sustainability

Article

Cloud–Fog–Based Smart Grid Model for Efficient Resource Management

Saman Zahoor [1], Sakeena Javaid [1], Nadeem Javaid [1], Mahmood Ashraf [2], Farruh Ishmanov [3,*] and Muhammad Khalil Afzal [4]

1 Department of Computer Science, COMSATS University, Islamabad 44000, Pakistan; samanzahoor@gmail.com (S.Z.); sakeenajavaid@gmail.com (S.J.); nadeemjavaidqau@gmail.com (N.J.)
2 Department of Computer Science, Federal Urdu University of Arts, Science and Technology, Islamabad 44000, Pakistan; mahmood@fuuastisb.edu.pk
3 Department of Electronics and Communication Engineering, Kwangwoon University, Seoul 01897, Korea
4 Department of Computer Science, COMSATS Institute of Information Technology, Wah Cantonment 47040, Pakistan; khalilafzal@ciitwah.edu.pk
* Correspondence: farruh.uzb@gmail.com

Received: 17 April 2018; Accepted: 7 June 2018; Published: 19 June 2018

Abstract: A smart grid (SG) is a modernized electric grid that enhances the reliability, efficiency, sustainability, and economics of electricity services. Moreover, it plays a vital role in modern energy infrastructure. The core challenge faced by SGs is how to efficiently utilize different kinds of front-end smart devices, such as smart meters and power assets, and in what manner to process the enormous volume of data received from these devices. Furthermore, cloud and fog computing provide on-demand resources for computation, which is a good solution to overcome SG hurdles. Fog-based cloud computing has numerous good characteristics, such as cost-saving, energy-saving, scalability, flexibility, and agility. Resource management is one of the big issues in SGs. In this paper, we propose a cloud–fog–based model for resource management in SGs. The key idea of the proposed work is to determine a hierarchical structure of cloud–fog computing to provide different types of computing services for SG resource management. Regarding the performance enhancement of cloud computing, different load balancing techniques are used. For load balancing between an SG user's requests and service providers, five algorithms are implemented: round robin, throttled, artificial bee colony (ABC), ant colony optimization (ACO), and particle swarm optimization. Moreover, we propose a hybrid approach of ACO and ABC known as hybrid artificial bee ant colony optimization (HABACO). Simulation results show that our proposed technique HABACO outperformed the other techniques.

Keywords: cloud computing; smart grid; fog; resource management; smart devices; load balancing

1. Introduction

The emergence of the internet of things (IoT) raises the concept of smart connected communities. These communities have smart transportation systems, smart homes, smart learning, smart health care services, and smart grids (SGs). All the components are tied to each other via an Internet connection. The SG is one of the important components in a smart connected community. An SG is an intelligent scattered infrastructure that controls energy requirements in a supportable and economic way with the facility of reliable communication systems for controlling and monitoring which is described by Ghasemkhani and Signorini et al. in [1,2]. Whereas, Blanco-Novoa et al. in [3] discuss that the merging of SGs and the IoT is called the internet of energy (IoE), which can act as an expansion of the SG. The objective of the IoE is to give an efficient framework for energy trading between consumers.

Scattered intermittent energy generation and storage need to be controlled and observed logically via the Internet.

To tackle growing complications and the huge volume of data produced by the immense usage of devices (i.e., sensors, smart meters, and actuators), robust processing resources are required, which must be processed, accessed, stored, and managed by cloud computing (CC). Moreover, the grouping of CC with IoT for the formation of an IoE platform can be considered as pervasive sensing facilities by Al Faruque et al. in [4]. CC allows the sensing information to be stored and utilized coherently for smart observation and strong handling of the detected information streams. However, the response time and latency in CC are increased by increasing the number of smart devices, which causes deviations for some delay-sensitive applications and smart devices.

The fog computing concept is recommended in [5] by Aazam et al. to overcome the above-mentioned challenges. Fog or edge computing extends the CC at the corner or edge of the network. Fog computing allows the information to be preprocessed, where a latency constraint is needed. The stated characteristics of fog computing are most beneficial such as: location awareness, minimum latency, geographical distribution, massive number of devices, mobility, real-time applications, and heterogeneity as discussed by Bonomi et al. in [6]. Some challenges arise when using smart devices, system copes some challenges, such as latency requirements, resource-constrained devices, network bandwidth, and cyber-physical systems. When smart devices are connected to the Internet, new security challenges are raised, such as protecting resource-constrained devices and maintaining security status. Moreover, up-to-date software for all smart devices which can assessing the security status of large distributed systems in a reliable way and respond to security compromises without creating insufferable troubles are not fulfilled by CC. To overcome the challenges of CC, Chiang et al. [7] presents a fog-based architecture which dispenses computing, storage, control, and system administration nearer to the end-user devices. Therefore, this paper presents a viable architecture for SGs in light of consolidating two emerging technologies: cloud and fog computing.

1.1. Motivation

A cloud–fog–based platform is presented by Suryawanshi and Luan et al. in [8,9], where fog devices are installed inside a multi-floor shopping center and an interstate bus to provide better services to end users. The work in [10,11] by Luo and Gan in et al. have considered for efficient resource allocation regarding electricity consumers in a SG. Luo et al. in [10] have discussed a cloud computing-based infrastructure for a future generation power grid. However, the concept of fog computing has not incorporated in [10], which could improve the latency and response time for efficient resource allocation In addition, Gan et al. in [11] presents a decentralized algorithm for optimally scheduling of electric vehicle (EV) charging. The power in the EV can be provided by a collector system from off-vehicle sources: a battery, solar panels or an electric generator, etc. However, authors have not consider the cloud and fog platforms in this scheme and no on-demand services were considered for the consumers, which creates a high latency and slow response time in the system. Furthermore, Hao et al. in [12] have considered the objectives of load shuffling facility by optimally scheduling the charging and discharging behavior of EVs in a decentralized manner. However, the cloud and fog platforms have not integrated in this study to schedule the demands of EV consumers in terms of minimizing cost, response time, processing time, and request loading time. Based on these studies, we incorporate the concepts of fog and cloud computing in order to effectively schedule resources in residential buildings to enhance the response, request loading time, and processing time with minimum latency. Moreover, it can be predicted that by integrating a cloud–fog–based platform in SGs, users will be furnished with less complexity described by Mohamed et al. in [13]. However, in this platform, by increasing the number of consumers, the resource management problem is also increased. Several load balancing techniques are described in the literature by Dam and Chen et al. in [14,15] for resource management in cloud environments. A cloud–fog–based SG model for resource management is presented in this study to handle SG consumers’ (end users’) requests, and various load balancing policies are also implemented. A graphical representation of this concept is depicted in Figure 1. This figure illustrates the mechanism of information sharing between the consumers and the utility through the services of cloud–fog–based servers. According to Luan et al. [9], the physical distance is shorter than the communication distance because users access the services of the cloud via an internet protocol (IP) network. The physical distance is kept equivalent to the communication distance for fog servers due to the involvement of single-hop wireless connection. The proposed scenario considers fog servers to reduce the latency and improve the response time to consumer requests.

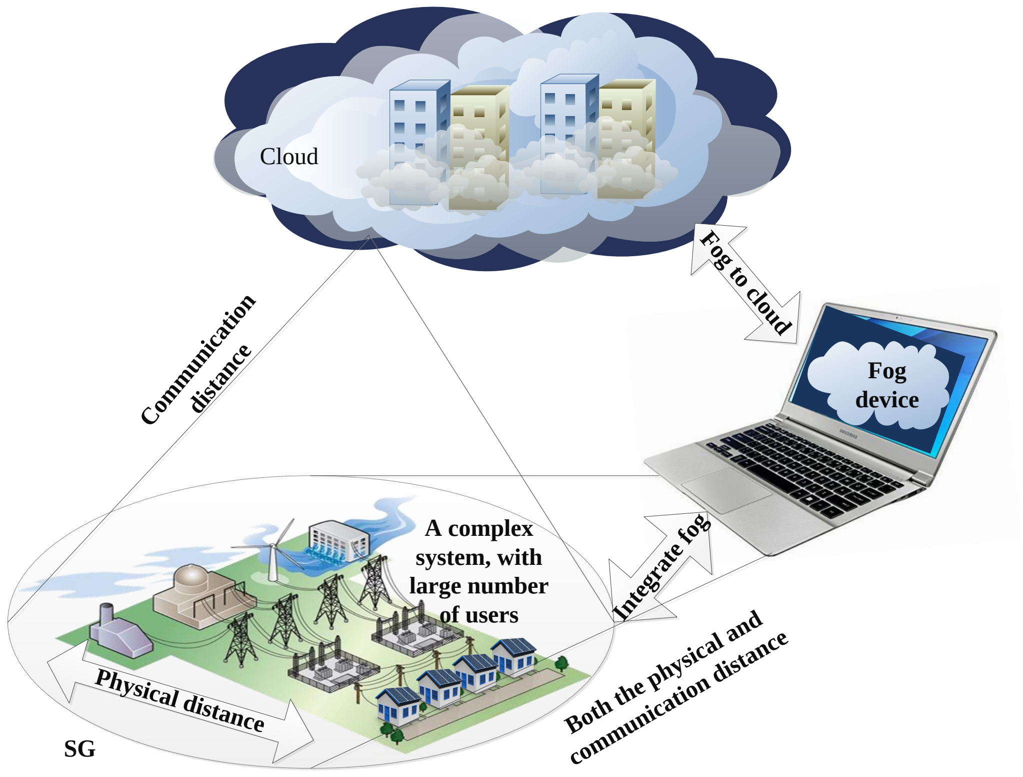
Cloud
Fog to cloud
Fog device
Communication distance
A complex system, with large number of users
Integrate fog
Physical distance
SG
Both the physical and communication distance

Figure 1. Cloud, fog, and smart grid (SG).

1.2. Contributions

To get benefits from the SG, a cloud–fog–based model is integrated (an extension of Saman et al. [16]) for efficient resource management in smart buildings. The main contributions of our work are:

- A new cloud–fog–based model is presented to optimally allocate the consumers' requests in the cloud–fog environment.
- Fog devices are integrated with the cloud environment in order to minimize the system latency because fog devices are placed nearer to the end users and can respond faster than the cloud server.
- Providing location awareness services through connected fog devices.
- A new hybrid artificial bee ant colony optimization (HABACO) algorithm is proposed to optimize the allocation of requests to the available virtual machines (VMs) in the cloud–fog–based SG model to deal with the request scheduling problem.
- In a residential area, two scenarios are considered, where renewable energy sources are used to overcome the environmental concerns.
- Extensive simulations are conducted in order to demonstrate that the cloud–fog–based SG model can fundamentally supplement the SG with minimum communication time.

- The performance of the proposed algorithm "HABACO" is evaluated and compared in order to demonstrate its effectiveness by estimating the performance parameters with other approaches: particle swarm optimization (PSO), artificial bee colony (ABC), ant colony optimization (ACO), round robin (RR), and throttled.

The rest of the paper is organized as follows: Section 2 provides an overview of related work. Section 3 introduces the system model. Section 4 presents the simulation results and discussion. Finally, conclusions are drawn in Section 5.

2. Related Works

Multiple solutions have been presented in order to cope with real-time management, energy efficiency, and to provide cost-effective solutions in power grid systems. These studies can be classified into three classes pursuant to already-defined architectures.

2.1. SG-Based Architecture

An improved home energy management algorithm is presented by Hussain in et al. in [17], which is based on the concepts of demand response and RESs (i.e., wind turbine, PV panels, hydro power systems etc.). This model includes demand response principles, photovoltaic panels, state of charge, and the sharing of multiple resources for supplying load. The energy cost and time-based discomfort are minimized; however, the model has some limitations. These limitations are optimized using a multi-objective genetic algorithm with Pareto optimization. After applying these schemes, consumers had the most optimal results. A review of revolutionary changes occurring in advanced power systems is discussed by Yoldas et al. in [18]. This study is based on certain challenges occur due to the increasing demands of worldwide electricity, consumers' lacking cognizance of carbon emission minimization, the lack of the integration of RESs on a global scale, and the incorporation of bidirectional communication technologies. The aforementioned challenges raised the notion of establishing MGs (for small scale power panels), which are currently becoming an emerging domain due to the integration of smart grid tools and technologies. In addition, this review also discusses the MG and its functionalities, as well as the integration of the intelligent services of the smart grid environment. A critical analysis of the future smart grid services is also described, with multiple suggested solutions.

Rajarajeswari and Ashutosh et al. in [19] have presented demand-side management in an SG using a genetic algorithm optimization scheme for residential, commercial, and industrial loads. Only fixed and elastic (shiftable) appliances are considered in this system. A genetic algorithm is used to reschedule the shiftable load in order to minimize the cost and peak to average ratio. After rescheduling the schedulable appliances, the cost and peak to average ratio of this system were reduced. In [20], Barbato et al. present a fully distributed strategy for demand-side management using SG infrastructure in order to minimize the peak formation ration for the consumers. This strategy uses dynamic pricing schemes and considers two types of scheduling strategies: a fully distributed strategy and a hybrid strategy. Every device makes an autonomous scheduling decision in the fully distributed strategy, whereas in the hybrid strategy, the consumer schedules the devices according to the demand. The authors discuss the performance of these strategies in terms of their numerical evaluation by considering them in the grid-connected modes. The system is modeled by a non-cooperative game theory following certain system constraints. After the numerical evaluation of the system, it was shown that 55% of the peak formation was minimized. In these papers, different numbers of appliances with distinct usage patterns are scheduled to meet the objectives of cost, peak to average ratio, and user comfort. Moreover, demand response management has turned out to be one of the growing technologies for SGs. However, with increasing consumer involvement and to handle large number of data, it become computationally tough. To tackle the issues of high computation, scalability, and emergencies, there is a need for a reliable and efficient system which meets the requirements efficiently.

2.2. Cloud- and Fog-Based Architectures

The CC paradigm has attracted a great deal of attention in the literature (e.g., Mora et al. in [21]), and they have suggested that CC applications are strongly dependent on their architectural frameworks. Authors have explained the concepts and distinctive points of conventional and cloud computing in terms of their technical and non-technical challenges, also discussing multiple future directions [22] by Armbrust et al. In [23], Xia et al. discuss an approach which supports the content-based image retrieval for encrypted images without exposing them to the cloud server. Firstly, corresponding images are extracted from the feature vectors. Secondly, they develop the pre-filter tables using a locality-sensitive hashing in order to enhance the surfing efficiency. Furthermore, a protection mechanism is applied by the kNN (k nearest neighbor) algorithm to secure the feature vectors and an encryption mechanism is applied using the standard stream cipher to secure the images. They also proposed a watermark-based protocol in order to avoid illegitimate spreading of the images, and this watermark is added by the cloud server before transmitting to the users. In any case, if an illegitimate copy of the image is obtained, then the user is tracked through the watermark extraction process. This approach helps in the protection and efficiency of the cloud resources.

Authors developed a consumer interest model in which a surf-able encryption scheme is developed in CC using a multi-keyword ranked search. The purpose of this scheme was to add privacy in the CC environment described by Fu et al. in [24]. Xia et al. in [25] describe the similar technique “multi-keyword ranked search” for CC dynamic operations’ updates. The technique is used for the encrypted data in CC, which assists in deletion and insertion of the documents and updates them dynamically. A vector-based model and TF $\times$ DF models are collectively used for index creation and query transmission. Tree-based index organization and a greedy depth-first search algorithm are used to enhance the surfing efficiency. Encryption of the indexes and generated query vectors is secured using the kNN algorithm, and these algorithms show the precise score computation for both indexes and query-generated vectors. This scheme achieved sub-linear surfing time for the deletion and insertion of documents, which enhanced the efficiency of the proposed scheme.

An alternative paradigm, fog computing, is proposed by Bonomi et al. in [6] to solve the problems of latency, delay, response time, and requests per hour. Fog computing provides us with very good solutions in terms of delay, reliability, processing time, etc. However, in both cloud and fog emerging technologies, consumers send requests randomly from any processor. Congestion and overburden problems are raised due to the large number of end users’ requests. However, the overburden of servers is linked with the unfair assignment of the tasks. Random assignment of tasks causes load imbalance, where some processors are overloaded and some are under-loaded. Khiyaita et al. in [26], a review of load balancing techniques in CC is presented by exploiting the significant research obstacles in future SGs. The main objective of load balancing is to transfer load explicitly from overloaded processes to under-loaded processes. In order to meet the consumers’ requests, the system must enable the coordination. It can create congestion and can also be directed to create imbalance in service management. To effectively balance the services, CC needs load balancing algorithms in the distributed systems with little modification. To overcome this issue, many researchers are working to resolve the problem. For example, Sambit et al. in [27] present a solution to resolve the load balancing problem. In this work, different types of loads in the cloud environment are considered: memory, computation, and network loads. To effectively balance them in the CC environment, different heuristic algorithms are considered: genetic algorithm, simulated annealing, tabu search, etc.

A hybrid ACHBDF (ant colony, honey bee with dynamic feedback) algorithm is also discussed in [28] by Nikhit et al. as a load balancing mechanism for optimized resource utilization in CC. This algorithm utilizes the collective scheme of the two runtime scheduling schemes using the dynamic time step feedback methodology. It also relies on the quality of ACO and honeybee algorithms for effective task scheduling. To maintain the dynamic feedback method, its feedback methodology supports the system to verify the load after each iteration. Bitam et al. in [29] propose a new bio-inspired algorithm—bees life algorithm (BLA)—for effective task assignment in the fog computing environment. This optimization approach is based on the equivalent distribution of fog nodes. The main purpose of this study is to obtain the tradeoff between the CPU execution time and the storage utilized by the fog nodes. The response time and cost for this system is also evaluated and compared with the previous PSO and genetic algorithms. In this case, it outperformed both of the previous algorithms.

2.3. SG with Cloud-Based Architecture

Many scholars have deliberated on the use of cloud–fog computing to support and manage SGs. Mohamed et al. in [13] have described a technique called "service-oriented middleware (SOM)" to effectively anatomize the obstacles during the development and operation of smart cities' functionalities through the cloud of things and fog computing. SOM is also referred to as SmartCityWare, which comprises of cloud and fog computing features. The service-oriented model (used as a middleware) provides services to the multiple functionalities and parameters of the smart city applications abstracted by the SmartCityWare. This improves the services' functionalities and their parameters demanded by the customers in the smart city. However, they do not consider the resource management problem of servers. Additionally, SG data management based model on a CC is presented in [30] by Reka et al. which takes advantage of distributed data management for real-time data gathering, ubiquitous access, and parallel processing for real-time information retrieval. Stochastic programming is embedded in CC for the effective load management of the SG users. The simulation results are obtained through a GUI (designed interface) and Gurobi optimizer in Matlab. The objective of this scheme is to minimize the energy requirements by adding smart energy hubs.

Moreover, for a faster response time in large-scale deployments, cloud-based demand response architecture is presented in [31] by Moghaddam et al. The studied demand responses are of two types: cloud-based demand response and distributed demand response. These are optimized by two models, including a demand response and a communication model. The objective of this study was to minimize the convergence time and efficient bandwidth utilization. In the end, these models showed the cost efficiency of the proposed system and verified that obtaining more consumer requests can increase the communication cost. Chekired and Lyes in [32] have proposed an efficient SG electric vehicle charging and discharging service at public supply stations based on cloud environment scheduling. It ensures the communication links between SG and cloud platforms. This system also considers the waiting time by using priority assignment scheduling algorithms. Two algorithms are used for the EV scheduling process: calendar priority optimization and random priority optimization algorithms. Four types of priorities are included for each EV user. However, this study only considers the problem of EVs and ignores the request management of homes or buildings. Moreover, a green scheduling for the cloud Datacenter is handled by Gu et al. in [33]. The focus of this work is energy trading with the power grid. The objectives are energy cost and carbon emission minimization, where renewable energy sources are used to cope with hazardous emission.

Overview of above mentioned studies demonstrate that there is need of a highly reliable cloud–fog–based SG platform for efficient resource management that handles the consumer and utility activities. Therefore, we consider a scenario where a number of smart home and smart building requests are handled by some servers. These servers optimize them in order to minimize the processing time, response time, and cost of the resources.

3. System Model

Cloud service providers generally have several DCs (reserved for computation and storage) in geographically scattered locations. The proposed system model of a cloud–fog environment contains multiple fog DCs and a cloud DC. The geo-distributed cloud–fog environment-based SG framework is depicted in Figure 2. The proposed model is comprised of three layers: end user layer, fog layer, and core cloud layer.

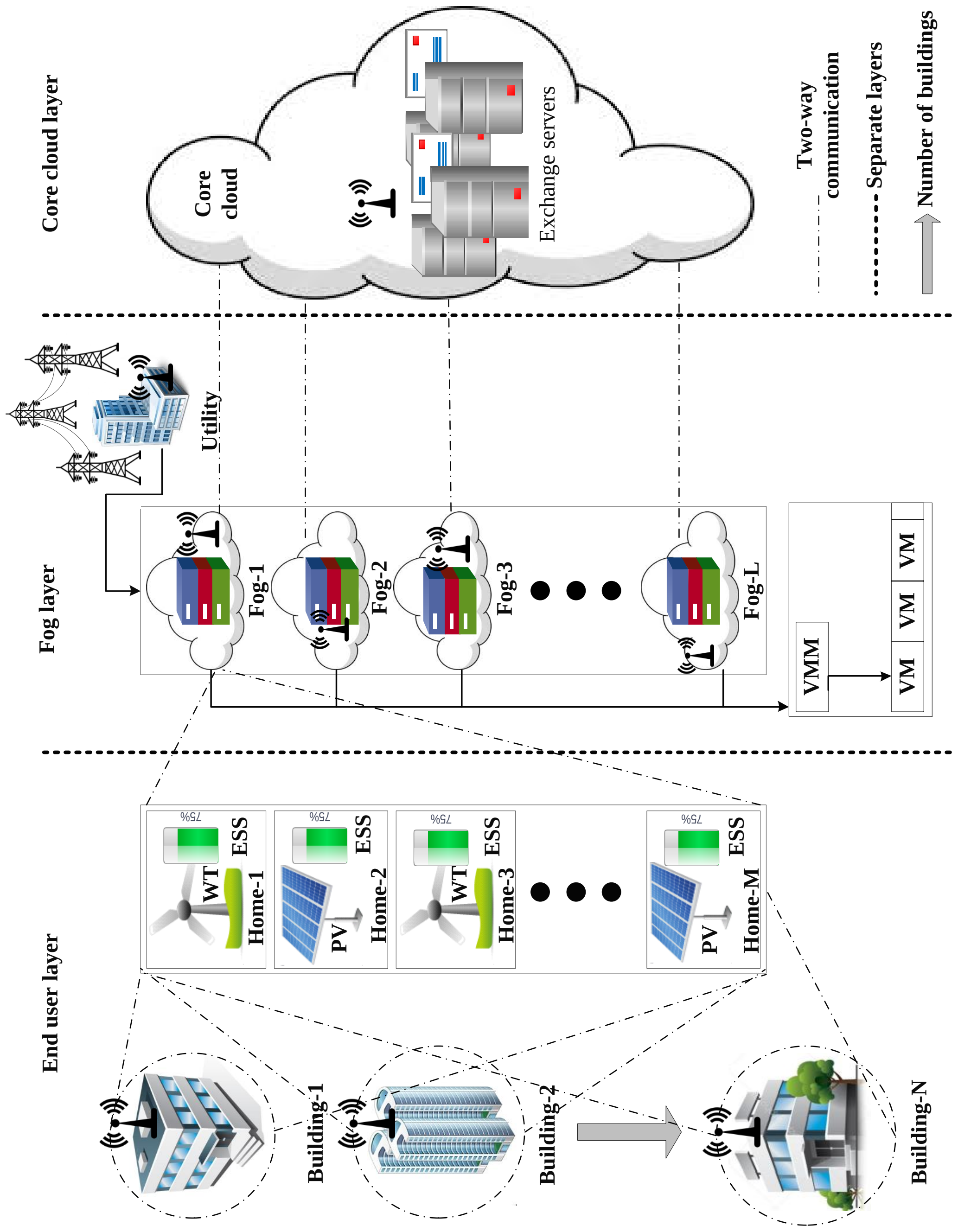

Figure 2. System model of cloud–fog–based SG. ESS: energy storage system; PV: photovoltaic; VM: virtual machine; VMM: VM monitor; WT: wind turbine.

We assume that the end user layer consists of N number of buildings $B = \{b_1, b_2, b_3, ..., b_N\}$, and each building has multiple homes $H = \{h_1, h_2, h_3, ..., h_M\}$. Every home has a renewable energy generation unit and an energy storage system (ESS) to fulfill the electricity demand. This type of generator has no emission or fuel cost and they are environmentally friendly due to the extraction of energy from natural sources. Further, the excessive generated energy is stored in the ESS to fulfill the load demand of the home in low generation hours. All information about a home energy consumption, energy generation, and scheduling of appliances is sent to the fog layer. This layer accesses numerous cloud resources to run their applications. The defined smart buildings or homes communicate with the fog devices via smart meters. All of the homes share their deficit and excessive power information with each other through the cloud–fog environment. The smart meters interact via local area network, wide area network, or metropolitan area network. There are numerous wireless solutions for communication link in the SG, such as Wi-Fi, Z-Wave, or ZigBee.

The second layer is the fog layer, which is used to effectively manage the latency issue and network resource management. The fog layer physically exists in the consumers' local region (i.e., in region 1, region 2, etc.), which is nearer to the consumers. In short, the fog node is situated closer to the consumer (i.e., one hop away from the consumer), as shown in Figure 1 where physical and communication distances are equivalent. These fogs are managed by the internet service providers as in [5,7]. The fog layer consists of $F = \{f_1, f_2, f_3, ..., f_L\}$ number of fogs. Here, each smart building is connected with a fog device. The fog devices consist of hardware (H/W) resources (i.e., main memory, storage, network bandwidth, and processor) which are virtualized. In a single physical machine, different numbers of VMs are working according to the virtualization concept which are managed by a virtual machine monitor (VMM). The VMM sustains various operating systems (OSs) to run applications on a single H/W platform simultaneously. The hypervisor (VMWare, Xen, UML, etc.) or VMM operates as an interface between the VMs and the guest OSs. A large number of heterogeneous applications are running on each VM or guest OS, which is the basic unit to execute an application or a request. Let $VM = \{vm_1, vm_2, vm_3, ..., vm_n\}$ be the set of VMs utilized in the fog devices. The fog layer is used for communication and works as an intermediate layer between the end user and the cloud. The last layer is the core cloud layer. The main components in this layer are the DCs which are used to facilitate the demanded storage and computing capability to the end users. They work on a pay-as-you-go basis as per the requirements of the applications.

The most important feature of CC is the computational load profile characteristics of the computing applications. When a large number of applications run on a single platform, it overburdens the server. To tackle this issue, various techniques are used. This concept is easily understood from Figure 3, where end users generate a large number of requests to access the service provider. Moreover, for efficient load balancing or resource utilization in VMs, a load balancer is used. For efficient computational load profile management in CC, different load balancing techniques are used. In SGs, the computational load profile is similar to the electricity load profile concept. So, when we integrate the SG with the cloud–fog–based environment, then efficient management of the computational load profile of all SG-related tasks is also necessary as well. In this work, five heuristic algorithms are implemented to solve the load balancing problem. Moreover, when this system model works for all regions of the world, every region has multiple numbers of buildings and fogs. These buildings may be in residential, commercial, or industrial areas. For performance evaluation, two scenarios are considered in this system model, which will be discussed in detail in the simulation section.

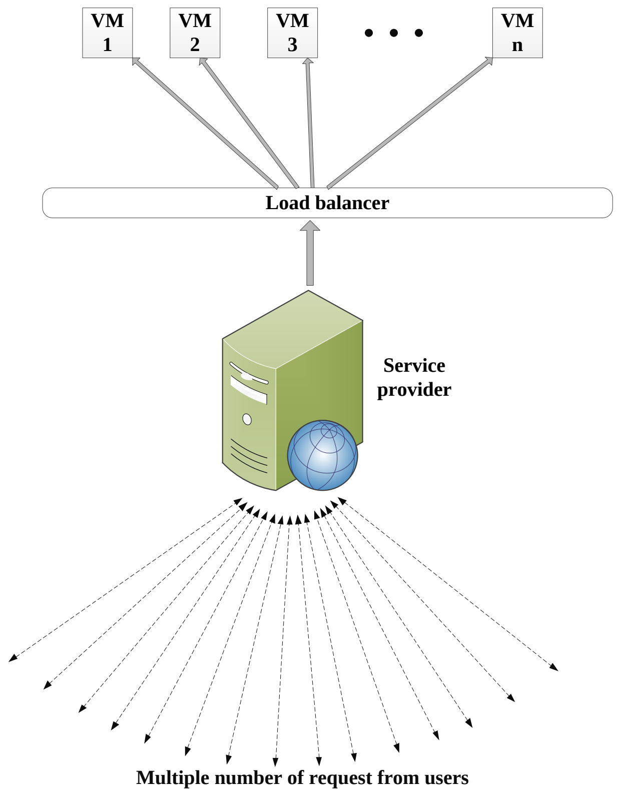

Figure 3. System model of load balancing considering fixed number of requests to cloud and fog environment.

However, some explanation of the algorithms used for efficient load balancing is provided here. When an algorithm works on equal time slicing, it is known as RR. Here, resources are allocated to each host by an equal time slicing mechanism for their efficient utilization. This algorithm is used to balance the load of requests coming from end users and allocate them to VMs. The throttled algorithm also maintains the resources. In this algorithm, all VMs' information is available at the start and it maintains the indexes table of VMs. However, these algorithms are sequential, and they do not work on local best or global best results. To overcome this issue, the PSO algorithm is used here. PSO gives us both the local best and global best results. In this algorithm, in every iteration, the current best (local best) value is compared with the previous best (global best) value, and if it meets the fitness criteria, the current best is selected as the global best. Otherwise, the previous one remains as the best value.

3.1. Problem Formulation

Every system has some performance parameters for stability improvement. Here, for the cloud–fog–based SG framework, load balancing over the available virtualized resources is done in order to improve the system stability. The performance parameters of this system are processing time, response time, and costs (VM cost, data transfer cost (DTC), and total cost (TC)).

As we know, there are N buildings, and each building has a number of homes M. Let VM = $\{v_1, v_2, v_3, ..., v_l\}$ be the set "l" of requests (tasks) which can be sent to L number of fogs, and each fog has a VMM. The VMM has complete information of active VMs, task queue length of the hosts, and availability of resources in different hosts. It will further assign these tasks to VMs. Every

VM uses its own resources (running in parallel). A VM does not share its resources with other VMs. To find the total number of tasks T_V for N buildings:

$$T_V = \sum_{i=1}^{N}(V_i). \tag{1}$$

Further, mapping of these T_V to "n" VMs affects numerous performance parameters [34]: processing time, response time, and different types of costs. The mathematical modeling of these performance parameters are written as outlined hereafter.

3.1.1. Processing Time

To calculate total processing time (y), first, collect all information of the tasks and VMs. Then, processing time $P_{i,j}$ of allocating task i to VM j is calculated by Equation (4) and λ defines the status of the tasks.

$$\lambda_{i,j} = \begin{cases} If\ task\ \text{`i'}\ is\ assigned, & 1, \\ Otherwise, & 0. \end{cases} \tag{2}$$

Objective is:

$$y = \sum_{i=1}^{T_V}\sum_{j=1}^{N}(P_{i,j} \times \lambda_{i,j}), \tag{3}$$

where

$$P_{i,j} = \frac{Length\ of\ i^{th}\ task}{Capacity\ of\ j^{th}\ VM} \times P_e. \tag{4}$$

3.1.2. Response Time

The response time R_T is the time T taken by fog DC to receive the tasks from the building (end user). $Finish_{Time}$ is the finishing time of the task. Arrival time is denoted by $Arrival_{Time}$. Moreover, delay time $Delay_{Time}$ is the time after reaching the request into DC. P_e is the earliest possible response time of the system.

$$R_T = Delay_{Time} + Finish_{Time} - Arrival_{Time}. \tag{5}$$

3.1.3. Costs

Each and every system must pay some cost according to the resource usage. As mentioned above, we calculate different types of costs, which are stated as below.

VM cost is calculated by:

$$Cost_{VM} = \frac{T_{Total}}{Cost_{VM}\ per\ hour \times U}, \tag{6}$$

where T_{Total} is the end time minus the start time of the VM and "U" represents the value of converting time from the milliseconds (ms) to hours.

DTC is calculated as:

$$Cost_{DTC} = Data\ in\ GB \times Data\ cost\ per\ GB. \tag{7}$$

DTC cost is the amount of data sent to the service provider in GBs. When we multiply the data sent in GBs with the per-GB cost, the cost of DTC is obtained.

After calculating the VM and DTC costs, total cost is calculated by Equation (8):

$$TC = Cost_{VM} + Cost_{DTC}. \tag{8}$$

3.2. Proposed Algorithm

In this section, the proposed hybrid algorithm is presented, which is the combination of ACO and ABC. ACO was proposed by Marco Dorigo in 1992, and mimics the behavior of ants. In this algorithm, ants search for food in the form of a group and connect to each other by pheromone laid down on a path by the ants. With the increase in the number of ants on a certain path, the intensity of the pheromone increases, resulting in a more reliable, accurate, and smaller path towards the food source. For independent task scheduling, the work in [35] is considered. When the number of specified tasks are equal to the number of ants, every ant starts with a random task and uses resources to process the task and then calculates the resources using the probability function as follows:

$$Prob_{ij} = \frac{(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}}{\sum(\tau_{ij})^{\alpha}.(\eta_{ij})^{\beta}}, \tag{9}$$

where τ_{ij} denotes the pheromone value related to task i and resource j, η_{ij} denotes the heuristic function, and α and β are the constant coefficients. After calculating the probability of each step, each ant builds a solution for assigning all the tasks to the resources. The pheromone value is initially set as a positive constant, then at the end of every iteration, the ants change this value. The ABC algorithm was proposed by Dervis Karaboga in 2005 [36]. It mimics the behavior of honey bees to achieve the best food source, called "nectar". For task scheduling in SGs, we assume that the number of bees are equal to the number of tasks and the number of food sources are equal to the number of VMs. Each ant starts with an arbitrary task and resource (VM) for processing this task. Furthermore, the task to be executed and the resource on which it has to be performed are calculated by the probability function Equation (10):

$$Prob_{j} = \frac{Fit_{j}}{\sum_{i=1}^{V} Fit_{i}}, \tag{10}$$

where Fit_j is the fitness of source j, Fit_i is the fitness of the task or the fitness of the requests from the users i, and V is the total number of tasks or requests. The proposed load balancing strategy depends on the best features of ABC and ACO to make the hybrid (HABACO) in order to increase the efficiency of the system for efficient resource management. Here, for load balancing, ACO is used in search of new sources of food (VM) based on best source utilization. However, it cannot change the obtained pheromone value for the VM in some iterations. So, as a result, an optimal solution is not found due to local optima. To find the best optimal solution (which task is assigned to which VM), we integrate the ABC fitness function (waggle dance) step into ACO to find the global optimum solution.

The description of smart grid scenarios is mapped to fog computing in the proposed system along with the integration of the heuristic algorithms: PSO, ABC, ACO, and our proposed hybrid HABACO. Fog is used in the local region in order to facilitate the local consumers' requests, and helps in minimizing the response time and latency of the consumers. It also helps in efficient resource allocation. Although CC provides the on-demand delivery of the resources, it increases the latency of the system. So, fog computing is integrated in this system to minimize the latency of the consumers' services (i.e., consumers' energy consumption requests), and allocation of the SGs' resources using the cloud and fog platform is performed by the existing algorithms: PSO, ABC, ACO, and the newly proposed algorithm HABACO. Since different homes (in one building (considered in our work) or in multiple buildings (taken as an example)) are different load requests and these algorithms are also stochastic in nature, handling the consumers' requests through the heuristic algorithms results in a more appropriate approach.

For load balancing, Algorithm 1 illustrates the steps to efficiently allocate the incoming tasks to the VMs. In this algorithm, every resource (i.e., VM, fogs, cloud datacenters, consumers' load requests from the SG environment, etc.) is visited once, which is based on expected processing time and response time of task i on VM j. According to these expected outcomes, we achieve the global optimal solution by using the bee fitness function.

Algorithm 1: HABACO Algorithm

```
1: Begin
2: Input: Set of user requests (tasks), number of VMs
3: Output: Processing time, response time, and costs (VMs, DTCs, and TC)
4: **Step-1: Creation of system's setup**
5: Create initial cloud and fog setup
6: **Step-2: Parameter initialization**
7: Initialize number of VMs, tasks, fogs, and broker policy for cloud datacenters
8: Initialize maxCount = maxVal, VM ID = 1
9: Set threshold value (maximum task or request size, which is considered fixed in this work) for each VM
10: Set the number of processors and other specifications (processor speed, memory size, user grouping factor,
    etc.) of VMs
11: Initially, set all the VMs in working mode
12: Set the broker policy (i.e., optimized response time is chosen in this work)
13: **Step-3: Resource allocation**
14: for VM = 1; VM $\leq$ length (VM list); VM++ do
15:     if length $(task_{i+1}) < length(task_i)$ then
16:         Add maxVal = $task_{i+1}$ infront of the task queue
17:     end if
18:     if length (taskVM) = 0 then
19:         upgrade VM task count (maxVal)
20:     end if
21:     Check threshold value
22:     Return VM ID
23:     Calculate probability for selected VM ID
24:     Use bee function to calculate global optimal solution
25:     Return VM ID
26:     if length(taskVM) $\geq$ 0 then
27:         Check the defined tasks allocation threshold for each VM
28:     end if
29:     Check available VMs and requests in queue
30:     Calculate probability for selected VM ID
31:     Use bee function to calculate global optimal solution
32:     Return VM ID
33:     **Step-4: Compute objective functions**
34:     Calculate processing time using Equation (4)
35:     Calculate response time using Equation (5)
36:     Calculate cost using Equations (6)–(8)
37:     **Step-5: Check the available resources**
38:     Check the broker policy
39:     Check the available and connected fogs or cloud datacenters
40:     Check the available VMs
41:     Check the remaining tasks in the queue
42:     Increase the task counter in the queue upto maxCount
43:     Go to step-3
44:     Repeat the process until requests are completed
45: end for
```

4. Simulation Results and Discussion

To determine the dependency of performance parameters on location-aware DCs of the cloud and fog environment, *N* number of regions, buildings, load requests, and a balancing policy were considered. A cloud analyst simulator was used to evaluate the efficiency of the proposed algorithm. This simulator is the extended version of cloudsim, which provides us with a real-world environment. In this experimental setup, the whole world was divided into six regions [34]. Here, we evaluated performance parameters in two scenarios. This work is the extension of Saman et al. [16], where scenario 1 was considered only for a single region. Scenario 1 consists of one region—namely, region 2. This region further considers two buildings and two fogs, and each building has ten homes in it. This scenario was simulated using the four algorithms: PSO, ACO, ABC, and our proposed algorithm HABACO. The performance metrics considered in this case were: hourly response time of buildings, processing time of data centres, and costs of VMs, DTC, and TC. However, in scenario II, we enhanced our system in terms of five regions and each region has a fog and a building. Every building has 60–160 homes in it. In this scenario, results were further obtained for 2 VMs and 5 VMs. The performance parameters for this scenario were: response time of buildings, processing time of fogs, VM costs, DTC, and TC. The results of the proposed algorithm were compared with the PSO, ACO, and ABC using the considered performance metrics. The aim of these assumptions was to check the performance of the proposed model that how efficiently it works for a single region and for multiple regions. The optimized response time service broker policy was used as a resource allocation policy.

The input parameters for both scenarios were: VM bandwidth was 1000 MB, DC architecture Was X86, VMM was Xen, memory per machine was 2048 MB, storage per machine was 100,000 MB, DC available bandwidth per machine was 10,000, DC number of processors per machine was 4, DC processor speed was 100 MIPS, users grouping factors were 100, requests grouping factors were 100, and executable task size was 250. According to these assumptions, results were obtained through simulations.

4.1. Response Time

Figure 4 shows the hourly response time for buildings using PSO, ACO, ABC, and hybrid HABACO algorithms considering two and five VMs, respectively. In Figure 4, we can see that with PSO, ACO, and ABC, the response time was greater as compared to the HABACO considering the 2 VMs and 5 VMs. Response time was optimized for both buildings simultaneously. For building-1: the average response time obtained using 2 VMs with the PSO, ABC, ACO, and HABACO was: 38 ms, 38 ms, 11 ms, and 9 ms, respectively. In this case, HABACO was found efficient upto 23% as compared to PSO and 81% as compared to ACO. It was found equivalent to ACO in terms of resource allocation for both buildings. In scenario 1, building 1 using 5 VMs, all algorithms were simulated for 15 ms, 25 ms, 10 ms, and 9 ms. Here, our proposed algorithm outperformed the PSO upto 60%, ACO by up to 90%, and it enhanced its efficiency by up to 36% in the case of ABC in scenario 1. For building-2 under scenario 1 with 2 VMs and one fog server: response times of 31 ms, 31 ms, 22 ms, and 12.5 ms were achieved using PSO, ACO, ABC, and hybrid HABACO, respectively. The proposed algorithm beats the previous algorithms by up to 40% in comparison to PSO and ABC, whereas it performed up to 44% better in comparison to ACO for building-2 using 2 VMs and one fog server. In scenario 1 using 5 VMs and one fog server, these algorithms performed the average resource allocation for the daily basis: 27 ms, 13.5 ms, 13 ms, and 12 ms with PSO, ABC, ACO, and HABACO, respectively. All of the performances of these algorithms were achieved based on the fixed set of the request size in that particular region, since these algorithms are stochastic in nature and evaluate the requests to their optimal time-slots. For building-2 using 5 VMs, our proposed algorithm performed upto 52% than PSO, 92% as compared to ACO, and 88% as compared to ABC, as shown in Figure 4.

Figure 5 shows the average response times of buildings using PSO, ABC, ACO, and HABACO for both the 2 and 5 VMs in scenario 2. Here, we observe that by increasing the number of VMs and regions, response time decreased after equivalent distribution of the fog servers in each region. However, the proposed algorithm gave us a more optimal solution in terms of response time by optimized request scheduling as compared to the other algorithms, as mentioned above. Results show the slight difference between the performance of PSO and ABC. However, ACO showed high performance as compared to PSO and ACO. The overall response time of this scenario is depicted in Table 1, where HABACO performed better than its counterpart algorithms.

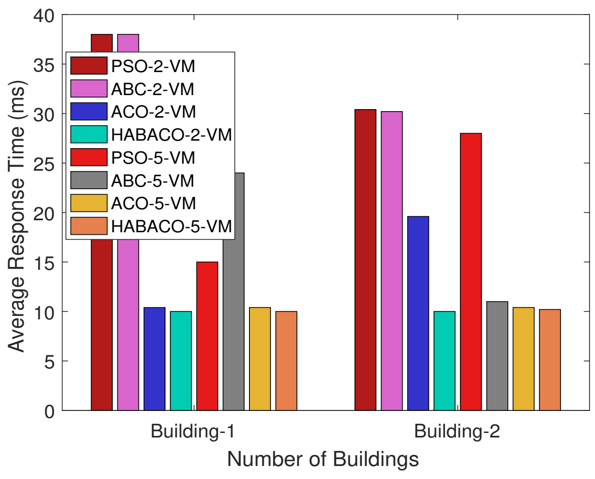

Figure 4. Average response time of buildings in scenario 1. ABC: artificial bee colony; ACO: ant colony optimization; HABACO: hybrid artificial bee ant colony optimization; PSO: particle swarm optimization.

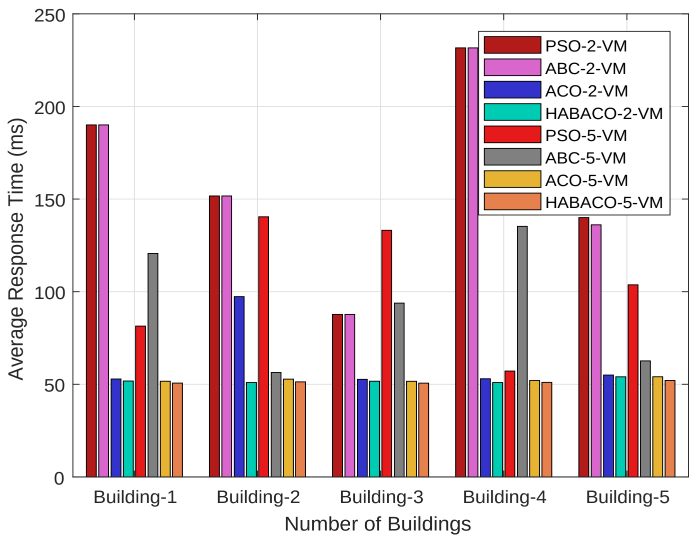

Figure 5. Average response time of buildings in scenario 2.

Table 1. Overall response time in scenario 2.

VMs	Load Balancing Algorithm	Average (ms)	Minimum (ms)	Maximum (ms)
2	PSO	154.83	38.07	66,055.39
	ABC	153.55	38.07	66,045.38
	ACO	79.64	39.53	223.17
	HABACO	53.67	38.07	72.13
5	PSO	140.27	38.51	72,051.46
	ABC	109.33	39.39	54,954.1
	ACO	52.64	38.42	69.49
	HABACO	52.63	38.42	69.49

4.2. Processing Time

Figure 6 shows the average processing time of fogs with PSO, ABC, ACO, and proposed hybrid HABACO. In addition, Figure 6 also shows the comparative analysis of the above-mentioned algorithms. Here, two fogs' processing capabilities are evaluated for two buildings. That is, one fog is assigned to one building for optimal resource allocation and efficient task processing. Fog-1 shows the effective distribution of the resources using these four algorithms. Furthermore, these algorithms took the average times of 25.5 ms, 25 ms, 3 ms, and 2 ms for any building on daily requests' scheduling bases using fog-1 and 2 VMs. Our proposed algorithm outperformed the other algorithms: the average processing time of HABACO was 7.8% as compared to PSO, 7.07% as compared to ABC, and 66% that of as compared to ACO. As this algorithm is the hybrid of ABC and ACO, it uses the best features of these algorithms and gives the best optimal solution to the consumers. In the case of the fog-1 with 5 VMs, these algorithms (i.e., PSO, ABC, ACO, and proposed hybrid HABACO) took the average times of 7 ms, 13.5 ms, 6 ms, and 2 ms, respectively. Here, our proposed algorithm performs better upto 28% in comparison to PSO, 14.8% in comparison to ABC and almost equivalent in comarison to ACO. Using fog-2 with 2 VMs and one building, it gives the optimal processing of the requests by considering the PSO, ABC, ACO, and proposed HABACO algorithms: 22 ms, 21 ms, 8 ms and 4 ms, respectively. Here, requests of the building were optimized using these algorithms and our proposed algorithm performed best: 19% as compared to PSO, 19.9% as compared to ABC and 50% as compared to ACO for 10 homes. For optimizing these requests in a more efficient manner, 5 VMs were then assigned with the fog in order to check its functionality. In this case, the resource allocation times taken by the PSO, ABC, ACO, and the proposed HABACO algorithms for the daily routine tasks were: 21 ms, 4 ms, 3 ms, and 3 ms, respectively. Our proposed algorithm outperformed the other algorithms, with an average processing time upto 70% in comparison to PSO, 1% in comparison to ACO, and 75% in comparison to ABC respectively". .

In case of scenario 2, the proposed algorithm took the least time to process the requests, as presented in Figure 7, showing a similar behavior as depicted for the response time. With five VMs, all algorithms performed better than with two VMs. With both 2 and 5 VMs, the overall processing time of fogs is shown in Table 2. It is concluded from Table 2 that increasing the number of VMs results in minimum processing time.

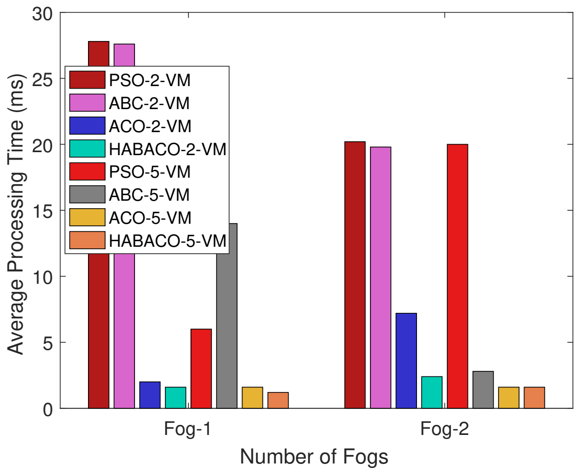

Figure 6. Average processing time of fogs for scenario 1.

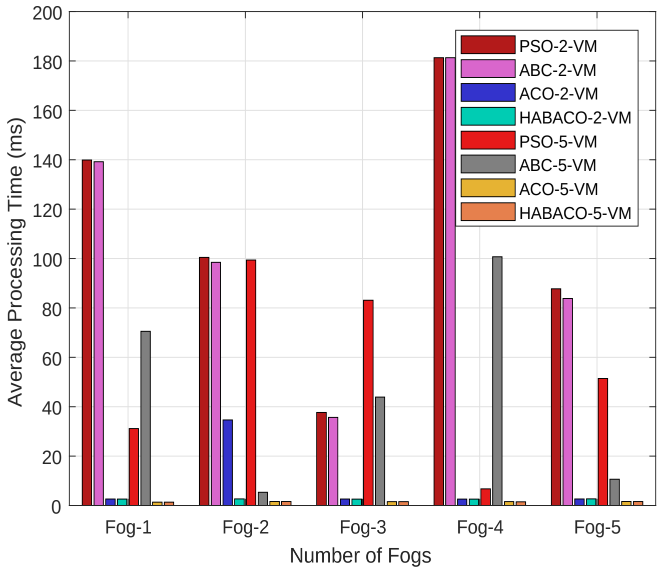

Figure 7. Average processing time of fogs in scenario 2.

Table 2. Processing time in scenario 2.

VMs	Load Balancing Algorithm	Average (ms)	Minimum (ms)	Maximum (ms)
2	PSO	103.86	0.05	66,000.85
	ABC	102.58	0.05	66,000.85
	ACO	21.67	0.05	61.92
	HABACO	2.64	0.04	10.98
5	PSO	89.29	0.06	71,998.92
	ABC	58.48	0.06	54,904.24
	ACO	1.56	0.05	8.62
	HABACO	1.56	0.03	10.41

4.3. Cost

In this section, three types of costs are calculated: VM cost, DTC, and TC. In any system, consumers have to pay some cost according to resource usage. So, the cost computed in this system is based on the number of fogs, cloud datacenters, the number of buildings, and the number of homes. Cost was also calculated for the two scenarios: scenario 1 and scenario 2. For scenario 1: VMs cost was optimized by using the PSO, ABC, ACO and our proposed algorithm, yielding values of upto $8, $9, $10, and $7, respectively, using 2 VMs. Our proposed algorithm gave the best optimal results in scenario 1, as shown in Figure 8. Using 5 VMs, the PSO, ABC, ACO, and HABACO, all algorithms gave the results upto $12, $13, $11, and $10.5, respectively. DTC was computed as $152, $151, $150, and $149 using PSO, ABC, ACO, and HABACO algorithms, respectively, considering 2 VMs as displayed in Figure 9. For 5 VMs, DTC was computed as $148, $147, $154, and $146. TC was calculated as $170, $175, $168, and $167 for all algorithms using 2 VMs as shown in Figure 10. For 5 VMs, TC was computed as $172, $167, $172, and $167, respectively, for PCO, ABC, ACO, and HABACO in scenario 1. Our proposed algorithm gave cost-effective results throughout scenario 1 of the proposed system.

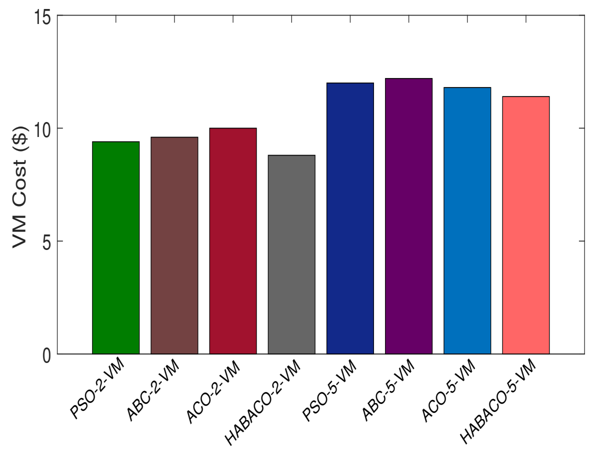

Figure 8. VM cost in scenario 1.

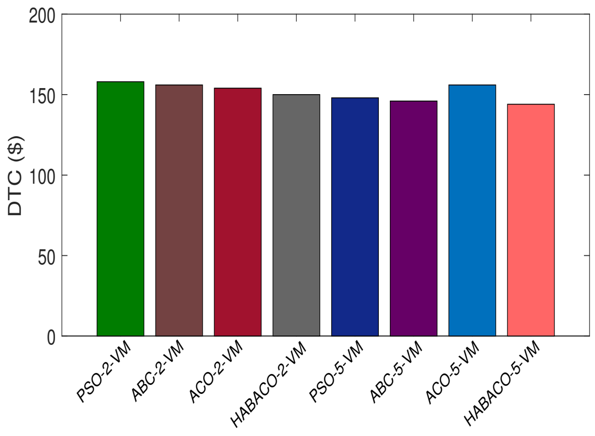

Figure 9. Data transfer cost (DTC) in scenario 1.

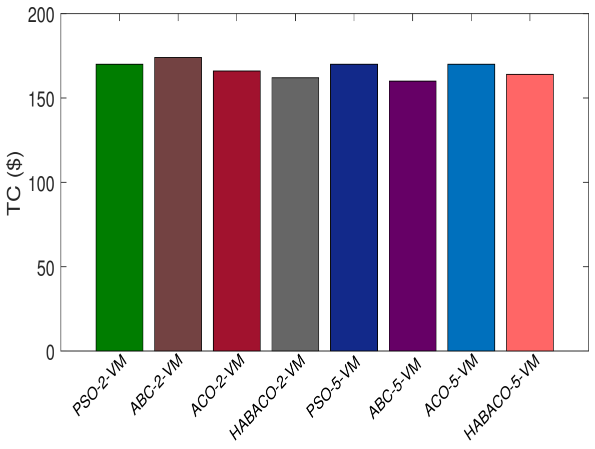

Figure 10. Total cost (TC) in scenario 1.

For scenario 2, VM cost was calculated for both 2 and 5 VMs, as displayed in Figure 11. As the number of VMs increased, the cost of the system also increased. However, service providers can tackle this issue by efficiently managing energy, as discussed earlier. Four algorithms were implemented for effective resource management; however, our proposed HABACO algorithm showed high performance as compared to its counterpart algorithms. The total DTC is shown in Figure 12. DTC is related to VMs and techniques. According to these resources, if consumers' requests are efficiently managed, then they have to pay less cost. Here, with five VMs, DTC was less than with two VMs. Moreover, the cost with HABACO was minimum as compared to the other algorithms. The TC of this system with both numbers of VMs is presented in Figure 13.

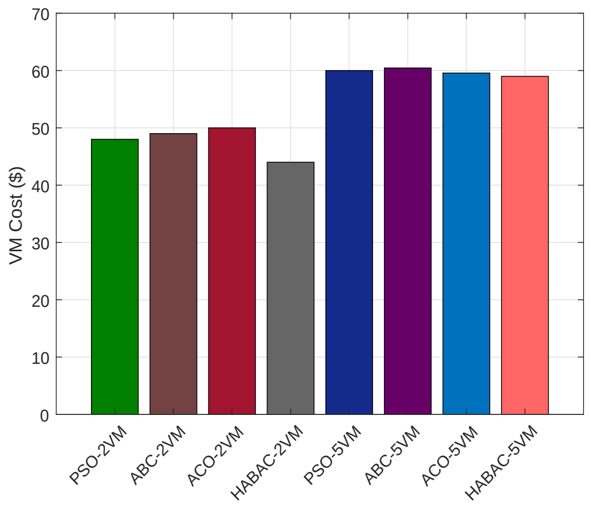

Figure 11. VM cost in scenario 2.

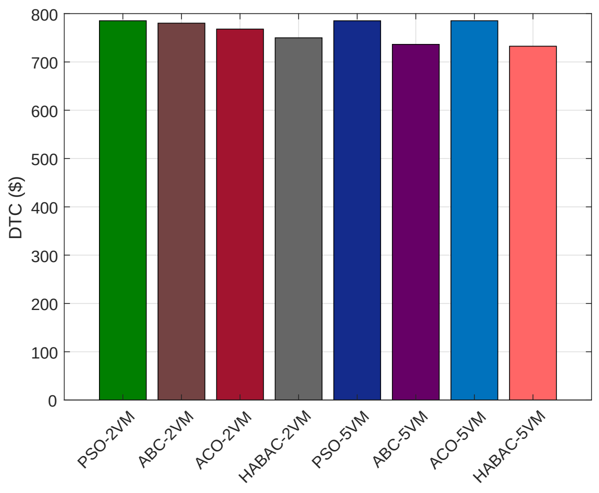

Figure 12. DTC in scenario 2.

Based on the simulation results, we can conclude that the proposed algorithm HABACO outperformed ABC, ACO, and PSO. The reason behind HABACO's superior performance is that it is a mixture of the best features of the ABC and ACO algorithms. PSO and ABC gave global best and local best solutions, respectively; however, response time, processing time, execution time, and cost were slightly higher due to their slow convergence. However, ACO sometimes becomes stuck in local optima, so it cannot find the global optimal solution. To overcome this issue, the ABC fitness step is added in the ACO algorithm, which yields better response time, processing time, cost, and execution time due to its higher convergence rate.

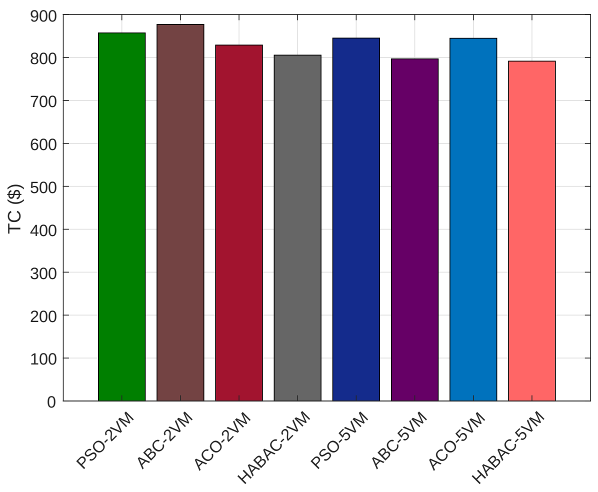

Figure 13. TC in scenario 2.

5. Conclusions and Future Work

In this paper, we presented a cloud- and fog-based model for efficient resource management in SGs. Moreover, a cloud–fog–based SG model is proposed to connect these domains. There is a great potential of using cloud–fog computing to serve in the SG domain regarding resource management, as fog computing helps in minimizing delay and enhancing the overall response time of the system. Moreover, four load balancing algorithms—PSO, ABC, ACO, and our proposed HABACO algorithm—were used to manage resources optimally in a cloud and fog based environment, and their results were compared with each other. Here, two scenarios were considered. Four load balancing algorithms were implemented in both scenarios. Simulation results showed that the overall performance of HABACO was better as compared to the other techniques presented in scenario 1. In the scenario 2, the performance of the proposed algorithm was also found to be better as compared to PSO, ABC, and ACO. From the numerical results, we conclude that in scenario 1, the response time of our proposed algorithm outperformed PSO upto 60%, ACO upto 90% and it enhances its efficiency upto 36% in case of the ABC using 2 VMs in scenario 1”. Our proposed algorithm also outperformed the other algorithms in terms of average processing time: 7.8% as compared to PSO, 7.07% as compared to ABC and 66% as compared to ACO for average processing time using one fog server and 2 VMs. In this manner, the proposed hybrid algorithm outperformed its counterpart algorithms due to its capability of finding the both global and local optimal solutions.

In the future, the investigation of this proposed model in the commercial sector is another direction of our research. Furthermore, it will be extended towards the management of multiple load balancing applications. That is, optimal scheduling of appliances, optimal generation of microgrids, etc. for enhancing the efficiency of SGs. The proposed HABACO algorithm can be implemented in real-time environments instead of simulation-based scenarios, and will be compared with more artificial intelligence based methods in the future.

Author Contributions: All authors equally contributed.

Acknowledgments: The present research has been conducted by the Research Grant of Kwangwoon University in 2018.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghasemkhani, A.; Hassan, M.; Ashkan R.-K.; Amjad A.-M. Optimal design of a wide area measurement system for improvement of power network monitoring using a dynamic multiobjective shortest path algorithm. *IEEE Syst. J.* **2015**, *11*, 2303–2314. [CrossRef]
2. Signorini, M. Towards an Internet of Trust: Issues and Solutions for Identification and Authentication in the Internet of Things. Ph.D. Dissertation, University of Pompeu Fabra, Barcelona, Spain, 2015.
3. Blanco-Novoa, Ó.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. An Electricity Price-Aware Open-Source Smart Socket for the Internet of Energy. *Sensors* **2017**, *17*, 643. [CrossRef] [PubMed]
4. Al Faruque, M.A.; Korosh, V. Energy management-as-a-service over fog computing platform. *IEEE Int. Things J.* **2016**, *3*, 161–169. [CrossRef]
5. Aazam, M.; Eui-Nam, H. Fog Computing and Smart Gateway Based Communication for Cloud of Things. In Proceedings of the 2014 International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 27–29 August 2014.
6. Bonomi, F.; Rodolfo, M.; Jiang, Z.; Sateesh, A. Fog Computing and its Role in the Internet of Things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012.
7. Chiang, M.; Tao, Z. Fog and IoT: An overview of research opportunities. *IEEE Int. Things J.* **2016**, *3*, 854–864. [CrossRef]
8. Suryawanshi, R.; Ganesh, M. Focusing on mobile users at the edge of internet of things using fog computing. *Int. J. Sci. Eng. Technol. Res.* **2015**, *4*, 3225–3231.
9. Luan, T.H.; Gao, L.; Li, Z.; Xiang, Y.; Wei, G.; Sun, L. Fog computing: Focusing on mobile users at the edge. *arXiv Preprint* **2015**, arXiv:1502.01815.
10. Luo, F.; Zhao, J.; Dong, Z.Y.; Chen, Y.; Xu, Y.; Zhang, X.; Wong, K.P. Cloud-based information infrastructure for next-generation power grid: Conception, architecture, and applications. *IEEE Trans. Smart Grid* **2016**, *7*, 1896–1912. [CrossRef]
11. Gan, L.; Ufuk, T.; Steven, H.L. Optimal decentralized protocol for electric vehicle charging. *IEEE Trans. Power Syst.* **2013**, *28*, 940–951. [CrossRef]
12. Hao, X.; Fu, M.; Lin, Z.; Mou, Y. Decentralized optimal scheduling for charging and discharging of plug-in electric vehicles in smart grids. *IEEE Trans. Power Syst.* **2016**, *31*, 4118–4127.
13. Mohamed, N.; Jameela, A.-J.; Imad, J.; Sanja, L.-M.; Sara, M. SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services. *IEEE Access* **2017**, *5*, 17576–17588. [CrossRef]
14. Dam, S.; Gopa, M.; Kousik, D.; Parmartha, D. An Ant-Colony-Based Meta-Heuristic Approach for Load Balancing in Cloud Computing. *Appl. Comput. Int. Soft Comput. Eng.* **2017**, *204*. [CrossRef]
15. Chen, S.-L.; Chen, Y.-Y.; Kuo, S.-H. CLB: A novel load balancing architecture and algorithm for cloud services. *Comput. Electr. Eng.* **2017**, *58*, 154–160. [CrossRef]
16. Saman, Z.; Nadeem, J.; Asif, K.; Bibi, R.; Fatima, J.M.; Maida, Z. A Cloud-Fog-Based Smart Grid Model for Efficient Resource Utilization. In Proceedings of the 14th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC-2018), Limassol, Cyprus, 25 June 2018.
17. Hussain, B.; Hasan, Q.U.; Javaid, N.; Guizani, M.; Almogren, A.; Alamri, A. An Innovative Heuristic Algorithm for IoT-enabled Smart Homes for Developing Countries. *IEEE Access* **2018**, *6*, 15550–15575. [CrossRef]
18. Yoldas, Y.; Ahmet, O.S.M.; Muyeen, A.V.V.; Irfan, A. Enhancing smart grid with microgrids: Challenges and opportunities. *Renew. Sustain. Energy Rev.* **2017**, *72*, 205–214. [CrossRef]
19. Rajarajeswari, R.K.V.; Ashutosh, M. Demand Side Management in Smart Grid using Optimization Technique for Residential, Commercial and Industrial Load. *Indian J. Sci. Technol.* **2016**, *9*, 43. [CrossRef]
20. Barbato, A.; Antonio, C.; Lin, C.; Fabio, M.; Stefano, P. A distributed demand-side management framework for the smart grid. *Comput. Commun.* **2015**, *57*, 13–24. [CrossRef]
21. Mora, M.; O'Connor, R.V.; Tsui, F.; Marx Gómez, J. Design methods for software architectures in the service-oriented computing and cloud paradigms. *Softw. Pract. Exp.* **2018**, *48*, 263–267. [CrossRef]
22. Armbrust, M.; Armando, F.; Rean, G.; Anthony, D.; Joseph, R.K.; Andy, K.; Gunho, L.; David, P.; Ariel, R.; Ion, S.; et al. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]

23. Xia, Z.; Wang, X.; Zhang, L.; Qin, Z.; Sun, X.; Ren, K. A privacy-preserving and copy-deterrence content-based image retrieval scheme in cloud computing. *IEEE Trans. Inform. Forensics Secur.* **2016**, *11*, 2594–2608. [CrossRef]
24. Fu, Z.; Ren, K.; Shu, J.; Sun, X.; Huang, F. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 2546–2559. [CrossRef]
25. Xia, Z.; Wang, X.; Sun, X.; Wang, Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 340–352. [CrossRef]
26. Khiyaita, A.; El Bakkali, H.; Zbakh, M.; El Kettani, D. Load balancing cloud computing: state of art. In Proceedings of the 2012 National Days of Network Security and Systems (JNS2), Marrakech, Morocco, 20–21 April 2012.
27. Sambit, K.M.; Bibhudatta, S.; Priti, P.P. Load Balancing in Cloud Computing: A big Picture. *J. King Saud Univ.-Comput. Inform. Sci.* **2018**, 1–32. [CrossRef]
28. Nikhit, P.; Umesh, K.L.; Nitin, A. A Hybrid ACHBDF Load Balancing Method for Optimum Resource Utilization In Cloud Computing. *Int. J. Sci. Res. Comput. Sci. Eng. Inform. Technol.* **2017**, *2*, 367–373.
29. Bitam, S.; Sherali, Z.; Abdelhamid, M. Fog computing job scheduling optimization based on bees swarm. *Enter. Inform. Syst.* **2017**, *12*, 373–397. [CrossRef]
30. Reka, S.S.; Ramesh, V. Demand side management scheme in smart grid with cloud computing approach using stochastic dynamic programming. *Perspect. Sci.* **2016**, *8*, 169–171. [CrossRef]
31. Moghaddam, M.H.Y.; Alberto, L.-G.; Morteza, M. On the performance of distributed and cloud-based demand response in smart grid. *IEEE Trans. Smart Grid* **2017**, 1–14. [CrossRef]
32. Chekired, D.A.; Lyes, K. Smart Grid Solution for Charging and Discharging Services Based on Cloud Computing Scheduling. *IEEE Trans. Ind. Inform.* **2017**, *13*, 3312–3321. [CrossRef]
33. Gu, C.; Fan, L.; Wu, W.; Huang, H.; Jia, X. Greening cloud data centers in an economical way by energy trading with power grid. *Future Gener. Comput. Syst.* **2018**, *78*, 89–101. [CrossRef]
34. Wickremasinghe, B.; Rajkumar, B. CloudAnalyst: A CloudSim-based tool for modelling and analysis of large scale cloud computing environments. *MEDC Proj. Rep.* **2009**, *22*, 433–659.
35. Kousalya, K.; Balasubramanie, P. To improve ant algorithm's grid scheduling using local search. *Int. J. Comput. Cogn.* **2009**, *7*, 47–57.
36. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Erciyes University: Kayseri, Turkey, 2005; Volume 200.