Sustainable Service Allocation using Metaheuristic Technique in Fog Server for Industrial Applications

Sambit Kumar Mishra, *Student Member, IEEE*, Deepak Puthal, *Member, IEEE*, Joel J. P. C. Rodrigues, *Senior Member, IEEE*, Bibhudatta Sahoo, *Member, IEEE*, and Eryk Dutkiewicz *Member, IEEE*

***Abstract*—Reducing energy consumption in the fog computing environment is both a research and an operational challenge for the current research community and industry. There are several industries such as finance industry or health care industry required rich resource platform to process big data along with edge computing in fog architecture. As a result, sustainable computing in fog server plays a key role in fog computing hierarchy. The energy consumption in fog servers depends on the allocation techniques of services (user requests) to a set of virtual machines (VMs). This service request allocation in a fog computing environment is a non-deterministic polynomial-time hard (NP-hard) problem. In this paper, the scheduling of service requests to VMs is presented as a bi-objective minimization problem, where a trade-off is maintained between the energy consumption and makespan. Specifically, this paper propose a metaheuristic-based service allocation framework using three metaheuristic techniques, such as Particle Swarm Optimization (PSO), Binary PSO (BPSO) and Bat algorithm (BAT). These proposed techniques allow us to deal with the heterogeneity of resources in the fog computing environment. This paper has validated the performance of these metaheuristic based service allocation algorithms by conducting a set of rigorous evaluations.**

***Index Terms*—Fog Computing, Cloud computing, Metaheuristic techniques, Service allocation problem, PSO, BAT, BPSO.**

I. Introduction

FOG computing presents some of the overlapping peculiarities of cloud with additional attributes such as location awareness and edge data center deployment. In recent days, finance industries such as SCADA (Supervisory Control and Data Acquisition) use edge datacenters in fog architecture for low latency. At the same time, such applications use fog servers for batch processing of big data [1]. Fog computing supports distributed computing solutions to achieve high scalability, elasticity, reduced computing cost, efficient sharing of information, etc. Consequently, fog sever resource management can be extremely challenging due to the complexity of the system. In recent days, fog computing has been proposed to utilize the cloud resources efficiently across the networks. This brings sustainability to the industrial data processing. Execution environment in a fog infrastructure is presented as a virtual machine (VM). The virtualization technique allows one to create virtual instances of a device or resource, where the framework partitions the resource into multiple execution environments virtually in the form of VMs. The technology underlying virtualization is a Virtual Machine Monitor (VMM) or hypervisor, which separates the computing environments from the actual physical infrastructure. It includes modifying the operating system to substitute non-virtualizable instructions with hypercalls to communicate immediately. Fog computing offers an intermediate fog layer between the users and cloud resources [2]. This paper uses the three-layer fog computing architecture [3] to propose sustainable service allocations in fog server. The bottom layer comprises several terminal devices such as wireless sensor nodes and smart devices, where the user submit their tasks to the upper layers. In the second layer, the fog includes highly intelligent devices like routers, switches, and gateways (edge data centers). The topmost layer points to be the cloud data center comprising several high-end servers (fog server).

The energy consumed in the fog computing system is mostly from the execution environment, cooling equipment, and power conditioning [4]. Energy consumption is becoming a pivotal issue for the service, performance, and maintenance of fog servers. We rely mostly on fossil fuels for energy. For example, according to [5], the energy used in fog servers is generated from 39% petroleum products, 24% natural gas, 23% coal, 8% nuclear, and 6% other. As fossil fuels are non-renewable energy sources, we need to use the energy optimally. A typical data center deployment consumes a significant amount of energy, and in turn, increases the carbon dioxide (CO_2) level indirectly. For example, it was estimated that approximately 8% of the global emission is from CO_2 emission [6], [7], which is one of the factors contributing to global warming. The required Quality of service (QoS) can be achieved through optimizing computing resource utilization. Current days industrial data are mainly considered to be evaluated at data centers. This paper mainly focusing on energy consumption and makespan as QoS constraints. To ensure the QoS, the system needs to map all tasks or services to the available resources efficiently.

Energy consumption is directly proportional to the resource utilization of data centers [4]. In the same way, cost is directly proportional to the energy consumption (i.e. when energy consumption increases, so does cost) [8]. To solve the NP-hard nature of the service allocation problem, it require suboptimal solutions as these are generally the most efficient and effective techniques. Therefore, researchers have presented a number of heuristic procedures to address the issue [2], [9]–[11]. Heuristic techniques are problem-dependent and because of their greedy nature, they are normally trapped in a

S. K. Mishra and B. Sahoo are with National Institute of Technology, Rourkela, India (e-mail: {skmishra.nitrkl, bibhudatta.sahoo}@gmail.com).

D. Puthal and E. Dutkiewicz are with University of Technology, Sydney, Australia (e-mail: {deepak.puthal,eryk.dutkiewicz}@uts.edu.au).

J. J. P. C. Rodrigues is with National Institute of Telecommunications (Inatel), Brazil and Instituto de Telecomunicaes, Portugal (e-mail:joeljr@ieee.org).

local optimum and hence, fail to reach the global optimum solution. Metaheuristic techniques are efficient as they are problem-independent, but not greedy. All metaheuristic techniques utilize a trade-off between randomization and local search. Randomization provides a decent approach to run away from local search to the global one, and comprises intensification and diversification of any metaheuristic algorithms. The diversification mechanism allows one to explore the search space globally by generating diverse solutions, while intensification searches for a decent solution in the local region. The combination of these two components will mostly guarantee the achievability of the global optimality. There are various mechanisms to reduce energy consumption in the fog server. Popular techniques include VM allocation, multicore architecture-based, service consolidation-based, thermal-aware-based, power-aware management, and bio-inspired computation-based techniques [12], [13]. In the service allocation problem, the allocation of services or tasks to different systems (or VMs) requires the implementation of scheduling policies [14]. Therefore, the scheduler is responsible for optimal allocation of services to different VMs with the minimum energy consumption [4]. The user requests (services) are allocated to a set of VMs for their execution.

This paper seeks to minimize energy consumption within a fog server, without compromising its capability to deliver services. The contributions of this paper are organized as follows:

- Initially, this paper presents a general fog system model, including host model, VM model, and service model.
- This paper formulates the Linear Programming Problem (LPP) for optimization of makespan and energy, and frame three metaheuristic based service allocation algorithms for the heterogeneous fog server system delivering heterogeneous service or task.
- Finally, this paper evaluates the proposed algorithms in terms of makespan, energy consumption, and the performance.

The remaining of this paper is organized as follows. Section II briefly discusses related work. In Section III, the fog system model is presented. The service allocation problem is explained in Section IV. Section V describes some performance metrics along with an example. A brief description of metaheuristic techniques and its utilization for the allocation problem are presented in Section VI, followed by simulation and experimental findings in Section VII. Finally, this paper give the conclusion in Section VIII.

II. Related Work

The increased adoption of fog and cloud computing in industrial data processing has resulted in an urgent need to examine ways of reducing CO_2 emission due to the significant amount of energy consumed at data centers, etc. The emission of CO_2 from a data center depends on the power plants that are directly used to deliver electricity to the data center. It was estimated in 2013 that data centers in U.S. consumed 91 billion kilowatt-hours of electricity and equates an annual output of 34 large (500-megawatt) coal-fired power plants, and this amount of electricity can reportedly power all houses in New York City twice over for a year [15]. The annual consumption of electricity by data centers is predicted to increase to 140 billion kilowatt-hours by 2020 [15]. This has resulted in one popular research area i.e., resource allocation techniques designed for fog server or virtualized server systems [2], [12], [16], [17]. Souza *et al.* [16] have proposed a strategy for the service allocation problem as a multidimensional knapsack problem. They have applied their model for the integration of fog and cloud computing to optimize delay, load and energy consumption. The workload allocations to the fog-cloud approaching the optimal power consumption with a fixed delay is suggested by Deng *et al.* [2]. They have investigated that the fog framework significantly improves the cloud computing performance.

The metaheuristic technique presented by Tsai and Rodrigues depends on the search space and the decision capability (intelligence) [18]. The technique searches for local optimum prior to searching for the global optimum. The authors focused on the use of metaheuristic algorithms to schedule tasks. They have summarized that metaheuristic techniques provide a better result as compared to deterministic methods. However, it is known that metaheuristic algorithms are slower, and the resulting solutions are not always optimal. Guo *et al.* presented a task scheduling optimization method for the cloud, which is based on PSO to minimize the cost [10]. The most direct and efficient method is to make use of more power efficient components during the hardware design phase. Other alternative proposals include the design of algorithms to scale down power or even shutting down of a system when it is not in use.

Bergmann *et al.* [17] presented a Simplified Swarm Optimization (SSO) scheme designed to consume less energy in distributed systems with dynamic voltage scaling. The proposed algorithm reportedly achieves minimum execution time and makespan within a reasonable time without delay. Kaur and Chana presented a comprehensive review of energy-efficient techniques designed for cloud computing [12]. The techniques were classified into several categories, including bio-inspired optimization method (e.g. Ant Colony Optimization (ACO), Genetic Algorithm (GA), swarm based optimization methods). Zhan *et al.* described the resource scheduling problem and their solutions using several nature-inspired evolutionary methods (e.g. ACO, GA, PSO) [19]. For a cloud system, the proportion of energy utilization of processing servers versus the energy utilization of storage servers and network segments is 75:15:10 [20]. This means that from the total energy consumption due to the data center, 75% is due to CPU-centric servers, 15% is due to storage servers and 10% is due to network components. Researchers provide different heuristic and metaheuristic techniques for the allocation problem in the fog computing environment, and among the metaheuristic techniques, PSO is most widely used.

III. Fog System Model

The fog computing architecture is highly scalable, is an abstract entity, and delivers different levels of services to the cloud user, achieve economies of scale, and delivers on-

demand and dynamic contents and services through virtualization. Fog computing offers cloud resources (e.g., servers, networks, applications, storage, and services) over the Internet, which can be rapidly provisioned and released with minimal management effort or service provider interaction. The system architecture of a single host in the fog server (cloud) with three layers is depicted in Fig. 1. The hardware layer consists of raw hardware resources (e.g. processor, main memory, secondary storage, and network bandwidth), which are virtualized. VMM or hypervisor like Xen, VMware, UML, and Denali act as an interface between the host operating system and VMs. The VMM also allows multiple operating systems to run applications on a single hardware platform concurrently. Different number of heterogeneous applications can run on each guest operating system or VM.

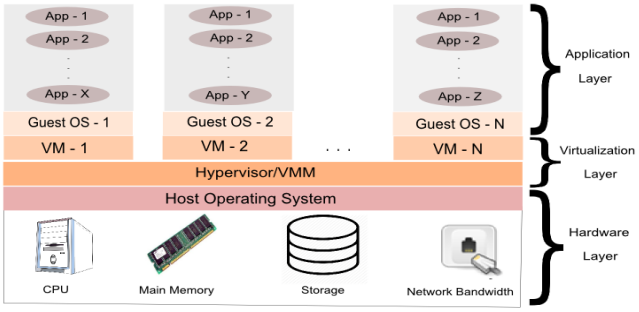


Fig. 1: Single Host Architecture in Fog Server (Cloud)

A data center has several physical servers and there is an interconnection of high-speed LAN-network and high bandwidth link to the Internet from each physical server [23]. For a fog server, each host has a unique identification number, number of processing elements or the number of CPU cores, main memory, bandwidth, and secondary storage. The system can scale the resource to fulfill the required QoS and SLAs. A VM is an emulation of a particular host. There are m number of VMs $(V_1, V_2, ..., V_m)$ running on various hosts. For a host, each VM has a unique id, processing element, main memory, bandwidth, secondary storage. A service request from the user is assigned to one VM only, and no service migration is allowed in this architecture. Let S be the set of n services $\{S_1, S_2, ..., S_n\}$. Each service S_i can be modeled as service id, workload (in terms of MIPS (million instructions per second)), the requirement of CPU time, main memory, and bandwidth. All services are mapped to the VMs on a one-to-many basis.

TABLE I: An ETC (Expected Time to Compute) matrix with m VMs and n service requests

An ETC matrix with n services and m VMs		V_1	V_2	...	V_j	...	V_m
	S_1	ETC_{11}	ETC_{12}	...	ETC_{1j}	...	ETC_{1m}
	S_2	ETC_{21}	ETC_{22}	...	ETC_{2j}	...	ETC_{2m}
	.	...	...	...	...	...	...
ETC =	S_i	ETC_{i1}	ETC_{i2}	...	ETC_{ij}	...	ETC_{im}
	.	...	...	...	...	...	...
	S_n	ETC_{n1}	ETC_{n2}	...	ETC_{nj}	...	ETC_{nm}

An ETC (Expected Time to Compute) matrix holds the time which is expected to compute a specific service (task) on different VMs as shown in TABLE I (where n number of services have different ETC time for m number of VMs). The element ETC_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$ denotes the expected time to compute the i^{th} service on j^{th} VM. To generate the ETC matrix, this paper followed an algorithm presented by Maheswaran *et al.* [21]. According to the resource requirement of the services, ETC matrix has some value. So, the ETC matrix represents the service request heterogeneity.

IV. Service Allocation Problem

Service requests originate from multiple users over the Internet. The challenge of allocating service requests to a set of VMs running on different hosts while achieving the terms and conditions stated in the SLAs and without degrading the QoS is referred to as the service allocation problem [16]. To design an energy efficient solution for the service allocation problem, the following assumptions are made for the system:

- Expected time to compute (ETC) value will be the total time (computation time + communication time) taken by any service on a VM. All the service requests are independent and heterogeneous.
- All the VMs are heterogeneous in terms of their resource capabilities.
- The system has adequate resources (VMs) for all accepted service requests.
- A VMM (Xen) is running on the top of each host.
- A service is allowed to execute only on a single VM.

This paper consider a heterogeneous fog server system that consists of a set of $H = \{H_1, H_2, ..., H_k\}$, k independent heterogeneous, uniquely addressable computing entity (hosts). It has a set of $V = \{V_1, V_2, ..., V_m\}$, m heterogeneous VMs, and for simplicity, this consider each host has a single VM (i.e., V_j runs on H_j). Let there be $S = \{S_1, S_2, ..., S_n\}$, n number of heterogeneous services, where each service S_i has a service length L_i in terms of a million instructions (MI). S_{ij} is the expected time to compute service S_i on VM V_j. Each VM is capable of executing all types of services. This can be represented by an ETC matrix of size $n \times m$, where n is the number of services and m is the number of VMs. In the ETC matrix, the elements along a row indicate the execution time of a given service on different VMs in seconds. Each VM V_j has a processing speed P_j in MIPS. Then, the ETC_{ij} is $L_i \div P_j$, where L_i is the service length of S_i and P_j is the processing speed of V_j, $1 \leq i \leq n$, $1 \leq j \leq m$. When a VM processes a task, it is either active or idle. Energy consumption of idle state is 60% of the active state of VMs [22], and energy consumed (Joules (J)) by the VM V_j is represented as

$$= \begin{cases} \beta_j \text{ Joules/MI} & \text{if } V_j \text{ is in active state,} \\ \alpha_j \text{ Joules/MI} & \text{if } V_j \text{ is in idle state.} \end{cases} \quad (1)$$

where $\beta_j = 10^{-8} \times (MIPS_j)^2$ [24] and $\alpha_j = 0.6 \times \beta_j Joules/MI$. The assignment of a service to a VM is given in equation (2). Here, X_{ij} is a decision variable based on whether a task is allocated to a specific VM or not. If the service S_i is allocated to VM V_j, then the value of X_{ij} is 1, otherwise 0.

$$X_{ij} = \begin{cases} 1 \text{ if } S_i \text{ is allocated to } V_j \\ 0 \text{ if } S_i \text{ is not allocated to } V_j \end{cases} \quad (2)$$

Where $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$.
Total execution time (ET_j) of all the services assigned to j^{th} VM is in equation (3) as follows.

$$ET_j = \sum_{i=1}^{n} X_{ij} \times ETC_{ij} \tag{3}$$

Makespan (M) is the maximum execution time among all the VMs i.e. Makespan, M = maximum(ET_j), $1 \leq j \leq m$. The aggregate energy consumption (i.e., the sum of energy consumption in the active and idle state) of a VM V_j in terms of Joules per MI is in equation (4) as follows.

$$E(V_j) = [ET_j \times \beta_j + (M - ET_j) \times \alpha_j] \times MIPS_j \tag{4}$$

The total energy consumption of the system is in equation (5).

$$\varepsilon = \sum_{j=1}^{m} E(V_j) \tag{5}$$

Our objective is to minimize the total cost, ξ, i.e. to minimize the cost of energy consumption, and the cost of makespan of the fog server system. This problem is a bi-objective problem, and it can be represented as LPP problem formulated in equation (6). Here, ∂ is the penalty value multiplied with the makespan (M), i.e., the cost of using cloud resources per unit time. Similarly, φ is the penalty value multiplied with the total energy consumption (ξ) of the system to determine the cost of the consumption of energy. The penalty values φ are higher than ∂ when energy consumption is more valuable than the makespan of the system, and vice-versa.

$$Minimize\ \xi = M \times \partial + \varepsilon \times \varphi \tag{6}$$

The problem of allocating services to the available computing resources is a well known NP-hard problem [25], where the main objective is the minimum utilization of energy. The minimization of energy consumption and makespan requires a proper allocation between all services and VMs, which is clearly a bin-packing problem. This bin-packing problem itself is a known NP-hard problem. Several proposed methods have used various heuristic techniques to achieve some near optimal solutions, where time complexity is also reasonable [26]. A survey of different metaheuristics techniques can be found in [18] for service allocation problem in the cloud environment.

V. Performance Metrics

To evaluate the performance of the above-discussed meta-heuristic algorithms, this paper use makespan and energy consumption as two performance metrics. Here, this paper also introduces some other metrics, namely flow-time and cost of using resources for the performance measurement of the algorithms. All metrics for the service allocation in the fog server are briefly discussed with an example as follows.

TABLE II: Services with their Size

Task Id	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
Task Size(MI)	2000	4800	4500	8000	19500	9600	4000	3600	6000	3600

TABLE III: VMs with their Processing Speed

VM Id	V_1	V_2	V_3
VM Speed(MIPS)	1000	1200	1500

A scheduling example having ten independent services or tasks (T1, T2, ..., T10) allocated to three different VMs (V_1, V_2, V_3) is presented. All task have different task length in terms of MI as in TABLE II, and all three VMs have different processing speed in terms of MIPS as in TABLE III. The ETC matrix for ten tasks and three VMs of size 10×3 matrix is shown in TABLE IV.

TABLE IV: ETC matrix for ten tasks and three VMs

	V_1	V_2	V_3
T1	2	1.66	1.33
T2	4.8	4	3.2
T3	4.5	3.75	3
T4	8	6.66	5.33
T5	19.5	16.25	13
T6	9.6	8	6.4
T7	4	3.33	2.66
T8	3.6	3	2.4
T9	6	5	4
T10	3.6	3	2.4

One of the allocation result is shown in Fig. 2 below.

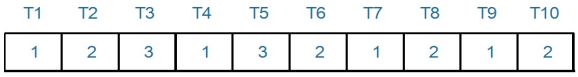


Fig. 2: An allocation of ten tasks to three VMs

After allocating tasks to three VMs, each VM takes some unit of time to complete their execution as shown in Fig. 3.

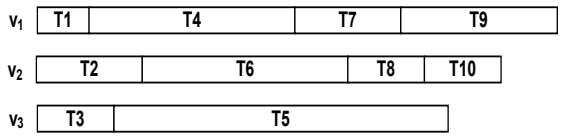


Fig. 3: The completion time of ten tasks in three VMs (v_1, v_2, v_3) based on the allocation as shown in Fig. 2.

A. Makespan

Makespan is the total time required to process all tasks assigned to the fog server system. To calculate the execution time taken by individual VM (ET_j, $1 \leq j \leq m$), ETC matrix is required. The expression for makespan calculation is expressed in equation (3) with the help of equation (2), where X is a Boolean variable as explained before. Now, the makespan (M) is the maximum time needed by a VM over all VMs to execute all tasks in the fog server. Mathematically, M = Max(ET_j), $1 \leq j \leq m$. By considering the above example, it has an allocation result as shown in Fig. 2 and it obtain the matrix X of 10×3 as shown in TABLE V.

TABLE V: The allocation matrix X

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

From the example in Fig. 3, ET_1 (execution time of V_1) is 20, ET_2 is 18, ET_3 is 16 after completion of all ten tasks. So, the makespan (M) of the schedule is 20 time units, which is the finishing time of T9.

B. Energy Consumption

It has been estimated that physical servers utilize around 45% and networking devices utilize around 15% of the total energy [27]. The energy consumption of a physical server comes from resources, such as CPU, RAM, and storage. In each physical server, several VMs are running to execute a large number of services. Therefore, to calculate the total energy consumption (ε) of a fog server system, this has to include energy consumption of all VMs running on that system by calculating at the VM level. The energy consumption of the scheduling is the summation of energy consumption of all VMs (V_1, V_2, V_3) as shown in Fig. 3. For this, it has to calculate β_j and α_j, $1 \leq j \leq 3$ by using equation (1) as follows.

$\beta_1 = 10^{-8} \times (1000)^2 = 0.01$ Joules/MI,
$\Rightarrow \quad \alpha_1 = 0.6 \times \beta_1 = 0.006$ Joules/MI
$\beta_2 = 10^{-8} \times (1200)^2 = 0.0144$ Joules/MI,
$\Rightarrow \quad \alpha_2 = 0.6 \times \beta_2 = 0.00864$ Joules/MI
$\beta_3 = 10^{-8} \times (1500)^2 = 0.0225$ Joules/MI,
$\Rightarrow \quad \alpha_3 = 0.6 \times \beta_3 = 0.0135$ Joules/MI

The energy consumption of the schedule is the summation of $E(V_1)$, $E(V_2)$, $E(V_3)$, and calculated using equation (4).
$E(V_1) = [ET_1 \times \beta_1 + (M - ET_1) \times \alpha_1] \times MIPS_1 = 200$ J
$E(V_2) = [ET_2 \times \beta_2 + (M - ET_2) \times \alpha_2] \times MIPS_2 = 331.776$ J
$E(V_3) = [ET_3 \times \beta_3 + (M - ET_3) \times \alpha_3] \times MIPS_3 = 621$ J
Therefore, the total energy consumption for the above schedule is 1152.776 Joules i.e, 1.1528 Kilo Joules.

C. Flow-time

Flow-time is an important optimization metric, and it is the sum of finishing time of all services. It can be calculated as: Flow-time = $\sum_{i=1}^{n} FT_i$, where FT_i is the finishing time of the i^{th} service. The execution of services should be in ascending order of their execution time for the minimization of the flow-time. The flow-time value for the example shown in Fig. 3 is the finishing time of all services.
Flow-time = FT_1 + FT_2 + FT_3 + FT_4 + FT_5 + FT_6 + FT_7 + FT_8 + FT_9 + FT_{10} = 2 + 4 + 3 + 10 + 16 + 12 + 14 + 15 + 20 + 18 = 114 time units.

D. Cost of using resources

Another important metric is the cost of using resources, which can be quantified using $\sum_{j=1}^{m} C_j \times ET_j$, where C_j is the cost of usage of j^{th} VM and ET_j is the j^{th} VM utilization time. If the cost of using resource v_1 is 1000 per unit time, v_2 is 1500 per unit time, and v_3 is 2000 per unit time, then the cost = 20 $\times$ 1000 + 18 $\times$ 1500 + 16 $\times$ 2000 = 79000.

VI. Service Allocation using Metaheuristic Techniques

Metaheuristic techniques are based on the behavioral instincts of different particles. One of the advantages of metaheuristic technique is that it provides a near optimal solution within a short period. The studied literature mainly focuses on a range of optimization criteria such as reduced makespan, minimum cost and response time, maximum throughput, and reduced energy consumption without violating SLA. Researchers have used different metaheuristic techniques for the optimization of the allocation problem. For example, Guo *et al.* used PSO [10], Wang *et al.* proposed a Particle Swarm Optimized Tabu search technique [28], and Bergmann *et al.* used SSO [17] for the allocation problem. PSO is the most popular metaheuristic technique utilized by different researchers to solve numerous issues, for example, task consolidation problems in the fog environment.

This paper attempt to design new frameworks for service allocation problem using PSO, BPSO, BAT. The following subsections describe these techniques.

A. Service Allocation Algorithm using PSO

PSO is a bio-inspired computational technique based on the social behavior of the organism/ particles. For example in bird flocking and fish schooling, each bird or fish is referred to as a particle and their behavioral instincts (e.g. sudden direction change, scattering, and regrouping) are studied for the purpose of computing. Each particle uses its local best and global best to adjust its current position and velocity in each iteration. The main advantage of PSO is that it is simple to use and can be applied to a broad range of applications with low computational cost. The velocity vector specifies the movement in search space. The velocity vector value of the object is updated in each iteration based on its previous velocity and its distance from its own best position (Pbest) and the best position (Gbest) between neighbours (existing allocation results). The particle (or the allocation result) is able to search around its Pbest and Gbest using regular update velocity.

The PSO-based Service Allocation Algorithm (PSO_SAA) customized to optimize scheduling problem in the fog server is given in Algorithm 1. The algorithm PSO_SAA accepts a set of service requests in the form of ETC matrix and processing speed of m number of VMs, and results the allocation of tasks to VMs. Each solution (allocation of services to VMs) is mapped to a particle. It is represented by $1 \times n$ vector, where n is equal to the number of tasks. Each element (position) in the particle can have an integer value between 0 and m, where m is the number of VMs. The individual position of a particle shows a mapping between a task and a VM. An example of a position of a particle with ten tasks and three VMs is shown in Fig. 4, based on the earlier example in Fig. 3, where each position of the vector represents a VM number.

In the next step, it generates an initial population randomly. The global velocity, the local velocity and position vector of each particle are initialized. Since PSO is designed for a continuous optimization problem, this need to revise it to apply

Algorithm 1 : PSO_SAA

Input: ETC matrix, processing Speed of VMs.

Output: Allocation result of services to VMs, Makespan, Energy, ξ.

1: Initialize random population as a service allocation vector as shown in Fig. 4, global velocity (C_2), local velocity vector (C_1) and velocity vector (V) for each particle in a population.
2: Convert all continuous vector to discrete vector including the service allocation vector.
3: Calculate fitness value (ξ) for each particle using equation (6).
4: P_{best} = Best position value for each particle.
5: G_{best} = Minimum fitness value from the set of service allocation vectors.
6: **for** each particle update the position and velocity vector **do**
7: $V_{i+1} = V_i + C_1 \times rand_1 \times [P_{best} - X_i] + C_2 \times rand_2 \times [G_{best} - X_i]$.
8: $X_{i+1} = X_i + V_{i+1}$.
9: **end for**
10: Repeat steps 2 to 9 until the termination condition is satisfied.

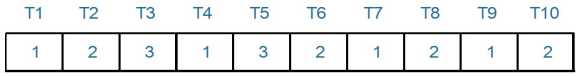


Fig. 4: An example of a position of a particle

in a discrete optimization problem. This paper then calculate the fitness value (ξ) of each particle in the population and update local best (Pbest) and global best (Gbest). The Pbest is the minimum ξ calculated for the particle over iterations and Gbest is the smallest ξ in the population. For each particle, we update velocity (V_{i+1}) and position vector (X_{i+1}) using steps 7 and 8, respectively. Two random values $rand_1$ and $rand_2$ are used in step 7 for variations in movement towards the optimal result locally and globally. This process is continued until the termination condition is satisfied (e.g. maximum iteration or getting the required result).

B. BPSO-based Service Allocation

Binary PSO is a variation of PSO, where the population is converted to binary vector. A service allocation algorithm ($BPSO\_SAAP$) for n service requests to m VMs is explained in Algorithm 2. This algorithm accepts a set of tasks in the form of ETC matrix and the speed of VMs and provides the allocation results with the values of performance metrics. If m number of VMs are present, then $log_2(m)$ bits are used to represent each element of a particle in a binary vector. Each particle update zeros and ones of velocity vector according to steps 7 to 16. Then velocity vector is updated and is converted to log sigmoid distribution to get the normalized velocity vector (V). After the position vector (X) is updated according to steps 27 to 35, it is converted to decimal vector. Then, the boundary value is checked and updated, if required.

Algorithm 2 : $BPSO\_SAA$

Input: ETC matrix, processing Speed of VMs.

Output: Allocation result of services to VMs, Makespan, Energy, ξ.

1: Initialize random population as a service allocation vector, global velocity (C_2), local velocity (C_1), zero velocity ($zerovel$), one velocity ($onevel$) and velocity vector (vel) for each particle in a population.
2: Convert all continuous vector to discrete vector including service allocation vector.
3: Calculate fitness value (ξ) for each particle using equation (6).
4: XP_{best} = Best position value for each particle.
5: XG_{best} = Minimum fitness value in the population.
6: For each particle update the position and velocity vector as follows.
7: $zerovel = zerovel - C_1 \times rand[XP_{best}. \times (-1)] - C_2 \times rand[XG_{best}. \times (-1)]$
8: $onevel = onevel - C_1 \times rand[XP_{best}. \times (-1)] + C_2 \times rand[XG_{best}. \times (-1)]$
9: **for** i = 1 to popsize **do**
10: **for** j = 1 to N **do**
11: **if** $vel(i,j) > V_{max}$ **then**
12: $zerovel(i,j) = V_{max} \times sign[zerovel(i,j)]$.
13: $onevel(i,j) = V_{max} \times sign[onevel(i,j)]$.
14: **end if**
15: **end for**
16: **end for**
17: **for** i = 1 to popsize **do**
18: **for** j = 1 to N **do**
19: **if** $X(i,j) == 1$ **then**
20: $vel(i,j) = zerovel(i,j)$.
21: **else**
22: $vel(i,j) = onevel(i,j)$.
23: **end if**
24: **end for**
25: **end for**
26: V = logsig(vel)
27: **for** i = 1 to popsize **do**
28: **for** j = 1 to N **do**
29: **if** $V(i,j) >$ rand() **then**
30: $X_{ij} = \sim X_{ij}$.
31: **else**
32: $X_{ij} = X_{ij}$.
33: **end if**
34: **end for**
35: **end for**
36: Repeat steps 2 to 35 until the termination condition is satisfied.

This process stops if the termination condition is achieved. The algorithm terminates by allocating n services and successfully executes allocated services to the various VMs.

C. BAT-based Service Allocation

BAT algorithm has its origins from the modeling of echolocation behavior in bats. Echolocation is used to estimate

the distance of the prey or obstacle (e.g. wall), as they fly towards the prey or obstacle with a random velocity, position, frequency, loudness, and pulse rate [29]. During the flight, bats can adjust their velocity, position, frequency, loudness, and pulse rate accordingly. This behavior of bats is explained in Algorithm 3. The encoding scheme is the same as PSO. Initially, all parameters such as position vector, velocity vector, frequency, loudness, and pulse rate of each bat, are initialized randomly. Then, the fitness value is calculated for each bat, and Xbest is set as the minimum fitness value. Next, this update parameters using Formulas 5 to 7. How to generate local and random solution is explained in the algorithm. The overall process continues until the termination condition is satisfied.

Algorithm 3 : BAT_SAA

Input: ETC matrix, processing Speed of VMs.

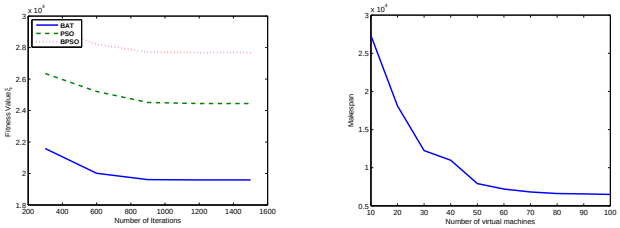
Output: Allocation result of services to VMs, Makespan, Energy, ξ.

1: Initialize random bat population : position vector or service allocation vector (X) and velocity vector(v), frequency(f), loudness(L), pulse rate(r) for each bat.
2: Calculate Fitness value (ξ) using equation (6).
3: X_{best} = bat having minimum fitness value.
4: Update the parameters as follows:
5: $f = f_{min} + (f_{max} - f_{min})\beta$
6: $v_i^t = v_i^{t-1} + (x_i^{t-1} - X_{best})f$
7: $x_i^t = x_i^{t-1} + v_i^t$
8: **Generate local solution**
9: **if** rand(0,1) $> r_i$ **then**
10: $X_i = X_{best} + w \times rand$
11: **end if**
12: **Generate random solution**
13: **if** rand(0,1) $< L_i$ and $f_i^t > f_i^{t-1}$ **then**
14: $X_i = x_i^t$
15: **end if**
16: Repeat steps 4 to 15 for each i^{th} bat.
17: Repeat steps 2 to 16 until termination condition is satisfied.

In order to evaluate the complexity of the Algorithm-1, this paper considered the time complexity of a random number generation function as $O(1)$. The input to the algorithms has n service requests (tasks) and m VMs. The time complexity of Algorithm-1 is $O(m \times n)$, because the step-2 and the loop of the algorithm take $O(m \times n)$. Since, the step-2 uses ETC matrix, it take $O(m \times n)$ time. The loop from step-9 to step-16 of Algorithm-2 take $O(nm^2)$, because of the $popsize \leq m$, j values vary up to n, and $O(m)$ time required to find out the maximum velocity value. So, the time complexity of Algorithm-2 is $O(n \times m^2)$. The time complexity of Algorithm-3 is $O(m \times n)$, because of the step-2 of Algorithm-3 take $O(m \times n)$ time due to the use of ETC matrix.

VII. Experiment and Evaluations

The experiment is carried out using an in-house simulator and MATLAB R2014a on an Intel (R) Core (TM) i7-4770 CPU @ 3.40 GHz 3.40 GHz CPU and 4 GB RAM running on Linux (Ubuntu 16.04.3 LTS). We remark that MATLAB simulator has been widely adopted to evaluate schemes proposed in the literature [2], [11], [30]. This paper consider two different experiment scenarios to evaluate the behavior of the three metaheuristic algorithms explained in the preceding section in the fog environment. There are two sets of VMs to represent a heterogeneous system environment. One set of VMs with 1000 MIPS for odd-numbered VM identification (VMID), and another set with 1500 MIPS for even-numbered VMID. If there are 10 VMs in total, then the VMID of the VMs is 1, 2, 3, . . . , 10. Services or tasks are also heterogeneous with respect to task length and resource requirement of the tasks. Two different Expected Time to Compute (ETC) matrices were used as input and one ETC matrix for scenario-1 and another for scenario-2 in the simulation.



(a) (b)

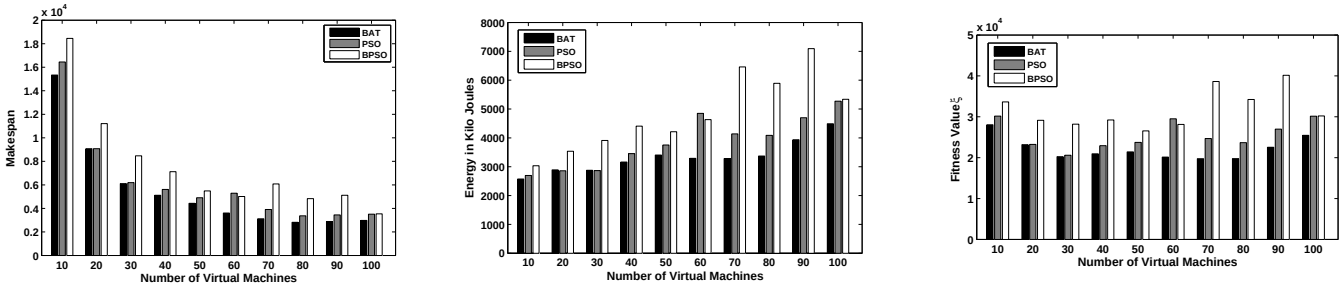
Fig. 5: (a) Fitness value ξ vs number of iterations, (b) Random Allocation Graph between number of VMs and makespan

To set the number of iterations of all mentioned algorithms, this paper computed the fitness value ξ for different number of iterations as shown in Fig. 5(a). It is observed from Fig. 5(a) that in between 800 and 1000 iterations, the steady state begins and after 1000 iterations, there is no change in the fitness value for the three metaheuristic techniques. Therefore, we set the number of iterations as 1000 for all algorithms. The population size was set to 10 for all algorithms. To fix the number of VMs, this checked the variations in makespan when the number of VMs varies from 10 to 100 and keeping the number of services (tasks) fixed to 500 as shown in Fig. 5(b). This paper uses the random allocation technique which gives an average makespan value for the corresponding number of VMs. From Fig. 5(b), it is observed that after 50 VMs, the makespan value is somehow steady and therefore, we kept 50 VMs fixed for the simulation. The algorithms computed makespan, energy consumption, and ξ using equations (3) to (6).

In scenario-1, the number of services or tasks is 500, and the number of VMs varies from 10 to 100 in intervals of 10. A comparative summary is shown in Fig. 6 (a), (b), (c), and it can be inferred that the optimization parameters in Y-axis for the three cases is less than in case of the BAT algorithm.

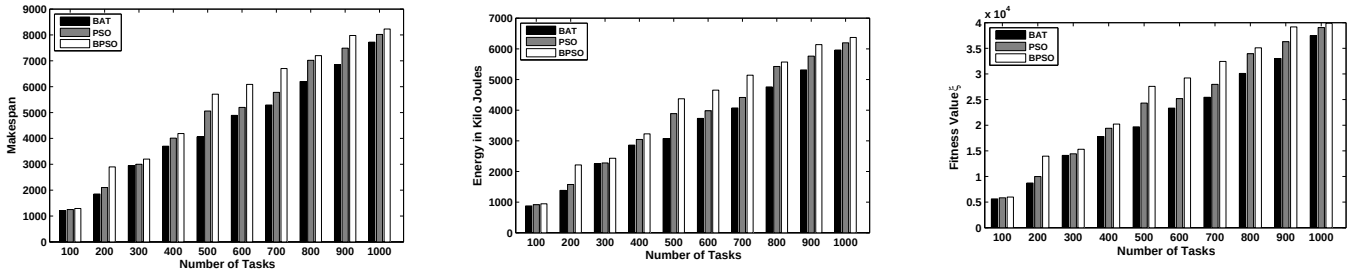
In scenario-2, the number of VMs is 50, and the number of services or tasks varies from 100 to 1000 in intervals of 100. A comparative summary is shown in Fig. 7 (a), (b), (c), and it can be observed that the optimization parameters for the three cases is less in case of the BAT algorithm.

From the above results, we conclude that among the three algorithms, BAT gives a better makespan for different scenarios. Makespan minimization results in energy conservation



(a) Makespan (b) Energy (c) Fitness Value ξ

Fig. 6: Performance of for PSO, BPSO, BAT algorithms, where the number of VMs varies with fixed services.



(a) Makespan (b) Energy (c) Fitness Value ξ

Fig. 7: Performance of for PSO, BPSO, BAT algorithms, where the number of services varies with fixed VMs.

or, the makespan of the system is directly proportional to the energy consumption of the system. The reduction in system makespan reduces the idle time of virtual machines. The energy consumption is less when the idle time of virtual machines is small. The results in Fig. 6 shows for a specific number of VMs, the system makespan proportionate to the energy consumption of the system. Similarly, the results in Fig. 7 shows for a specific number of VMs, the system makespan proportionate to the energy consumption of the system. Therefore, with the small system makespan value, the service providers can deliver services with less energy consumption.

VIII. Conclusion

This paper highlighted the service allocation problem in heterogeneous fog computing environments for the industrial applications using different nature-inspired algorithms. A system model, with distinct models for host, VM, and service (task) was proposed that takes into account the ETC model. This paper also emphasized the importance of resource heterogeneity, which is indicated in the ETC matrix. The metaheuristic techniques help to achieve service allocation energy efficiency as well as achieving the desired QoS. Since the allocation problem in fog server system does not have polynomial time algorithms and nature-inspired algorithms provide solutions in a reasonable time, this paper proposed and implemented the PSO, BPSO, and BAT algorithms for the service allocation problem. Findings from the experiment results for industrial applications (with scenarios such as variation in number of tasks and VMs) demonstrated that BAT-based service allocation algorithm outperforms the other two algorithms.

References

[1] D. Puthal, X. Wu, S. Nepal, R. Ranjan, and J. Chen, "SEEN: A Selective Encryption Method to Ensure Confidentiality for Big Sensing Data Streams." *IEEE Transactions on Big Data* (In Press), 2017.

[2] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, 3(6), pp. 1171-1181, 2016.

[3] J. Kang, R. Yu, X. Huang, and Y. Zhang, "Privacy-Preserved Pseudonym Scheme for Fog Computing Supported Internet of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[4] G. S. Aujla, N. Kumar, A. Y. Zomaya, and R. Rajan, "Optimal Decision Making for Big Data Processing at Edge-Cloud Environment: An SDN Perspective," *IEEE Transactions on Industrial Informatics*, 2017.

[5] A. Bouchentouf, "Commodities for dummies," *John Wiley & Sons*, 2011.

[6] Y. M. Al-Saleh, G. Vidican, L. Natarajan, and V. V. Theeyattuparampil, "Carbon capture, utilisation and storage scenarios for the Gulf Cooperation Council region: A Delphi-based foresight study," *Futures*, 44(1), pp. 105-115, 2012.

[7] M. D. Dikaiakos, D. Katsaros, P. Mehre, G. Pallis, A. Vakali, "Cloud computing: Distributed internet computing for it and scientific research," *Internet Computing, IEEE*, 13(5), pp. 10-13, 2009.

[8] S. K. Mishra, D. Puthal, B. Sahoo, P. Jayaraman, S. Jun, A. Y. Zomaya, R. Ranjan. "Energy-Efficient VM-Placement in Cloud Data Center." *Sustainable Computing: Informatics and Systems (SUSCOM)*, (Accepted), 2018.

[9] S. K. Mishra, D. Puthal, B. Sahoo, S. K. Jena, and M. S. Obaidat, An adaptive task allocation technique for green cloud computing. The Journal of Supercomputing, pp. 1-16, 2017.

[10] L. Guo, S. Zhao, S. Shen, C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," *Journal of Networks*, 7(3), pp. 547-553, 2012.

[11] D. A. Chekired, and L. Khoukhi, "Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling," *IEEE Transactions on Industrial Informatics*, 2017.

[12] T. Kaur, I. Chana, "Energy efficiency techniques in cloud computing: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, 48(2), pp. 22, 2015.

[13] L. Zuo, L. Shu, S. Dong, C. Zhu, and T. Hara, A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing, *IEEE Access*, 3, pp. 2687-2699, 2015.

[14] L. Zuo, L. Shu, S. Dong, C. Zhu, and G. Han, A Multi-queue Interlacing Peak Scheduling Method Based on Tasks Classification in Cloud Computing, *IEEE Systems Journal*, 2016.

[15] NRDC. https://www.nrdc.org/experts/pierre-delforge/new-study-americas-data-centers-consuming-and-wasting-growing-amounts-energy. 2013.

[16] V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, W. Ramirez, and S. Sanchez, "Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios," *In Global Communications Conference (GLOBECOM), IEEE*, pp. 1-6, 2016.

[17] N. Bergmann, Y. Y. Chung, X. Yang, "Using swarm intelligence to optimize the energy consumption for distributed systems', *Modern Applied Science*, 7(6), pp. 59-66, 2013.

[18] C. W. Tsai, J. J. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *Systems Journal, IEEE*, 8(1), pp. 279-291, 2014.

[19] Z. H. Zhan, X. F. Liu, Y. J. Gong, J. Zhang, H. S. H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys (CSUR)*, 47(4), pp. 63, 2015.

[20] INFRARATI. https://infrarati.wordpress.com/2014/03/25/data-center-co2-emissions/. 25 March 2014.

[21] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of parallel and distributed computing, 59(2), pp. 107-131, 1999.

[22] A. M. Sampaio, J. G. Barbosa, R. Prodan, "Piasa: A power and interference aware resource management strategy for heterogeneous work-loads in cloud data centers," *Simulation Modelling Practice and Theory*, 57, pp. 142-160, 2015.

[23] L. Zuo, L. Shu, S. Dong, C. Zhu, and Z. Zhou, Dynamic Weighted Load Evaluation Model Based on Self-adaptive Threshold in Cloud Computing, *Mobile Networks and Applications*, 22(1), pp. 4-18, 2017.

[24] T. Shi, M. Yang, X. Li, Q. Lei, and Y. Jiang, "An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds," *Pervasive and Mobile Computing, Elsevier*, Vol-27, pp. 90-105, 2016.

[25] M. R. Garey, D. S. Johnson, "Computers and intractability: a guide to the theory of np-completeness," *WH Freemann, New York, San Francisco*, 1979.

[26] D. Klusacek, H. Rudova, "A metaheuristic for optimizing the performance and the fairness in job scheduling systems," *Artificial Intelligence Applications in Information and Communication Technologies, Springer*, pp. 3-29, 2015.

[27] X. Dai, J. M. Wang, B. Bensaou, "Energy-efficient virtual machines scheduling in multi-tenant data centers," pp. 210-221, 2016.

[28] Z. Wang, K. Shuang, L. Yang, and F. Yang, "Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter," *Journal of Convergence Information Technology*, 7(1), pp. 62-70, 2012.

[29] S. Raghavan, P. Sarwesh, C. Marimuthu, and K. Chandrasekaran, "Bat algorithm for scheduling workflow applications in cloud," *In IEEE International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, pp. 139-144, 2015.

[30] T. S. Somasundaram, K. Govindarajan,"CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud," *Future Generation Computer Systems*, 34, pp. 47-65, 2014.

Sambit Kumar Mishra is pursuing a Ph.D. in the Department of Computer Science & Engineering at National Institute of Technology, Rourkela, India. His area of research is Cloud Computing, Parallel and Distributed computing System, Wireless Sensor Networks. He obtained his M.Tech. and M.Sc. in Computer Science from Utkal University, India. He is a member of IEEE computer society.

Deepak Puthal (M17) earned his Ph.D. degree in computer science and information systems from the University of Technology Sydney (UTS). He is a lecturer (assistant professor) in the Faculty of Engineering and Information Technology at UTS. His research interest includes Cyber Security, Internet of Things, Edge/Fog Computing. He received the IEEE Distinguished Doctoral Dissertation Award for Excellence in Special Technical Community on Smart Computing for the year 2017. He is serving as associate editor in IEEE Consumer Electronics Magazine, Internet Technology Letters (Wiley), KSII Transactions on Internet and Information Systems (TIIS).

Joel J. P. C. Rodrigues [S01, M06, SM06] is a professor and senior researcher at the National Institute of Telecommunications (Inatel), Brazil and senior researcher at the Instituto de Telecomunicaes, Portugal. He has been professor at the University of Beira Interior (UBI), Portugal and visiting professor at the University of Fortaleza (UNIFOR), Brazil. He received the Academic Title of Aggregated Professor in informatics engineering from UBI, the Habilitation in computer science and engineering from the University of Haute Alsace, France, a PhD degree in informatics engineering and an MSc degree from the UBI, and a five-year BSc degree (licentiate) in informatics engineering from the University of Coimbra, Portugal. Prof. Rodrigues is the leader of the Internet of Things research group (CNPq), Member of the IEEE ComSoc Board of Governors as Director for Conference Development, IEEE ComSoc Distinguished Lecturer, the President of the scientific council at ParkUrbis Covilh Science and Technology Park. He is the editor-in-chief of three International Journals and editorial board member of several high-reputed journals. He has been general chair and TPC Chair of many international conferences, including IEEE ICC, IEEE GLOBECOM, and IEEE HEALTHCOM. He has authored or coauthored over 550 papers in refereed international journals and conferences, 3 books, and 2 patents. He had been awarded several Outstanding Leadership and Outstanding Service Awards by IEEE Communications Society and several best papers awards. Prof. Rodrigues is a licensed professional engineer (as senior member), member of the Internet Society, and a senior member ACM and IEEE.

Bibhudatta Sahoo obtained his M. Tech. and Ph.D. degree in Computer Science & Engineering from NIT, Rourkela. He is presently Assistant Professor in the Department of Computer Science & Engineering, NIT Rourkela, INDIA. His technical interests include Data Structures & Algorithm Design, Parallel & Distributed Systems, Networks, Computational Machines, Algorithms for VLSI Design, Performance evaluation methods and modeling techniques Distributed computing system, Networking algorithms, and Web engineering. He is a member of IEEE & ACM.

Eryk Dutkiewicz received the B.E. degree in electrical and electronic engineering and the M.Sc. degree in applied mathematics from the University of Adelaide in 1988 and 1992, respectively, and the Ph.D. degree in telecommunications from the University of Wollongong in 1996. He is currently the Head of the School of Electrical and Data Engineering, University of Technology Sydney, Australia. He has held visiting professorial appointments at several institutions including the Chinese Academy of Sciences, Shanghai Jiao Tong University, and Macquarie University. His current research interests cover 5G networks and medical body area networks.