_{21} lead to highest surplus time of 13 for D_1. However, these selected pairs are not correlated with each other, and therefore, it has to be arranged in such a way that it leads to maximum correlation and minimum lead time and this is the job of the second layer optimization of the architecture.

In the second layer, the objective is to find the operation plans of the devices, which lead to maximum correlation. The correlation between devices is a function of lead time. If the lead time is minimum, the correlation will be maximum, and the scheduling footprints will be more scalable. Figure 2 illustrates the Gant chart of the different combination and the resultant lead time of every pair.

TABLE 5. Re-arrangement of micro-tasks (operations) in a given operation plan and its effect on surplus time.

Operation Plan	$\sum_{i=0}^{n} spt_i$	$\mu\tau name$	st	ft	spt
$D_1(O_{10}, O_{21})$	13	getTemp	0	1	8
		repHumid	2	6	5
$D_2(O_{20}, O_{11})$	17	getHumid	0	2	10
		repTemp	2	3	7
$D_1(O_{21}, O_{10})$	6	repHumid	3	7	5
		getTemp	7	8	1
$D_2(O_{11}, O_{20})$	18	repTemp	0	1	9
		getHumid	1	3	9
$D_1(O_{10}, O_{21})$	11	getTemp	0	1	8
		repHumid	4	8	3
$D_2(O_{11}, O_{20})$	16	repTemp	1	2	8
		getHumid	2	4	8
$D_1(O_{21}, O_{10})$	7	repHumid	2	6	5
		getTemp	6	7	2
$D_2(O_{20}, O_{11})$	12	getHumid	0	2	10
		repTemp	7	8	2

Although, the optimal arrangement for D_1 is (O_{10}, O_{21}) and for D_2 is (O_{11}, O_{20}) but the lead time of this pair is higher than the pair $D_1(O_{10}, O_{21})$ and $D_2(O_{20}, O_{11})$ and thus the later pair is more correlated than the one highest individual surplus time. Therefore, not only the surplus time is important, but we must also have an orchestration system in which devices with strong correlation must be selected to implement a task. The correlation function between devices is computed based on a utility function DCU which is given by:

$$C = maximizeDCU(D_1, D_2, ...D_n) \tag{4}$$

The DCU is computed as follows.

$$DCU(D_x, D_y) = \frac{1}{1 + T_{Idle}} \tag{5}$$

where T_{idle} is the idle time in a given scheduling plan and is given by

$$T_{Idle} = \sum_{i=0}^{n} st(\mu\tau_y) - ft(\mu\tau_y) \tag{6}$$

For the sample problem in the above-mentioned paragraphs, the device correlation is computed in table 6. It shows the idle time T_{Idle} and DCU for every operation plan in a given device pair. The first device pair has a very strong correlation of 0.5 as compared to other pairs. This is due to the lowest idle time. The same pair also generates the lowest lead time, which can be proved from the Gantt chart in figure 2. Consequently, the optimal pairing of the device can enable a very low idle and lead time and very high surplus time which would mean the scheduling and task allocation is not only flexible and scalable but also very efficient. If surplus time is low, then any change in the scheduling plan may make

TABLE 6. Re-arrangement of micro-tasks (operations) in a given operation plan and its effect on DCU.

Operation Plan	$\sum_{i=0}^{n} spt_i$	$\mu\tau name$	st	ft	spt	T_{Idle}	DCU
$D_1(O_{10}, O_{21})$	13	getTemp	0	1	8	(0-0) + (2-1) + (0-0) + (2-2) = 1	$\frac{1}{1+1} = 0.5$
		repHumid	2	6	5		
$D_2(O_{20}, O_{11})$	17	getHumid	0	2	10		
		repTemp	2	3	7		
$D_1(O_{21}, O_{10})$	6	repHumid	3	7	5	(3-0) + (7-7) + (0-0) + (1-1) = 3	$\frac{1}{1+3} = 0.25$
		getTemp	7	8	1		
$D_2(O_{11}, O_{20})$	18	repTemp	0	1	9		
		getHumid	1	3	9		
$D_1(O_{10}, O_{21})$	11	getTemp	0	1	8	(1-0) + (4-1) + (1-1) + (2-2) = 4	$\frac{1}{1+4} = 0.20$
		repHumid	4	8	3		
$D_2(O_{11}, O_{20})$	16	repTemp	1	2	8		
		getHumid	2	4	8		
$D_1(O_{21}, O_{10})$	7	repHumid	2	6	5	(2-0) + (6-6) + (0-0) + (7-2) = 7	$\frac{1}{1+7} = 0.12$
		getTemp	6	7	2		
$D_2(O_{20}, O_{11})$	12	getHumid	0	2	10		
		repTemp	7	8	2		

the surplus time into negative, which would mean the task is dropped. Consequently, the scheduling footprint is said to be non-scalable and not in line with the requirements of mission-critical IoT applications.

It has been highlighted in many literature studies that reliability of mission-critical applications not only depends upon the sound architecture and design of scheduling but also on dependent on network delays. Owing to this fact, a group of devices if correlated will likely be hit in the same request and if they reside in the same network, the delay will be lesser than if they reside on a different network. Thus, to decrease the network latency, the device registered on the network must be highly correlated. The third optimization layer is the optimal selection of devices for the network registry.

The objective of the optimized network is to find a set of devices which contributes to highest cumulative correlation index (CCI). The objective function is to maximize CCI. The CCI of a network is given by:

$$CCI = \sum_{i=0}^{n} \sum_{j=0}^{k} (DCU_i + DCU_j) \tag{7}$$

Based on the above objective function, the job of the optimizer in the third layer is to find an arrangement of devices within a network to maximize the CCI. For instance, for the devices D_1, D_2 and D_3 in table 3. The optimal DCUs are computed based on the second layer optimizer and based on them; the network CCI is computed, as shown in table 7. The CCI for the network in the second row, i.e., (D_1, D_3, D_2) is maximum and is considered the optimal network for a given operation plan. Having said that, this network will always lead to the most flexible scheduling plan and the most scalable orchestration. Moreover, due to the highest CCI, the lead time and idle time will be the lowest making it the most efficient plan out of all the arrangements.

TABLE 7. Re-arrangement of devices in different networks and its effect on CCI.

Network Plan	DCU_i	DCU_j	CCI
(D_1, D_2, D_3)	$DCU(D_1, D_2) = 0.2$	$DCU(D_2, D_3) = 0.3$	0.7
(D_1, D_3, D_2)	$DCU(D_1, D_3) = 0.5$	$DCU(D_3, D_2) = 0.4$	0.9
(D_2, D_1, D_3)	$DCU(D_2, D_1) = 0.1$	$DCU(D_1, D_3) = 0.5$	0.6
(D_2, D_3, D_1)	$DCU(D_2, D_3) = 0.3$	$DCU(D_3, D_1) = 0.5$	0.8
(D_3, D_1, D_2)	$DCU(D_3, D_1) = 0.5$	$DCU(D_1, D_2) = 0.3$	0.8
(D_3, D_2, D_1)	$DCU(D_3, D_2) = 0.3$	$DCU(D_2, D_1) = 0.2$	0.5

V. DESIGN OF OPTIMAL ORCHESTRATION ARCHITECTURE

In this section of the paper, a detailed design of the proposed optimal orchestration system is described. The prime job of this architecture is to allow efficient and scalable mission-critical communication. The flow of the architecture is described in figure 3. First off, all modules of the architecture are bootstrapped, and the libraries required for mission analysis are initialized. The mission analyzer module processes the requirements and tokenizes the information into mission objects. The mission objects are parsed, and the process responsible for generating tasks are instantiated. The list of tasks based on mission analyzer tokens is created and

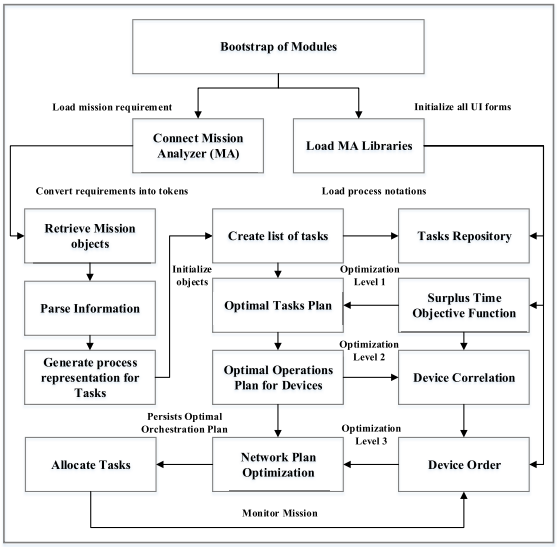


FIGURE 3. Functional Flow of Proposed Architecture.

persisted in tasks repository. The same tasks are supplied to optimal tasks plan which incentivizes the arrangements of the operations within a task in such a way to maximize the surplus time. The objective function to compute the surplus time is provided to the level 1 optimizer. The optimal tasks plan are provided to the next level of the optimizer, where the operations of the tasks deployed on the devices are optimized to form an optimal operation plan. The device correlation is computed, which forms the objective function for this level. Once the optimal operation plans for devices are found with level 2 of the optimizer, the same information is provided to form a network of devices in which the devices have more relevancy with each other. Based on this optimal orchestration, the tasks are allocated with the computed operation plans and the mission is monitored.

The flow of the sequence of operations among various processes of the orchestration is illustrated in figure 4. IoT devices are registered with edge nodes typically a Raspberry PI with an IoT server acting as a gateway for external world application and physical things. A typical IoT device has a name, the port to which it is connected, an identifier which is unique for it, a location [latitude and longitude], type of device whether it is an actuator or a sensor or a hybrid device having both capabilities and finally, the data it senses. Once the device is registered to the edge node, the same information is passed to the device virtualizer where it is represented in the cyber domain in the form of a virtual object. A virtual object is the digital counterpart of a physical IoT device as defined in many recent research studies. The task generator modules tasks and micro-tasks based on mission requirements and the set of these generated entities are passed to the device-mapper and task allocator. The device-mapper also receives virtual objects from the device virtualizer, and couple it with devices to make an initial operation plan. At this point, the operation plan and the arrangement of micro-tasks are not optimized, and hence, the lifeline of the first level of the optimizer is instantiated. The role of the optimizer in this level is to find an arrangement of micro-tasks [operations] which maximizes the surplus time. Once the optimal surplus time is achieved, it is passed alongside the device operation plan to find the optimal correlation among devices in an operation plan. The role of this optimizer is also to maximize the correlation of devices based on the objective function defined in earlier sections. Finally, the number of devices, the operation plan and the optimal SPT-based task arrangement are passed to the third level of the optimizer to find the optimal network by maximizing the CCI. The mechanism to compute CCI is already illustrated in earlier sections. Once the optimization is performed, the optimal orchestration plan is shared with the edge node. The edge node, in turn, forwards it to task allocator, which allocate the operation inside the tasks to devices. The devices are being operated and finally monitored for the correct operation.

A. DESIGN OF MULTI-LEVEL OPTIMIZER

In order to optimize the problem, we have divided the solution into different sub-problems. Particle swarm optimization (PSO) is used as a base algorithm. PSO is based on the meta-heuristic class of optimization algorithms. It is first introduced by Kennedy and Eberhart. PSO works by generating particles equal to the size of problem variables. Each particle p has a position $x(p) \in \mathbb{R}^n$ and a velocity $v(p) \in \mathbb{R}^n$ in the solution space. Apart from position and velocity, *Pbest* and *Gbest* are also used. *Pbest* is used for personal best, which is the best solution locally for a particular particle, whereas *Gbest* is the global best solution for every particle. Variables such as c_1, c_2 and w are used as the local best solution learning rate, the global best solution learning rate and the particle inertia, respectively. Finally $r_1, r_2 \in \mathbb{W} \forall\, 0 \leq r_1, r_2 \leq 1$.

In the start, particles are positioned on a purely random basis in the solution search space. Initial speeds are also assigned to each particle. Every particle is associated with three metrics; the current position which represents the current solution in search space, the velocity which determines which way the particle will move in the search space and lastly, the performance, which is the cost value of the objective function and in terms of fitness. The movement of the particle is based on the following two equations; equation (8) computes the velocity and equation (9) move the particle to the next position in search space.

$$v(p) = wv(p) + c_1 r_1 (Pbest(p) - x(p)) + c_2 r_2 (Gbest(p) - x(p)) \tag{8}$$

$$x(p) = v(p) + x(p) \tag{9}$$

For multi-level optimization, the global problem of optimal orchestration is divided into three sub-problems; SPT optimization, DCU optimization and CCI optimization. Multi-level PSO (MLPSO) is designed as a new solution algorithm which uses PSO in the backend. The use of multi-level optimization approach follows three main rules; first, the optimization path is split into multiple sub-path in their respective

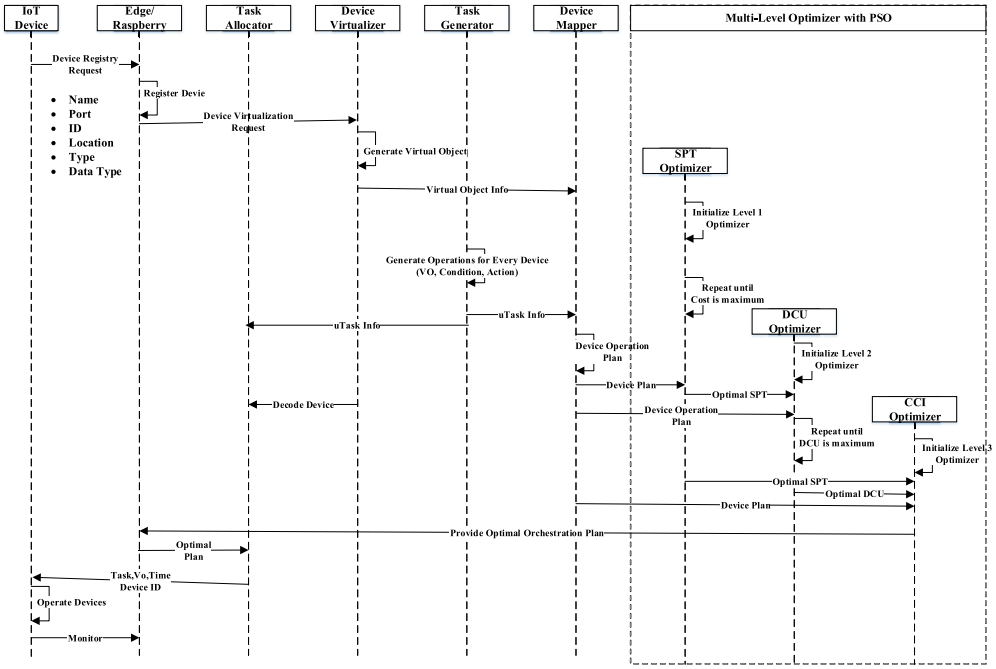


FIGURE 4. Sequence of Operations within the optimal orchestration architecture.

sub-space, the results collected in each sub-path is provided as an input parameter to the next level, and finally, the evaluation is performed at each respective level is also summed up to find the overall evaluation. This is has been illustrated in figure 5. It has been shown that on level 1 plane, the particles are moved to the global best position of that plane. The optimal solution is provided to the next level (immediate top plane) where the same optimization performed for the input from level 1 output and a new objective function. The same process is repeated to the next level at the topmost level. In this paper, the first level deals with surplus time optimization, the second level is device correlation utility optimization, and the third level is cumulative correlation index optimization for optimal network selection.

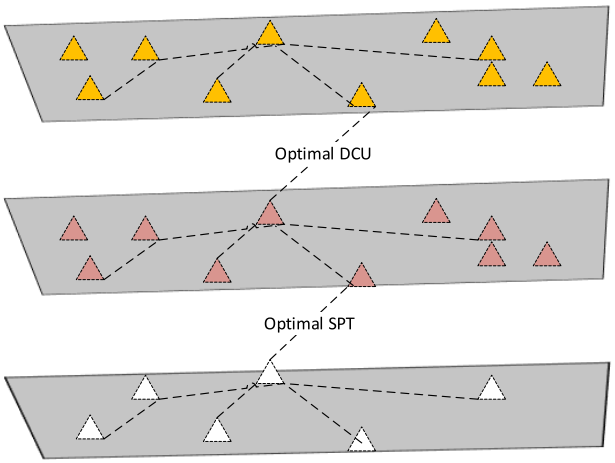


FIGURE 5. Illustration of Multi-level PSO and Particle movement.

VI. IMPLEMENTATION ENVIRONMENT

In this section of the paper, the implementation environment of the system is described in details. The technology stack for this work is selected on a careful look of the recent trend in research and development. Python is a programming language which offers a variety of benefits not even for application developers but also for researchers who are looking for optimal architecture. The IoT server deployed on Raspberry PI is programmed in a Python-based web framework named Flask. Flask is very lightweight and is the preferred choice in many research studies due to its modularity and on-demand instantiation of libraries. For MLPSO, PySwarm library is enabled, which provides a variety of swarm intelligence optimization algorithms in which PSO is one of them. For result analysis Bootstrap 3 and HTML5/CSS3 are used to visualize them in a client application such as Chrome. Additionally, we use MySQL for the persistence of data, TexTBlob library for natural language processing and tokenization and jsPlumb for drag-and-drop features, which is needed to intuitively visualize the correlation among different objects in the mission plane. The summary of the technology stack used is shown in table 8.

On the embedded hardware end, we mainly use three sensors already discussed in table 3 and actuators such as LED and Fan for notification and monitoring. We used two distinct hardware Raspberry and Arduino. The Nova Dust sensor is an analogue sensor, so it is convenient to connect it with Arduino to avoid the need for any extra hardware for analogue

TABLE 8. Technology Stack of Proposed Architecture.

Technology	Description
System Specifications	PC, Raspberry PI and Arduino
Operating System	Windows 10 for PC Server, Raspbian for Edge Raspberry PI
Programming Language	Python Flask, JavaScript, HTML, CSS
Optimization Algorithm	PSO
Optimization Package	Python PySwarm
Mission Analysis Package	TextBlob
Libraries	Bootstrap 3, Jinja 3, jsPlumb for mapping
Server	Flask Server
Database	MySQL
Client	Chrome and Firefox
Core Programming Language	Python 3
IoT Devices	Temperature Sensor, Humidity Sensor, Wind Sensor, Fan Actuator and LED actuator

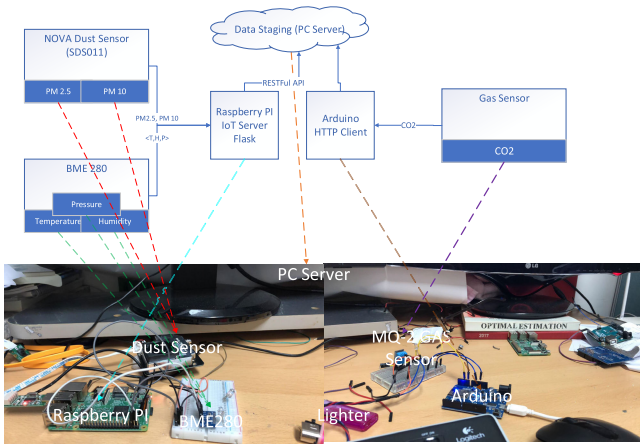


FIGURE 6. Embedded Hardware for Implementation of Optimal Orchestration.

to digital conversion. The other two sensors are connected with Raspberry PI as they can provide data in digital format, and thus, Raspberry is a viable option. The implementation hardware is pretty simple but is enough to demonstrate the optimal orchestration on real mission-critical applications. Figure 6 shows the setup. The conceptual setup is mapped on the corresponding devices with dotted arrows. On the left side is the Raspberry PI-based edge node while o the right-hand side is the Arduino edge node. It has devices such as dust sensor, MQ-2 CO_2 sensor and BME280 sensor.

VII. RESULTS

In this section of the paper, the results of the different processes in mission-critical optimized orchestration is discussed. We use MLPSO for the optimization of surplus time, device correlation utility and network cumulative correlation index. The parameter of MLPSO is summarized in table 9. Each level has a small number of particles which gets multiplied with the next level to produce almost a million recursive solutions. Therefore, it is vital to have a fair compromise between the consumption time to find the best solution and

TABLE 9. Multi-level PSO paramters as a fair compromise between time and accuracy.

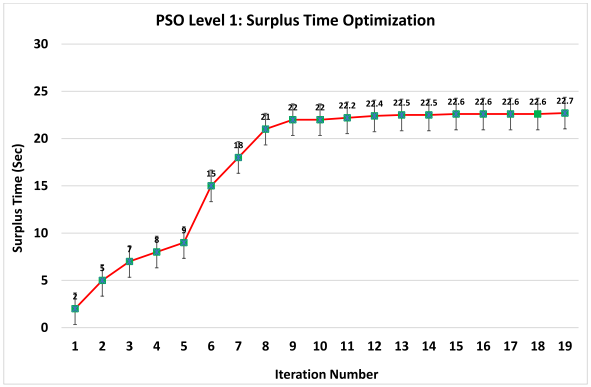
Level	No. of particles	No. of Generations
1	24	50
2	72	80
3	91	100

the accuracy of the solution, which is shown in the aforementioned table.

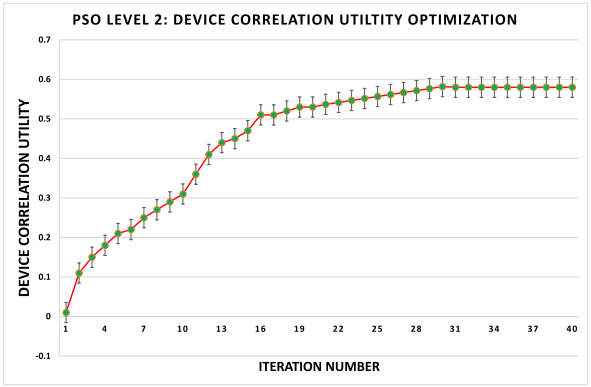
The learning factor c_1 and c_2 is taken as 1.23 and 1.31 respectively whereas the inertia coefficient w is taken as 0.81. The results of the different level function are shown in figure 7. The cost of each level optimization is shown in respective subfigures. The first subfigure portrays surplus time optimization. The cost is maximized with each iteration as the objective is to maximize the cost. The number of particles is taken as 50, and the number of iterations is taken 20 for level 1. The surplus time is 2 in the start, and the optimizer repeatedly checks for best solution by moving the particles in the optimal direction. The rate of finding the best solution is very fast in the initial iterations, but as the solution is getting closer to the optimal solution, the rate of change is decreased drastically. This is why the slope of the graph is approaching 0 after iterations 13. The optimal solution shows a total of 22.7 seconds for the input tasks. In level 2, the optimal solution of the level 1 is provided as discussed earlier in the design section. The number of iterations is also added up to 40 in contrast to 20 in level 1. The optimizer takes the tasks and their surplus time and find the optimal device correlation. The correlation is based on the idle time in the operation plan. DCU is the inverse of idle time, and the graph shows that the idle time is decreasing with each iteration, and thus the overall DCU is maximized, which is the objective of PSO on level 2. Here also the rate of finding the best solution is high, but when the particles are reaching closer to the global best solution, the rate of change is approaching 0. In level 2, the optimal DCU is recorded as 0.6. The optimal task arrangement, coupled with highest correlated devices, are provided to the next level. Subfigure 7(c) shows the third and final level of PSO. In this layer, the network is optimized based on CCI value. The higher the CCI value, the more optimal, will be the solution.

In this level, the iterations are 80 for the same 50 particles. The algorithms traverse different order of devices in the networks and compute the CCI value and place it in gbest. In the next iteration, it checks whether the current value is better than gbest if it is, it replaces it with gbest, and this process goes on till it finds the optimal best solution. The graph reflects a steady increase in the CCI on each iteration and the value is maximum on iteration number 76, after which it remains the same. Table 1 shows the result of every 8^{th} iteration of level 3 PSO to form an optimal network.

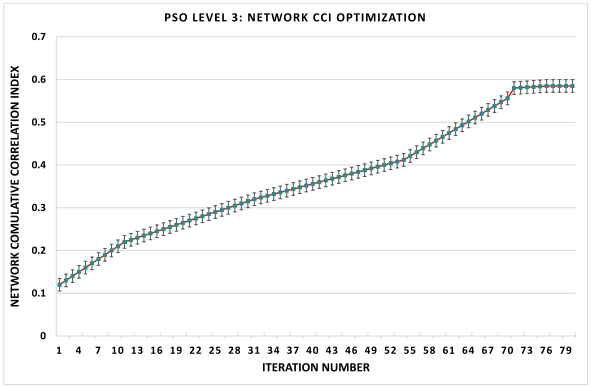
For evaluation, we compared the proposed multi-level PSO with a single-level PSO. In single-level PSO, we have



(a) SPT Optimization



(b) DCU Optimization



(c) CCI Optimization

FIGURE 7. MLPSO-based Optimization Result for Each Level.

implemented the same objective using a single cost function. The single-level optimization by intuition is supposed to be slow due to executing the objective sequentially. The experiments are carried out, and it was found that the single-level optimization is not only on the slower end but also found to have less accuracy — the optimal cost which the single-level optimizer achieve always falls lower than the multi-level. Since the objective of the paper is maximization, therefore, the fitness of the single-level is always less. The significance

TABLE 10. Allocation of micro-tasks (operations) on devices in Optimized network.

OP No.	Device 1	Device 2	CCI
80	$\mu\tau_9$-$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_8$-$\mu\tau_1$-$\mu\tau_4$-$\mu\tau_2$-$\mu\tau_5$-$\mu\tau_6$-$\mu\tau_3$	$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_8$-$\mu\tau_9$-$\mu\tau_5$-$\mu\tau_1$-$\mu\tau_3$-$\mu\tau_2$-$\mu\tau_4$	0.79
72	$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_9$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_8$-$\mu\tau_1$-$\mu\tau_2$-$\mu\tau_6$-$\mu\tau_{10}$	$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_8$-$\mu\tau_9$-$\mu\tau_5$-$\mu\tau_1$-$\mu\tau_3$-$\mu\tau_2$-$\mu\tau_4$	0.7
64	$\mu\tau_8$-$\mu\tau_6$-$\mu\tau_5$-$\mu\tau_1$0-$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_2$-$\mu\tau_1$-$\mu\tau_9$-$\mu\tau_3$	$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_9$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_8$-$\mu\tau_1$-$\mu\tau_2$-$\mu\tau_6$-$\mu\tau_{10}$	0.6
56	$\mu\tau_7$-$\mu\tau_6$-$\mu\tau_9$-$\mu\tau_{10}$-$\mu\tau_1$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_2$-$\mu\tau_4$-$\mu\tau_8$	$\mu\tau_8$-$\mu\tau_6$-$\mu\tau_5$-$\mu\tau_{10}$-$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_2$-$\mu\tau_1$-$\mu\tau_9$-$\mu\tau_3$	0.42
48	$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_8$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_9$-$\mu\tau_1$-$\mu\tau_2$	$\mu\tau_7$-$\mu\tau_6$-$\mu\tau_9$-$\mu\tau_{10}$-$\mu\tau_1$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_2$-$\mu\tau_4$-$\mu\tau_8$	0.32
40	$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_4$-$\mu\tau_9$-$\mu\tau_8$-$\mu\tau_3$-$\mu\tau_1$-$\mu\tau_5$-$\mu\tau_2$	$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_4$-$\mu\tau_7$-$\mu\tau_8$-$\mu\tau_3$-$\mu\tau_5$-$\mu\tau_9$-$\mu\tau_1$-$\mu\tau_2$	0.31
32	$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_2$-$\mu\tau_9$-$\mu\tau_8$-$\mu\tau_4$-$\mu\tau_5$-$\mu\tau_3$-$\mu\tau_1$	$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_4$-$\mu\tau_9$-$\mu\tau_8$-$\mu\tau_3$-$\mu\tau_1$-$\mu\tau_5$-$\mu\tau_2$	0.26
24	$\mu\tau_{10}$-$\mu\tau_7$-$\mu\tau_4$-$\mu\tau_8$-$\mu\tau_9$-$\mu\tau_1$-$\mu\tau_2$-$\mu\tau_6$-$\mu\tau_3$-$\mu\tau_5$	$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_2$-$\mu\tau_9$-$\mu\tau_8$-$\mu\tau_4$-$\mu\tau_5$-$\mu\tau_3$-$\mu\tau_1$	0.19
16	$\mu\tau_4$-$\mu\tau_8$-$\mu\tau_2$-$\mu\tau_1$-$\mu\tau_5$-$\mu\tau_9$-$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_3$	$\mu\tau_{10}$-$\mu\tau_7$-$\mu\tau_4$-$\mu\tau_8$-$\mu\tau_9$-$\mu\tau_1$-$\mu\tau_2$-$\mu\tau_6$-$\mu\tau_3$-$\mu\tau_5$	0.14
8	$\mu\tau_9$-$\mu\tau_7$-$\mu\tau_{10}$-$\mu\tau_8$-$\mu\tau_1$-$\mu\tau_4$-$\mu\tau_2$-$\mu\tau_5$-$\mu\tau_6$-$\mu\tau_3$	$\mu\tau_4$-$\mu\tau_8$-$\mu\tau_2$-$\mu\tau_1$-$\mu\tau_5$-$\mu\tau_9$-$\mu\tau_{10}$-$\mu\tau_6$-$\mu\tau_7$-$\mu\tau_3$	0.11

of multi-level optimization is illustrated in table 11. Iteration column represents the iteration number and the corresponding level. For instance, iteration 10–3 denotes 10th iteration of level 3 optimization. The corresponding sequential iteration number for single-level optimizer is $10 * 1 + 10 * 2 + 10 * 3 = 60$.

The solution is found to be more slower and also is not achieving the same level as the multi-level PSO does. For instance, the highest value of cost for level 1 is 21 in contrast to 17 in single-level. The CPU time is also on the lower end.

As described in Introduction and Related Works sections, task-level orchestration refer to processes on task level rather than conventional service level. Automating process at this granular level is more flexible than on service level, and compliant with the demand of mission-critical IoT applications. For this, the processes of orchestration is simulated with and without optimization to signify the contribution of this work. We have simulated task generation and analyzed surplus time with multi-level PSO. Task mapping on devices to make an optimal operation plan is also optimized and finally the network in which devices are registered are also optimized with optimal task deployment. The results are portrayed in figure 8.

TABLE 11. Comparison of the results of Multi-level PSO with Single-level PSO.

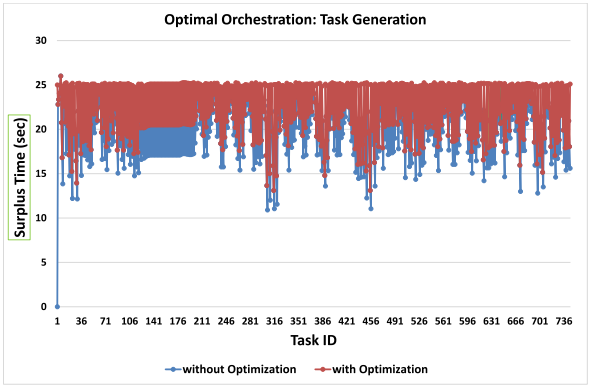
Iteration	Multi-Level PSO		Single-Level PSO	
	Cost	Time (s)	Cost	Time (s)
10–1	21	0.1	17	0.7
20–1	23	0.5	19	1.2
10–2	0.3	1.1	0.1	4.2
20–2	0.5	2.2	0.3	6.6
30–2	0.56	3.5	0.35	10.2
40–2	0.58	4.1	0.45	12.1
10–3	0.2	0.6	0.09	15.2
20–3	0.28	1.5	0.1	22.1
30–3	0.32	7.3	0.15	28.2
40–3	0.38	12.5	0.21	38
50–3	0.44	21.4	0.23	41.2
60–3	0.52	25.1	0.29	51.5
70–3	0.57	31.2	0.37	66.2
80–3	0.58	36.2	0.41	79.2

Figure 8 portrays the simulation of tasks and the arrangement with mere automation and optimal automation. The simulation is purely software-based, and the loop iterates till 740. For each iteration, the respective operation is performed with optimization using multi-level PSO and without optimization by just computing the respective metrics. The results are recorded in a file for both cases. Subfigure 8(a) illustrates the first case of task generation and surplus time analysis based on the arrangement of tasks. The vertical axis shows the surplus time. Without optimization, the surplus time is always lower than the optimized way of task generation. The red line is always above the blue line, which depicts there is always a subtle difference of surplus time with and without optimization. The surplus time is always in the range of 22 to 25 seconds, but without optimization, it is around 16 seconds on average.

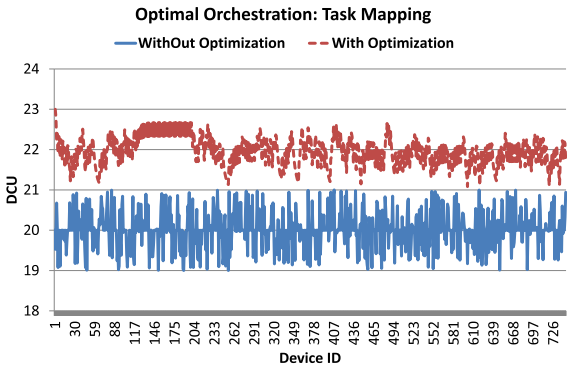
Subfigure 8(b) the task mapping on devices to form an operation plan. In the case of task generation, the difference between optimized and non-optimized orchestration was not subtle, but in the second level, the difference is quite visible. The operation plan modelling is the most crucial process in orchestration. If the operation plan is optimal, the task deployment will be efficient, and the idle time of the CPU will be minimum, and thus, more tasks can be added without any deadline-miss. The clear gap between the two lines of the graph illustrates the significance of optimization, which would mean a subtle amount of idle time and efficiency is optimized.

The last subfigure 8(c) portrays the task deployment on the optimal network. The CCI of each network is given with and without optimization. There is some overlap of the lines within the graph, but the network CCI with optimization is more than without optimization.

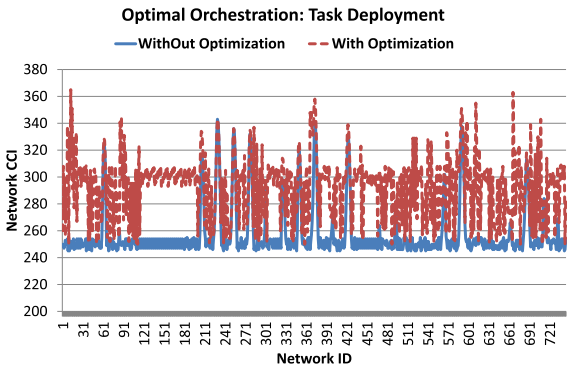
From these results, it has been signified that orchestration at the task level in an optimal way can greatly unearth the



(a) Orchestration: Task Generation



(b) Orchestration: Task Mapping Optimization



(c) Orchestration: Task Deployment

FIGURE 8. MLPSO-based Optimization Result for Each Level.

flexibility which can be easily overlooked if the orchestration is done in an ad-hoc manner. Consequently, the paper goal is a step forward for the realization of mission-critical IoT application by proposing an optimal orchestration architecture, which was clearly felt in previous studies.

VIII. DISCUSSION

A. CONTRIBUTION TO STATE-OF-THE-ART METHODS

This paper proposed a design for mission-critical IoT applications based on their unique set of requirements. The proposed approach is not only flexible and efficient but can also scale

well with the addition of more devices pairs. The proposed orchestration is flexible as in each level as it optimizes the path of schedule by maximizing the gap of the scheduling footprints with the critical path. This gap allows further changes in the schedule and also the addition of new devices. The less this gap is, the tighter will be the schedule and consequently adding a single device pair can affect the real-time behaviour of mission-critical IoT applications.

B. COMPLEXITY OF THE PROPOSED ALGORITHM FOR OPTIMAL ORCHESTRATION

In this paper, we have designed a multi-level optimizer which considers part of the solution space in each level and applies the standard PSO algorithm. The general computational complexity of PSO is in the order of $O(p(n + C))$, where p is the population size, n is the number of iteration, and C is the cost function. The proposed approach splits the solution into three levels. If true parallelism is achieved, the computational complexity of this approach is three times less than the complexity of standard PSO algorithms for the same cost function.

C. RESULTS ANALYSIS

The results based on the proposed design produces a surplus time of more than 20 seconds which in the cyber world is considered high enough to execute more than ten tasks. This flexibility leads to more robustness of the design, which is one of the vital requirements of mission-critical IoT applications. Secondly, the gap between the critical path and scheduling plan also makes the proposed approach highly fault-tolerant. In case any device failed, it has ample time to reschedule the plan and deploy on the next highly correlated device. The CCI function indicates that the network has devices of very strong correlation which, in turn, reduces the cross-network requests. This makes the proposed approach time-efficient. The design of the optimizer is also better than the general PSO on each level which makes the choice of multi-level PSO more appropriate considering mission-critical IoT applications.

D. LIMITATIONS

In this paper, it is assumed a scheduling pair of two devices, i.e., sensor and actuator. The proposed design illustrates the vertical scalability and flexibility when more device pairs are added. Consequently, it faces some limitations in terms of horizontal scalability when more devices are added to the pair. In general mission-critical IoT applications, the devices work in pair of sensor and actuator, which makes the work in this paper a valuable contribution to state-of-the-art. Nevertheless, in future mission-critical applications, the proposed work needs further investigation in terms of efficiency if the device pairs are scaled horizontally.

IX. CONCLUSION

Mission-critical IoT applications have become a prime category of modern IoT applications. Due to the distinctive requirements of these applications, their architecture plays a vital role. In this paper, an architecture is proposed to address the scalability and flexibility of mission-critical IoT application and introduced a multi-level optimized orchestration mechanism on task-level. The central goal of the optimizer is to maximize the flexibility of the system and to make it more scalable for newly added devices. Flexibility is achieved with a three-level optimizer. The first level optimizes the arrangement of the task in such a way to lead to maximum surplus time. The second level arranges devices in a group which have a stronger correlation than the rest of the possibilities. The final layer optimized the network by registering devices which lead to maximum correlation among them. The results of the paper suggested that the optimal operation plan is flexible and scalable due to the lowest possible idle time and highest possible surplus time, which has been defined as quantitative measures of flexibility and scalability. The results are also compared with single-level optimization, which has been found to be not only on the slower end but also could not achieve the same accuracy as multi-level PSO does.

CONFLICT OF INTERESTS

The authors declare that there is no conflict of interests regarding the publication of this paper.

REFERENCES

[1] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer, and P. Doody, "Internet of Things strategic research roadmap," in *Internet of Things: Global Technological and Societal Trends*, vol. 1, O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, and A. Bassi, Eds. Gistrup, Denmark: River, 2011, pp. 9–52.

[2] P. Friess, *Internet Things-Global Technological Societal Trends From Smart Environments Spaces to Green ICT*. Gistrup, Denmark River, 2011.

[3] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *Int. J. Commun. Syst.*, vol. 25, no. 9, p. 1101, Sep. 2012.

[4] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, and R. and Spalazzese, "Model-driven engineering for mission-critical iot systems," *IEEE Softw.*, vol. 1, no. 1, pp. 46–53, Jan./Feb. 2017.

[5] S. Ahmad, S. Malik, I. Ullah, M Fayaz, D.-H. Park, K. Kim, and D. Kim, "An Adaptive approach based on resource-awareness towards power-efficient real-time periodic task modeling on embedded IoT devices," *Processes*, vol. 6, no. 7, p. 90, Jul. 2018.

[6] S. Ahmad, S. Malik, I. Ullah, D.-H. Park, K. Kim, and D. Kim, "Towards the design of a formal verification and evaluation tool of real-time tasks scheduling of IoT applications," *Sustainability*, vol. 11, no. 1, p. 204, Jan. 2019.

[7] S. Malik, S. Ahmad, I. Ullah, D. H. Park, and D. Kim, "An adaptive emergency first intelligent scheduling algorithm for efficient task management and scheduling in hybrid of hard real-time and soft real-time embedded IoT systems," *Sustainability*, vol. 11, no. 8, p. 2192, Apr. 2019.

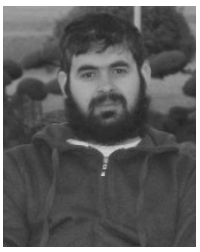
[8] M. Rostan, J. E. Stubbs, and D. Dzilno, "Ethercat enabled advanced control architecture," in *Proc. IEEE/SEMI Adv. Semiconductor Manuf. Conf. (ASMC)*, Jul. 2010, pp. 39–44.

[9] D. Jansen and H. Buttner, "Real-time Ethernet: The ethercat solution," *Comput. Control Eng.*, vol. 15, no. 1, pp. 16–21, Feb. 2004.

[10] S. Ahmad, F. Mehmood, and D.-H. Kim, "A DIY approach for the design of mission-planning architecture using autonomous task–object mapping and the deployment model in mission-critical IoT systems," *Sustainability*, vol. 11, no. 13, p. 3647, Jul. 2019.

[11] K. Velasquez, D. P. Abreu, D. Gonçalves, L. Bittencourt, M. Curado, E. Monteiro, and E. Madeira, "Service orchestration in fog environments," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2017, pp. 329–336.

[12] Z. Ding, K. Ota, Y. Liu, N. Zhang, M. Zhao, H. Song, A. Liu, and C. Zhiping, "Orchestrating data as a services-based computing and communication model for information-centric Internet of Things," *IEEE Access*, vol. 6, pp. 38900–38920, 2018.
[13] *How far is the Hype of IoT*. Accessed Mar. 15, 2019. [Online]. Available: https://www.rcrwireless.com/20160628/opinion/reality-check-50b-iot-devices-connected-2020-beyond-hype-reality-tag10
[14] A. Orsino, I. Farris, L. Militano, G. Araniti, S. Andreev, I. Gudkova, Y. Koucheryavy, and A. Iera, "Exploiting D2D communications at the network edge for mission-critical IoT applications," in *Proc. Eur. Wireless 23th Eur. Wireless Conf.*, May 2017, pp. 1–6.
[15] A. Orsino, A. Ometov, G. Fodor, D. Moltchanov, L. Militano, S. Andreev, O. N. C. Yilmaz, T. Tirronen, J. Torsner, G. Araniti, A. Iera, M. Dohler, and Y. Koucheryavy, "Effects of heterogeneous mobility on D2D- and drone-assisted mission-critical MTC in 5G," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 79–87, Feb. 2017.
[16] Q. Zhang and F. H. Fitzek, "Mission critical iot communication in 5G," in *Future Access Enablers of Ubiquitous and Intelligent Infrastructures*. Cham, Switzerland: Springer, 2015, pp. 35–41.
[17] C. Starkey and C. Garvin, "Knowledge from data in the built environment," *Ann. New York Acad. Sci.*, vol. 1295, no. 1, pp. 1–9, Aug. 2013.
[18] A. Bratukhin and A. Treytl, "Applicability of RFID and agent-based control for product identification in distributed production," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom.*, Sep. 2006, pp. 1198–1205.
[19] M. Huang, Y. Liu, N. Zhang, N. N. Xiong, A. Liu, Z. Zeng, and H. Song, "A services routing based caching scheme for cloud assisted CRNs," *IEEE Access*, vol. 6, pp. 15787–15805, 2018.
[20] X. Liu, Y. Liu, H. Song, and A. Liu, "Big data orchestration as a service network," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 94–101, Sep. 2017.
[21] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3512–3524, Apr. 2018.
[22] W. Li, F. C. Delicato, P. F. Pires, Y. C. Lee, A. Y. Zomaya, C. Miceli, and L. Pirmez, "Efficient allocation of resources in multiple heterogeneous wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1775–1788, Jan. 2014.
[23] E. A. Khalil, S. Ozdemir, and S. Tosun, "Evolutionary task allocation in Internet of Things-based application domains," *Future Gener. Comput. Syst.*, vol. 86, pp. 121–133, Sep. 2018.
[24] V. Pilloni, E. A.-Elrahman, M. Hadji, L. Atzori, and H. Afifi, "IoT_ProSe: Exploiting 3GPP services for task allocation in the Internet of Things," *Ad Hoc Netw.*, vol. 66, pp. 26–39, Nov. 2017.
[25] H. Ali, X. Zhai, U. U. Tariq, and L. Liu, "Energy efficient heuristic algorithm for task mapping on shared-memory heterogeneous mpsocs," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun., IEEE 16th Int. Conf. Smart City, IEEE 4th Int. Conf. Data Sci. Syst. . (HPCC/SmartCity/DSS)*, Jun. 2018, pp. 1099–1104.
[26] B. Billet and V. Issarny, "From task graphs to concrete actions: A new task mapping algorithm for the future Internet of Things," in *Proc. IEEE 11th Int. Conf. Mobile Ad Hoc Sensor Syst.*, Oct. 2014, pp. 470–478.
[27] R. Kuo and Y. Han, "A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem—A case study on supply chain model," *Appl. Math. Model.*, vol. 35, no. 8, pp. 3905–3917, Aug. 2011.
[28] L. Chen, T. Monteiro, T. Wang, and E. Marcon, "Design of shared unit-dose drug distribution network using multi-level particle swarm optimization," *Health care Manage. Sci.*, vol. 22, no. 2, pp. 304–317, Jun. 2019.
[29] C.-B. Cheng, H.-S. Shih, and E. S. Lee, "Metaheuristics for multi-level optimization," in *Fuzzy Multi-Level Decision Making: Soft Computing Approaches*. Cham, Switzerland: Springer, 2019, pp. 171–188.
[30] H. Garg and S. Sharma, "Multi-objective reliability-redundancy allocation problem using particle swarm optimization," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 247–255, Jan. 2013.
[31] W.-C. Yeh, "A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems," *Expert Syst. Appl.*, vol. 36, no. 5, pp. 9192–9200, Jul. 2009.
[32] S. Suresh, P. Sujit, and A. Rao, "Particle swarm optimization approach for multi-objective composite box-beam design," *Composite Struct.*, vol. 81, no. 4, pp. 598–605, Dec. 2007.
[33] S. C. Satapathy, N. S. M. Raja, V. Rajinikanth, A. S. Ashour, and N. Dey, "Multi-level image thresholding using Otsu and chaotic bat algorithm," *Neural Comput. Appl.*, vol. 29, no. 12, pp. 1285–1307, 2016.
[34] J. Yi, L. Gao, X. Li, C. A. Shoemaker, and C. Lu, "An on-line variable-fidelity surrogate-assisted harmony search algorithm with multi-level screening strategy for expensive engineering design optimization," *Knowl.-Based Syst.*, vol. 170, pp. 1–19, Apr. 2019.
[35] A. Ahilan, G. Manogaran, C. Raja, S. Kadry, S. N. Kumar, C. A. Kumar, T. Jarin, S. Krishnamoorthy, P. M. Kumar, G. C. Babu, N. S. Murugan, and Parthasarathy, "Segmentation by fractional order darwinian particle swarm optimization based multilevel thresholding and improved lossless prediction based compression algorithm for medical images," *IEEE Access*, vol. 7, pp. 89570–89580, 2019.



SHABIR AHMAD received the B.S. degree in computer system engineering from the University of Engineering and Technology, Peshawar, Pakistan, and the M.S. degree in computer software engineering from the National University of Science and Technology, Islamabad, Pakistan, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, Jeju National University, South Korea. He is also a Faculty Member with the Software Engineering Department, University of Engineering and Technology. His research interests include the Internet of Things application, cyber-physical systems, and intelligent systems.

AZIMBEK KHUDOYBERDIEV received the B.S. degree from the Tashkent University of Information Technologies, Uzbekistan, in 2018. He is currently pursuing the master's degree in computer engineering with Jeju National University, South Korea. His research interests include the development of IoT, optimization, and AI-based smart systems.

DOHYEUN KIM received the B.S. degree in electronics engineering and the M.S. and Ph.D. degrees in information telecommunication from Kyungpook National University, South Korea, in 1988, 1990, and 2000, respectively. He joined the Agency of Defense Development (ADD), from March 1990 to April 1995. Since 2004, he has been with Jeju National University, South Korea, where he is currently a Professor with the Department of Computer Engineering. From 2008 to 2009, he was with the Queensland University of Technology, Australia, as a Visiting Researcher. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service, and mobile computing.