epcAware: A Game-Based, Energy, Performance and Cost-Efficient Resource Management Technique for Multi-Access Edge Computing

Muhammad Zakarya, Lee Gillam, Hashim Ali, Izaz Ur Rahman, Khaled Salah, Rahim Khan, Omer Rana, and Rajkumar Buyya

Abstract—Internet of Things (IoT) is producing an extraordinary volume of data daily, and it is possible that the data may become useless while on its way to the cloud, due to long distances. Fog/edge computing is a new model for analysing and acting on time-sensitive data, adjacent to where it is produced. Further, cloud services provided by large companies such as Google, can also be localised to improve response time and service agility. This is accomplished through deploying small-scale datacentres in various locations, where needed in proximity of users; and connected to a centralised cloud that establish a multi-access edge computing (MEC). The MEC setup involves three parties, i.e., service providers (IaaS), application providers (SaaS), network providers (NaaS); which might have different goals, therefore, making resource management difficult. Unlike existing literature, we consider resource management with respect to all parties; and suggest game-theoretic resource management techniques to minimise infrastructure energy consumption and costs while ensuring applications' performance. Our empirical evaluation, using Google's workload traces, suggests that our approach could reduce up to 11.95 percent energy consumption, and ~17.86% user costs with negligible loss in performance. Moreover, IaaS can reduce up to 20.27 percent energy bills and NaaS can increase their costs-savings up to 18.52 percent as compared to other methods.

Index Terms—Resource management, Internet of Things, multi-access edge computing, energy efficiency, performance, game theory

1 Introduction

Real-time applications such as on-line gaming and video conferencing have on-demand requirements to provide high-quality results within the agreed time e.g., shorter response time through communication with the closest application server. Using cloud platform to deploy real-time applications offers several benefits including reduced OpEx (operational costs), but not necessarily, and on-demand resource allocation - assign resources based on needs of the application. However, real-time applications may be sensitive to the quality of network e.g., latency between users and services. Therefore, real-time application requirements could be addressed through combining edge computing technology with MEC and fog - which allows computations to be accomplished at the edge of the network. The rationale of commissioning this technology is to offer services within the proximity of customers and closer to where computational results are desirable. This can be achieved through deploying small-scale datacentres (cloudlets) closer to customers and connected to regional datacentres. Note that, the management systems to run and practice such infrastructures are, largely, implemented now – for example AWS outposts[1] and revised OpenStack [1], and lots are under investigation. Further, with the Multi-access edge computing (MEC) framework, there are certain questions that still need to be investigated. For example; (i) where these cloudlets should be deployed; (ii) which services should be installed; (iii) where and how the resources should be allocated to users' applications; (iv) how user mobility (service migration) should be handled; and (v) how the MEC framework should be optimised to minimise or maximise various objectives such as users' monetary costs, energy consumption and workload performance in terms of latency, execution time and throughput etc. Albeit, (iii) to (v) can be seen, largely, similar to traditional clouds; but the management policies should be redesigned for fog infrastructure.

This research aims to examine resource allocation/placement and consolidation challenges associated with the MEC platforms. The main questions that this research will

- *Muhammad Zakarya, Hashim Ali, Izaz Ur Rahman, and Rahim Khan are with the Department of Computer Science, Abdul Wali Khan University, Mardan, Khyber Pakhtunkhwa 23200, Pakistan. E-mail: {mohd.zakarya, hashimali, izaz, rahimkhan}@awkum.edu.pk.*
- *Lee Gillam is with the University of Surrey, GU2 7XH Guildford, United Kingdom. E-mail: l.gillam@surrey.ac.uk.*
- *Khaled Salah is with the Khalifa University, Abu Dhabi, UAE. E-mail: khaled.salah@ku.ac.ae.*
- *Omer Rana is with the University of Cardiff, CF10 3AT Cardiff, United Kingdom. E-mail: ranaof@cardiff.ac.uk.*
- *Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Parkville, VIC 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 4 Jan. 2020; revised 10 June 2020; accepted 24 June 2020. Date of publication 26 June 2020; date of current version 15 June 2022. (Corresponding author: Hashim Ali.) Digital Object Identifier no. 10.1109/TSC.2020.3005347

1. https://aws.amazon.com/outposts/

answer include: (i) where cloudlets should be installed to meet users' demand while abating the global infrastructure cost (CapEx - capital expenditures + OpEx); (ii) once the platform has been installed, what, how and where should global control, scheduling and management services be deployed; and (iii) at what size/scale. Furthermore, this will open opportunities for advising a generic resource allocation/placement and management/migration framework for various kind of fog/edge services. While the model of cloud computing provided by a few mega/large providers (Google and Amazon AWS) is still widely used, the beginning of innovative and emerging technologies such as internet of things (IoT) applications, edge computing and MECs is challenging this approach [1]. To cope with this technological change, cloud and network communities are now working in the direction of large-scale distributed, small-sized, infrastructures (cloudlets) that are installed at the edge of the network - closer to users i.e., fog infrastructure (hence, distributed-shape applications that consist of various modules) [2], [3]. The fog, edge and cloud paradigm are attracting rising interest as it also improves services' agility and performance in terms of response time. For example, IoT applications can take benefits from edge nodes' deployment to perform real-time analysis while conserving remote clouds for in-depth data analytics [4]. This can be seen as a mixture of fog/edge/cloudlet/MEC (the latter now being multi-access rather than merely mobile) – where MEC suggests being within the radio access network (RAN) but fog/edge could relate consumer devices aggregating data from sensors (before passing to the remote cloud). Furthermore, "cloudlet" as a mini datacentre, presumably consistent with a large cloud provider's provisions, would be able to support such aggregation but would always be further from one or more such sensors. Nevertheless, the need of such a cloudlet has no direct relationship with the RAN.

In addition to recognising where edge clouds should be deployed/installed, the drivers of such an evolution lie in the design of suitable management systems that will permit: (i) an operator i.e., cloud, network, or edge to aggregate, supervise and expose such massively distributed resources; and (ii) to implement new kinds of services that may be deployed and managed by the operator/user. However, designing such a management system is challenging because fog/edge infrastructures differ from traditional clouds regarding heterogeneity, dynamicity, the possible huge distribution of resources, and economics of scale - if smaller, heterogeneity is less likely. The objective of this research is to explore placement-related questions of a massively distributed MEC infrastructure. The research is organised around the subsequent activities: (i) propose placement algorithms that can satisfy QoS expectations while optimising different objectives such as infrastructure cost minimisation, energy requirements and reliability (performance in terms of response time and QoS); and (ii) evaluate the proposed algorithms through simulations by leveraging the iFogSim toolkit [5] and Google cluster datasets [6].

Our work makes the following major contributions:

- we model MECs resource allocation and service migration problems using non-cooperative (NC) and semi co-operative (SC) game-theoretic approaches;
- bidding-based, game-theoretic resource allocation and migration algorithms are proposed that ensure benefits to all parties within the MEC; and
- performance of the algorithms is evaluated through simulations in iFogSim [5], using real workload traces from Google's cluster [6].

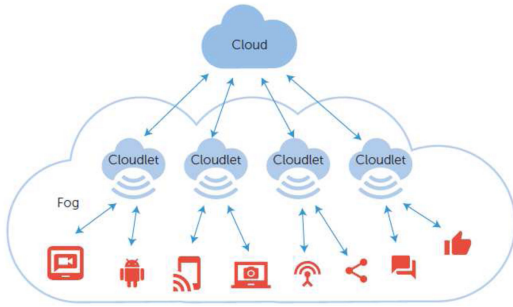

Fig. 1. Cloud, cloudlet and fog architecture [9].

The rest of the paper is organised as follows. In Section 2, we discuss MECs and resource management. In Section 3, we model the MECs resource allocation problem as a game. A game theoretic solution is presented, is Section 4, to solve the problem. We validate the proposed approach using real workload traces from Google cluster in Section 5. We offer an overview of the related work in Section 6. Finally, Section 7 concludes the paper with several directions for future research.

2 Background

MECs offer finite resources at the edge of the network; making it possible to run the user's application in its proximity, as shown in Fig. 1. The resources at the edge are provisioned at the cloudlet or MEC server. Moreover, the application, in cloudlet, runs at one (or more than one) hop communication distance unlike to their native execution either in the mobile/fog device or core internet (remote cloud) - zero and two+ hop distance, respectively. Therefore, application latency might be potentially affected depending on the network services. The finite number of cloudlets' resources also put questions on their efficient allocation to connected users. Although, there are several proposals to share the resources of several cloudlets in a particular geographic area [7]. However, this will be a challenging problem when the cloudlet resources are offered by different service providers having different goals and objectives. Similarly, beside providers, mobile users, who run the applications, are also usually selfish and competitive; each user wants to optimise his/her own pay-off or application's performance [8].

Moreover, different resources can be offered at different costs, energy consumption and performance levels. Resource placement can significantly affect service providers (IaaS, network) and customers (SaaS) economics [10]. For example, reduced performance of applications increases users' costs as well as energy consumption. Therefore, it is essential to provide appropriate resources in order to meet application QoS requirements and providers objectives. Moreover, if mobility is involved - users are moving or application modules are explicitly migrated among hosts for energy efficiency or performance gains, then resource management complexities will potentially increase. Appropriate resource management

TABLE 1
List of Players With Objectives and Conflicting Aims

Player	Objectives	Conflicting aims
IaaS	Revenue, Energy	lower performance, SLA violations
SaaS	Performance, Cost	non-appropriate resources, user costs, energy consumption
NaaS	Bandwidth QoS	latency that produces SLA violations, less revenue due to users QoE

techniques are, therefore, essential to cope with various objectives. In this paper, we elaborate: (i) how the infrastructure and available resources (IaaS, SaaS, NaaS) should be managed in order to increase service agility, performance in terms of response time and minimise the energy-related costs; (ii) investigate how users' workloads or services should be run and, most importantly, which resources or instance types should be provisioned for their execution; (iii) develop algorithms and mathematical models to solve the resource allocation (network and computing) problem, efficiently, for MEC to support emerging mobile applications; (iv) investigate the effect of CPU heterogeneity and co-location; and (v) realize and implement the models and algorithms into a software to demonstrate their feasibility and practicability.

3 Problem Description

Management of resources in MECs is very challenging, because offering quality services to the end-users depends on various players, with moderately conflicting goals, such as infrastructure owners (IaaS), network operators (network as a service - NaaS), and application providers (SaaS), where each player controls a particular part of the whole system. These three business models are further explained in Section 3.1. Integral to the problem is the fact that both communication and computation capacity are needed to guarantee high QoS in terms of low response time and high throughput. Since, each player may have different objectives to optimise where the objectives of one player may potentially affect the objectives of another player and vice versa - as shown in Table 1. Existing works [1], [7], [10], [11] either assume that the whole infrastructure is managed by a single player, largely, the resource providers [10], or separate the management of the network, application resources from the core edge computing capacity [7]. We believe, resource allocation in MECs should be assumed as a multi-objective optimisation problem in such a way that players' competition for their objectives can be optimised. The objectives of all parties are somehow aligned. For instance, the insufficient resource provisions for the user's application would ruin revenues for all parties concerned. As a result, customers will prefer to go elsewhere, and so do the providers. Therefore, there is an incentive to only minimise to the point at which such situations are avoided i.e., either all 'win' together (can minimise absent impact) or all 'lose' together (minimising has impact). Moreover, simple optimisation methods cannot ensure a win-win situation for all players, in similar resource allocation scenarios [12], [13]. Therefore, we use game theory to model and solve such a complex, multi-player resource allocation problem.

Largely, SaaS providers host their applications on virtualised resources provided by an IaaS provider. Moreover, SaaS providers need to comply with every application's quality of service (QoS) requirements, as described in Service Level Agreement (SLA) with the customers, which determine the SaaS revenue on the basis of achieved level of performance. However, application performance is not only dependent on computational resources (provided by IaaS), but, as well as, on the network bandwidth provided by the network operators. Users quality of experience (QoE) with network operators affect IaaS, as well as, SaaS revenues. Similarly, service providers would prioritise their workloads based on the nature of applications (native or third-party) [7]. Furthermore, in MECs, the network operators could be: (a) the IaaS owners (internal network); and (b) third-party mobile network operators (external network). Therefore, network resources from various providers should be provisioned at affordable prices. The focus of SaaS providers would be to maximise their revenues through minimising SLA violations, while reducing the total cost of using computation, as well as, network resources provided by the IaaS and third-party network service providers. Moreover, in the case of multiple IaaS providers distributed over various geographic areas or cities (MECs), SaaS providers would compete and bid for the use of infrastructural resources based on prices. On the other hand, the IaaS and third-party network service providers could maximise their revenues through providing their virtualised resources as much as possible. Moreover, IaaS providers could maximise their resource usage (utilisation) in order to minimise energy consumption.

3.1 Business Models

Largely, IaaS providers sell their resources in the form of VMs and storage. Users are billed, for pre-defined VMs/container types; in currency per hour (using PAYG model - pay as you go) based on their workload execution times. AWS EC2, and Google Cloud are common examples of IaaS services. According to Gartner Inc.,[2] the world's IaaS market, while AWS ranked no. 1, grew 21.6 percent in 2019 to total $38.9 billion, up from $30.5 billion in 2018. SaaS is a web-based software delivery model in which the software is hosted in a centralised datacentre; and is sold on subscription basis - usually on monthly fee or annual fee. Google App Engine, and on-line games are the most common examples of SaaS service model. The cost of a SaaS application varies with respect to its certain parameters such as the total number of users accessing it. Besides these, certain SaaS providers offer freemium services (with limited functionalities) - Gmail and completely free software [10]. The world's SaaS market was valued $134.44 billion in 2018; and is expected to grow as high as $220.21 billion by 2022.[3]

Similarly, NaaS providers have network infrastructure which is virtually offered to a third-party in the form of bandwidth capacities using an on-demand provisioning model [14]. NaaS enables IaaS companies to use their network with high dynamism, scalability and flexibility, adapting to SaaS requirements as they emerge. Network virtualisation, software defined networks (SDNs), bandwidth on demand,

2. https://www.gartner.com/en/newsroom/
3. https://www.prnewswire.com/news-releases/

TABLE 2
Notations Used in Problem Formulation

Notation	Description
N	List of players so that $m \in N$
K	List of available resources
C_K^m	Type $k \in K$ resources offered by $m \in N$ provider
C	List of total offered resources by $m \in N$
M	Set of applications
$\mathcal{H}$	List of hosts in datacentre such that $h \in \mathcal{H}$
L	List of users such that $u \in U$
$\mathcal{M}$	Number of cloudlets such that $e \in \mathcal{M}$
j	A particular job that belongs to an application $\in M$
R	Resource request matrix
A	Resource allocation matrix
D	Allocation decision
U	Utility function of all providers
X_{iN}	Resource - provider mapping function, constraint
x_{ij}	VM - host mapping function, constraint
b_{cost}	Cost of network bandwidth B
T_{rate}	Rate of transmission over the B
bid_j	Bid of jth provider for its resources
e_{ij}	Energy consumed at host j for VM i
t_{ij}	Runtime of VM i at host j
t_k	Execution time of application k
X_{ij}	Mapping function of provider i to application j
p_{ij}^{IaaS}	Utility of IaaS for application j on host i
p_{ij}^{SaaS}	Utility of SaaS for application j of user i
p_{ij}^{NaaS}	Utility of NaaS for application j for ith channel
E_m	Energy consumed during a VM migration
VM_{data}	The data copied during the migration of VM

datacentre connectivity, and virtual private networks (VPNs) are some well-known approaches to network services [15]. Very similar to SaaS, NaaS resources along-with mobile networks (4G, 5G) are also offered on subscription fee or, usually, for a contracted period of time. However, renting high network capacity, WAN, dedicated links and bandwidth may not be cost-effective for IaaS providers. NaaS services could also be oversubscribed to increase revenues. In 2016, the NaaS market was valued at $1.85 billion.[4]

3.2 Mathematical Formulation

In our game, we assume three players with conflicting goals - as shown in Table 1: (i) IaaS – whose aim is to minimise energy consumption through consolidating the workload onto the fewest resources; (ii) SaaS – who wants to maximise service performance and avoids SLA violations (and, therefore, increase revenue); and (iii) third-party network service providers and operators – whose aim is to minimise network traffic (increase bandwidth) in order to ensure QoS in terms of performance (response time). We assume that $N = \{1, 2, \ldots, n\}$ denotes a set of all service providers that act as players. A list of all mathematical notations can be found in Table 2. Moreover, each player has a set of $K = \{1, 2, \ldots, k\}$ resources including computation (IaaS), application (SaaS) such as on-line services and emails, and communication (NaaS) resources. The mth service provider denotes its offered resources as $C^m = \{C_1^m, C_2^m, \ldots, C_K^m\}$ where C_k^m is the amount of resources of type k offered by service provider m. Therefore, all offered resources at various service providers is given by:

$$C = \left\{ \sum_{m \in N} C_1^m, \sum_{m \in N} C_2^m, \ldots, \sum_{m \in N} C_K^m \right\}. \quad (1)$$

Moreover, every job j (i.e., application module) asks for a set of resources (in the form of coalition) that belongs to a set of all applications given by $M = \{M_1 U M_2 \ldots U M_K\}$ subject to the condition that every application module is allocated resources once in every $m \in N$ - an application module can run exactly once. We assume that every application module j runs in a virtual machine (VM) or container. Furthermore, it is possible that a job may comprise a multiplicity of amounts of containers/VMs/network resources which can be co-located - unless there can be only one VM or container per host or application, in which case this readily simplifies. However, to simplify our formulation, we assume that each application module requires one VM or container, at most and the total number of VMs or containers provisioned cannot exceed the application M resource requirements R (subject to constraint in Eq. (2))

$$\sum_{i=1}^{VMs|containers} M_i \geq R_M. \quad (2)$$

Suppose a MEC system with a single cluster (cloud datacentre), several edge locations (cloudlets) and numerous mobile/fog devices. These resources are interconnected through networks such that cloudlets are in proximity to fog devices. An application's modules are distributed and run over these resources. The cloud datacentre which consists of $\mathcal{H}$ heterogeneous hosts and each host is denoted by h, such that $1 \leq h \leq \mathcal{H}$. For $k \in K$ resources (such as CPU, memory, storage, network) each h can be denoted as a capacity vector $\mathcal{C}^h = \{c_1^h, c_2^h, c_3^h, \ldots\ldots, c_k^h\}$; and each kind of resource is denoted as n. For example, $h(2, 4, 10, 1)$ describes that a particular host h has 2 CPUs, 4 GB memory, 10 GB disk storage and 1 Gbps network card. Moreover, we assume that there are $\mathcal{M}$ edge locations (cloudlets) and each edge cloud e consists of several heterogeneous hosts S; and each edge host $s \in S$ resources are also represented as capacity vector $\mathcal{C}^s$ – similar to cluster host $\mathcal{C}^h$. Moreover, cloudlet resources are extremely lower than datacentre resources i.e $\sum S << \sum \mathcal{H}$ and $C^s << C^h$. The resources in datacentre and cloudlets are virtualised, therefore, offered through VMs. Each VM can run a particular application module or job. The application or job submitted by a particular user u is denoted as J_u, where $u \in \{1, 2, \ldots\ldots, U\}$. Every application comprises several modules that run concurrently [9]. Furthermore, a variety of VM or container types are predefined by each cloud provider (cloud and cloudlets); and each type's resources are encoded by the capacity vector $\mathcal{R}$ such that $R_u = \{r_{u1}, r_{u2}, \ldots\ldots, r_{uj}, \ldots\ldots, r_U\}$. Note that, each VM or container can run a single job (application module) at a time (subject to constraints in Eq. (12)), both in the remote datacentre and cloudlets. Various resources like CPU, memory, storage and network of a host $h \in \mathcal{H}$ or $s \in S$ will be occupied, only, when a particular VM or container is created on $h \in \mathcal{H}$ or $s \in S$. Mobility of application modules is, therefore, possible through consolidation with VM migration [10].

4. https://www.cisco.com/c/en/us/solutions/enterprise-networks/

With the above terms, resource requests for a particular job j (or user) can be defined as a $u \times v$ dimensional matrix ($\mathcal{R}^j$); where rows represent the VM or container type and columns denote the amount of various resources associated with the VM/container type

$$R^j = \begin{bmatrix} r_1^j \\ r_2^j \\ \dots \\ r_u^j \end{bmatrix} = \begin{bmatrix} r_{11}^j r_{12}^j r_{13}^j \dots r_{1v}^j \\ r_{21}^j r_{22}^j r_{23}^j \dots r_{2v}^j \\ \dots \ \dots \ \dots \ \dots \ \dots \\ r_{u1}^j r_{u2}^j r_{u3}^j \dots r_{uv}^j \end{bmatrix}. \tag{3}$$

Note that, the request matrix R is an augmentation of all the request matrices (from all the service providers) as given below:

$$R = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_U \end{bmatrix} = \begin{bmatrix} r_{11} r_{12} r_{13} \dots r_{1v} \\ r_{21} r_{22} r_{23} \dots r_{2v} \\ \dots \ \dots \ \dots \ \dots \ \dots \\ r_{u1} r_{u2} r_{u3} \dots r_{UV}^j \end{bmatrix}. \tag{4}$$

We assume that a particular job can be allocated to at most one host; and various resources mean CPU cores, RAM, storage capacity and network bandwidth. Moreover, for a particular host h, a possible resource allocation state can be described as an allocation matrix $\mathcal{A}^h$

$$A^h = \begin{bmatrix} a_1^h \\ a_2^h \\ \dots \\ a_u^h \end{bmatrix} = \begin{bmatrix} a_{11}^h a_{12}^h a_{13}^h \dots a_{1v}^h \\ a_{21}^h a_{22}^h a_{23}^h \dots a_{2v}^h \\ \dots \ \dots \ \dots \ \dots \ \dots \\ a_{u1}^h a_{u2}^h a_{u3}^h \dots a_{uv}^h \end{bmatrix}, \tag{5}$$

where a_{ab}^h represents the amount of resources b on a particular host h allocated to a container or VM a. Similar to the request matrix R, the allocation metric A is an augmentation of all the allocation matrices (from all the service providers) as given below:

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \dots \\ a_u \end{bmatrix} = \begin{bmatrix} a_{11} a_{12} a_{13} \dots a_{1v} \\ a_{21} a_{22} a_{23} \dots a_{2v} \\ \dots \ \dots \ \dots \ \dots \ \dots \\ a_{u1} a_{u2} a_{u3} \dots a_{uv} \end{bmatrix}. \tag{6}$$

For every host, an allocation decision $\mathcal{D}$ is a possible allocation status from a set of all possibilities based on the resource requirement matrix $\mathcal{R}$

$$D = \left[A^1, A^2, A^3, \dots, A^h, \dots, A^p\right]. \tag{7}$$

The aim of the resource allocation problem, given the resource requirement matrix $\mathcal{R}$ and the capacity sets of hosts $\mathcal{C}$, is to calculate a reasonable mapping from resources to user's jobs. This is usually accomplished by the resource management system (or the scheduler) in a centralised, hierarchical or distributed fashion [16]. In our game, various players (resource, application and network providers) collectively arrive at a single decision, with a centralised bidding-based scheduling strategy as described in Section 4.1, that describes VM mappings/allocations which are collectively best for the whole MEC system; and also ensuring that the allocations are both energy and performance (runtime) optimised.

If we assume the above problem as a single-objective optimisation problem, then each service provider, individually, wants to maximise its utility through allocating its available resources such that: (i) energy consumption and workload performance (runtimes) are minimised (IaaS); (ii) applications' runtimes are minimised (SaaS); and (iii) network traffic is minimised (NaaS) – to ensure QoS and availability of the bandwidth. From a single-objective optimisation problem of a single service provider, the objective of all providers is given by:

$$max_A \left(\sum_{j \in N} u_j^n A_{i \in M}^n \right). \tag{8}$$

Moreover, for each VM or application i its allocation matrix to each service provider $N \in \{IaaS, SaaS, NaaS\}$ is given by $X_{iN} = \{0, 1\}$ such that $\sum X_{i,N} \leq 1$ i.e., each application is allocated exactly once to every provider. Once allocated, each user pay p (utility of server providers) for its application i to each service provider $n \in N$; where p is the sum of p_{in}^{IaaS}, p_{in}^{SaaS}, and p_{in}^{NaaS} that represents the cost owned by IaaS, SaaS, and NaaS, respectively. Therefore, the cost-based utility of the whole MEC system from a particular user with application i is given by $U_i = X_{ij}(p_{ij}^{IaaS} + p_{ij}^{SaaS} + p_{ij}^{NaaS})$. For m applications, the total utility (cost-based) of the MEC system is given by:

$$U = \sum_{k=1}^{m} U_i. \tag{9}$$

Therefore, the objective of the whole MEC system with respect to users' monetary costs is given by:

$$max(U). \tag{10}$$

The three constraints of the above optimisation problems are: (i) the allocated resource capacities cannot exceed the total capacities; (ii) each module is exactly placed on a single VM; and (iii) each user is allocated resources exactly once in their proximities.

3.3 The Optimisation Problem

We can formulate the above problem as a Min-Max multi-objective optimisation problem. Consider a MEC which comprises a datacentre, several edge locations (hosts), and users' jobs that run on a variety of VMs. Find the VM to host mapping, such that: (i) the cumulative energy consumed by the MEC is minimised; and (ii) the performance of the VM is maximised (or VM runtime is minimised). Similarly, regarding SaaS performance of various applications is ensured. Moreover, from networks point of view the available bandwidth (B) is maximised. We further assume performance as the VM runtime, the longer the VM runs the worse will be its performance and vice versa. Mathematically, we can integrate all these into Eq. (11)

$$\begin{aligned} &min\left(\sum_{i=1}^{N} \sum_{j=1}^{M} e_{ij} x_{ij} \sum_{i=1}^{N} t_{ij} x_{ij} + \sum_{k=1}^{M} t_k \right) \\ &+ \max \sum B + max_A \left(\sum_{j \in N} u_j^n A_{i \in M}^n \right) + max(U), \end{aligned} \tag{11}$$

where the mapping factor x_{ij} equals 1 if a particular VM i is mapped to host j. Furthermore, e_{ij} represents the energy consumed by host j when a VM i is run. Similarly, t_{ij} denotes the runtime of VM i over host j. Moreover, t_k

denotes the execution time of application k and B is the available bandwidth. The constraints of the above optimisation problems are

$$\sum_{i} x_{ik}^{n} \leq C_{k}^{n} \quad \& \quad \sum_{l \in N}^{m} x_{ik}^{l} \leq r_{ik}^{n}, \tag{12}$$

where the first one indicates that various resources allocated to VMs cannot exceed the hosts (providers) capacities while the second one ensures that only the required resources are offered to VMs. Note that, application runtimes, network performance, and energy consumption are inversely proportional to each other i.e., an increase in one value can decrease the other one's value. In the single-objective optimisation, the SaaS provider wants to maximise income through minimising users' monetary costs. However, lower costs may increase the number of customers and demand for services that may have negative performance impacts on users' workloads. Therefore, cost minimisation indirectly implies maximising SLA violations; SLA violations, then, depends on resources procured from their providers of network and IaaS (who, presumably want to achieve the same for the same reasons). In such scenarios, an assumption would be essentially needed in order to avoid the unnecessarily demand for IaaS resources by the SaaS providers (left-hand side of Eq. (12)). Furthermore, it is also possible that various providers are offering services to the highest paying customers first - in which case an additional constraint would be necessary over the allocation. The later one can also be assumed similar to native applications of IaaS providers - Gmail on Google cluster will get preference than running on Azure cluster [7]. This means that the MEC allocation problem is complex and cannot be easily solved with simple optimisation techniques.

An alternative approach is to account for individual player's objectives, separately. For example, the providers (IaaS, NaaS) deploying the MEC services aim to maximise their profits through selling more resources (using certain price models) and/or reducing energy consumption (paying less energy bills). Moreover, the SaaS has to account for: (i) gains from selling their applications; and (ii) the amount paid to providers (IaaS, NaaS), when deciding on their resource demands. The providers, first, set prices for their services. The SaaS providers decide, later on, their required computing and network resources for running user's application, being aware of the providers' prices. The utility function U of each player comprises: amount for selling; and cost incurred in providing resources [17]. We can divide the whole game into two sub-games, subject to various constraints, as discussed in Section 3.2:

1) every SaaS decides on the resource demand while maximising the expected utility, given all other SaaS' demands, i.e., strategies, as well as the MEC resource prices (Eq. (8)); and
2) the profit of each provider (IaaS, NaaS) is the revenue obtained from charging the SaaS for IaaS, NaaS and MEC resources (SaaS x pays a unit price p_x to each MEC provider) minus the incurred cost (Eq. (10));

where cost is a function of the resource demand, e.g., energy consumed, performance gains and throughput etc. Moreover, the SaaS providers may want to run users' applications in their proximity (the nearest available resources) in order to ensure expected levels of performance (in terms of low latency). Furthermore, IaaS, NaaS compete each other for providing resources and the SaaS compete for provisioning better services. The game solution guarantees that the resource price or allocation, chosen by various providers (IaaS, NaaS), increases their profits, such that SaaS providers achieve optimal performance for applications which also increases their utilities.

4 Proposed Solution

Game theory is largely used for analysing competitive interaction among various providers [18]. We model the above problem as a non-cooperative game where customers - SaaS, resource providers - IaaS, and access networks (or network providers - NaaS) act selfishly according to their own particular objectives [17], [19], as described in Section 3. As described in [12], multi-objective optimisation problems can be solved in two ways: (i) concurrently solve all objectives; and (ii) solve one objective first, and then make it a constraint on the next one. Moreover, various objectives can be combined into a single metric, and then solved as a single-objective problem [10]. To concurrently solve multi-objectives, Lagrange multipliers is one of the classical techniques to address such problems. The Lagrangian will converge all objectives to a single saddle point. The Hungarian method is also used to solve such problems, particularly, cooperative games where coalition can be formed among various service providers [19]. Since, we assume our game of non-cooperative nature [8], [20], and auction theory is a suitable tool to solve such kinds of games [17]. Therefore, we also solve the allocation problem with an auction theory using the bidding strategy. Our game theoretical approach is inspired by the previous work, as presented in [19].

We assume the whole MEC as a multi-agent system that consists of three different layers. This closely resembles hierarchical auction framework in the context of multi-agent systems [21], [22]. At the top-level, a global resource manager (broker) is responsible for assigning VM requests to a particular MEC. In the middle layer, a local manager is associated with each MEC that is responsible for assigning VM requests to appropriate computational resources (also known as agents). In the third layer, agents are responsible for running VMs. In our game, the local manager can submit bids for execution contracts to the global manager. Subsequently, the broker will select the winning MEC through a sealed-bid auction. The bids are computed (by a local manager) using a particular strategy at each server (for different providers) using various characteristics of the application and infrastructure. To effectively estimate the runtime for a contract bid, every local manager will ask all agents in its related MEC for estimates in order to create a runtimes matrix that comprises VM execution times. These estimates can be either: (a) computed using application characteristics and machine learning techniques; or (b) achieved through certain probabilistic methods [10]. The local manager then chooses which VMs it can execute and at what price. These details are then passed to the global resource manager for taking appropriate allocation decisions.

Algorithm 1. VM Placement Algorithm

Input: List of MECs (N), List of hosts in nth MEC (H_n), List of VM requests (V)
Output: Efficient VM placement

for *each player* $p \in N$ **do**
  for *each* $mec \in M$ **do**
    resource manager de-queues its job queue and announces that a VM_i is ready for bidding ;
    for *each agent* j **do**
      estimate runtime for VM_i ;
      tempraray en-queues VM_i into local job queue to check its possiblity of execution ;
      if VM_i *is executable on* j **then**
        $bid_j =$ compute bid using Eq. (13)
      else
        $bid_j = 0$ (since runtime = ∞)
      end if
    end for
    sort agents in ascending order of their bids ;
  end for
  take bid from network provider ;
  take bid from application provider ;
end for
sort all bids in ascending order with respect to the group value ;
convert all bids to a combinatorial bid using Eq. (16) ;
$agt \leftarrow$ the agent with the lowest bids (Hungarian) ;
allocate VM_i to agt ;
return *output*

The steps involved in resource allocation are described in Algorithm 1. The core module of the proposed allocation technique is the bidding strategy. Each service provider is associated with its own and a particular biding strategy, which is described later in Section 4.1. For every VM request, all the bids from various providers i.e., IaaS, SaaS, NaaS, are computed at each server (local resource manager) using the bidding strategy as described later in Section 4.1. The bids are, then, sorted in ascending orders of their utilities, and converted to a single (combinatorial) bid, given by Eq. (16), which is shared with the global resource manager. The global resource manager, then, chooses the local manager with the highest bid to run the VM. We can also use the Hungarian method to choose the optimal allocation strategy for a particular application (SaaS) [12]. Besides resource allocation, the global resource manager (broker) is responsible for consolidation workloads within the remote cloud and across several cloudlets. The consolidation process ensures that all cloudlets are balanced (with respect to workloads) and can be achieved using service migration technique, as described in Algorithm 2. Moreover, appropriate service migration techniques guarantee energy savings and workload expected levels of performance. Furthermore, VMs reallocation through service migration techniques, across various servers or MECs, can be modelled as a cooperative or semi-cooperative game in which various agents or local resource managers can help each other to run them, on appropriate resources [19]. In this paper, albeit we consider service migrations, however, they are modelled and considered a semi co-operative game, but, not a complete co-operative game. In the near future, we will consider service migrations a complete co-operative game; and will try to ensure the existence of the Nash equilibrium. The overheads and time complexities of both Algorithms 1 and 2 are elaborated in Section 4.2, particularly at large-scale in terms of number of players, hosts, and VM requests.

Algorithm 2. Service Migration Technique

Input: $optimise()$, M, T_v, T_l
Output:migration decision d

for *each cloudlet* $\in M$ **do**
  compute utilisation level of the cloudlet (T_e) ; compute network condition (T_c) ;
  if $T_e \geq T_v$ *or* $T_c \geq T_l$ **then**
    select module m from cloudlet ;
    choose cloudlet t as destination node ;
    $d \leftarrow true$;
  else
    continue with the **for** loop ;
  end if
end for
return m, t, d

4.1 The Bidding Strategy

The core component of the proposed technique is the bidding strategy that varies with various service providers involved within the MEC system. Each bid represents a possible VM schedule at certain cost of energy. For example, for IaaS with H total number of hosts the bid of each server h is computed using

$$bid_{h \in H} = \left(h_e - \frac{1}{h_e} \right) \times r_h, \tag{13}$$

where h_e represents the energy consumed and r_h denotes the expected runtime (therefore, performance with respect to SaaS) of a particular VM on host $h \in H$. The fraction $h_e - \frac{1}{h_e}$ converts h_e to a higher bid. The lowest bid demotes an optimal agent from both IaaS and SaaS perspective. For NaaS, we assume that the bandwidth is offered in sub-channels and is, largely, used for data transmission and communication. Important parameters, here, include the total distance between the IaaS and user, data size, transmission rate, execution delay, and link power consumption. These parameters should be considered in computing the NaaS bid. Furthermore, the broker is aware of the agent's distance from each user. The NaaS bid is given by:

$$bid_{b \in B} = D \times b_{cost}, \tag{14}$$

where D denotes the distance between the edge cloud and the agent, while b_{cost} is the channel (bandwidth) cost. We assume that NaaS offers various channels with numerous capacities at different costs just like EC2 instance types. The above bidding strategy can be converted to combinatorial bidding approach where all bids can be computed in one go [23]. In such scenarios, each VM request $\mathcal{R}$ can be represented as a 2-tuple i.e., $\mathcal{R}(\mathcal{C}, \mathcal{B})$ where $\mathcal{C}$ denotes the instance type (size) and $\mathcal{B}$ the required number of bandwidth channels. Note that, the required number of channels B_{ij} are computed using the transmission rate T_{rate}, as given by:

$$T_{rate} = B_{ij}\ log_2(1 + \frac{P_{ij}.h_{ij}}{N}), \tag{15}$$

where B_{ij} represents the bandwidth allocated to VM or user j, h_{ij} denotes the channel gain for user j at service provider i and P_{ij} is the transmission power of user j. Furthermore, N is the background noise [23]. Due to experimental simplification, we use, here, the combinatorial bidding approach, given by Eq. (16), in order to allocate IaaS, NaaS resources to different services i.e., various modules of the applications (SaaS)

$$\prod bid_{h \in H} bid_{b \in B}. \tag{16}$$

4.2 Time Complexity

This section briefly describes the average and best-case complexities for Algorithms 1 and 2. Note that, Eqs. (11) and (12) show that the allocation procedure is a two-dimensional knapsack problem; which is widely known as NP-hard. In the average case, Algorithm 1 will take $\mathcal{O}(n^2\ log(n))$ - where n^2 denote the number of players, hosts; and $log(n)$ is the time needed to compute the bid for VM. Note that, the number of players will not exceed a few i.e., constant numbers. For instance, in our case it is 3 that leads to an average complexity of $\mathcal{O}(n\ log(n))$. However, complexity would increase up to $\mathcal{O}(n^3)$ for large number of players, hosts and VM requests - if unluckily a VM cannot be placed. Similarly, Algorithm 2 will approximately take $\mathcal{O}(n\ log(n))$ - where n denotes the number of edges and $log(n)$ is the time needed to compute configuration states. The best case occurs at $\mathcal{O}(log(n))$ plus the time needed to complete all possible migrations.

5 Performance Evaluation

Resource allocation and consolidation can be seen as a kind of bin-packing problem by means of different sizes and costs of bins – where bins represent the MEC's resources (hosts) and items represent various applications for placement. Furthermore, the sizes of bins represent host's CPU, RAM, storage capacities and costs relate to hosts' energy consumption. Energy and performance efficient resource allocation can be assumed as a multi-objective optimisation problem with the objective(s) to reduce the number of used hosts – as fewer hosts, possibly, decrease the energy consumption. However, this statement may not hold for heterogeneous MECs [24]. Therefore, an alternative approach for heterogeneous systems is to minimise the sum of total bins costs instead of a number

$$min \sum_{k=1}^{n} C_{host_k}, \tag{17}$$

where C_{host} is the cost of host $k \in n$. We consider C as the product of energy (E), performance – execution time (T) and user monetary costs (U); that corresponds to all three parties in our game. Usually, bin-packing problems are solved using various heuristic approaches which may not guarantee optimal results, but they are considered fast enough to deal with, particularly, large problems [25]. It is possible to assume an analogous resource allocation problem as moving from a particular state of the datacentre to an ideal state – the one using the fewest hosts. We achieve a datacentre state through implementing various placement techniques (Random, first come first serve - FCFS, Cloud-Only, Edge-ward, Delay-priority, and Bidding-based, Game-theoretic - epcAware), with application packing then needing to ensure energy, performance and cost-efficiencies. The Random placement policy allocates an edge or datacentre server through pseudo-random number generator. If the randomly picked server has not enough capacity to run the workload, then, another one is selected; and the process is repeated until the workload is placed. Further, the FCFS policy is like a first fit approach that puts the workload from a queue on first suitable host out of n hosts - starting from 1 to n.

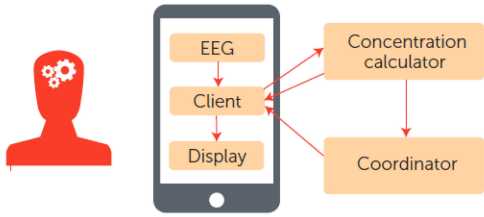

Fig. 2. EEGTBG application modules [9].

5.1 Modelling Applications

To demonstrate the efficiency of the proposed technique, we use two kinds of applications: (i) near real-time – where we model the well-known electroencephalography (EEG) tractor beam game (EEGTBG); and (ii) delay-tolerant – where a video surveillance/object tracking (VSOT) application is modelled. In respect of (i), several players attempt to collect items through concentrating on them - the better the concentration, the higher chances to collect more items. A true, online, real-time, experience can be observed through fast processing and low response times.

The EEGTBG application has 5 modules: (a) EEG sensor; (b) display; (c) client; (d) concentration calculator; and (e) coordinator. The EEG headset probes user concentration and communicates raw data to the client module. Subsequently, the client module transmits reliable data to the concentration calculator module, which calculates the user level of concentration. Furthermore, the computed concentration level is sent back to the client module, to update the game status (display) on the player's device. The coordinator module collects and distributes measured concentration among all players. As described in [9], the three modules i.e., sensor, display, and client are placed in the mobile device. However, the other two modules i.e., the concentration calculator and coordinator could be placed either in cloudlets or datacentre. Various modules of the EEGTBG application are shown in Fig. 2.

The VSOT application depends on a set of distributed cameras that could track movement, having six modules: (a) camera; (b) motion detector; (c) object detector; (d) object tracker; (e) user interface; and (f) pan, tilt, and zoom (PTZ) control. The camera streams video to the motion detector that, subsequently, filters the streamed video and transmits the video of interest, i.e., in which motion was detected, to the object detector module. The object detector recognises the moving objects and sends their identification and position data to the object tracker module. Sequentially, the object tracker calculates the required PTZ and sends the command

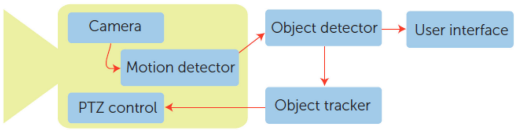

Fig. 3. VSOT application modules [9].

to PTZ control. We further deliberate that the two modules i.e., motion detector and PTZ control are permanently located within the camera, whereas the user interface runs in the cloud (datacentre). The other two modules i.e., the object detector and object tracker might be placed either in a cloudlet or datacentre.

The above applications can be set up in a MECs infrastructure to yield benefits of lower latency due to the use of edge devices. Moreover, VSOT can work reasonably well under datacentre-distance latencies (greater than 100 milliseconds) [9]. Alternatively, higher delays in EEGTBG application can impact the players' real-time observation, making the game weird as its playability might be damaged. We contemplate that both these applications belong to two diverse classes of applications types i.e., delay-sensitive and delay-tolerant, that could benefit from a MECs infrastructure. Various modules of the VSOT application are shown in Fig. 3.

5.2 Experimental Set-Up

We use the "iFogSim" simulator [5] to model and evaluate the performance of various resource placement policies because it: (i) supports the hierarchical composition of IaaS clouds, cloudlets and edge devices; (ii) runs on top of "CloudSim" [26] – the most widely used simulator in the cloud research community; and (iii) supports the measurement of application performance in terms of delays, response and execution times. We assume twelve instances of the VSOT application that run over cloudlet-2 and thirty-six instances of the EEGTBG application that run over cloudlets-1 and cloudlet-3, collectively. Initially, eighteen EEGTBG users are playing the game in proximity of cloudlet-1 location and the other eighteen players are in proximity of cloudlet-3 location. To emulate mobility and to assess performance degradation that may cause from poor resource allocation and service migrations, we move the EEGTBG players one by one to cloudlet-2. Due to low-latency requirements of EEGTBG application, we assume that a particular player in a cloudlet plays only against other players in the same cloudlet [9]. To simulate service mobility in the context of epcAware allocation and migration policies, appropriate migration decisions are, then, triggered using Algorithm 2. The migrations may happen either: (a) among hosts of the remote cloud – inter-datacentre; (b) among hosts of an individual edge cloud or across several edge clouds – inter-fog and intra-fog; and/or (c) among hosts of the edge and remote clouds – fog-datacentre. In respect of (a) and (b), migrations occur if hosts' utilisation levels drop below certain threshold value e.g., 20 percent. In theory, to avoid SLA violations, if server utilisation surpasses an upper threshold value i.e., 100 percent (over-subscription), some workloads could be migrated from it to the least utilised server. Nevertheless, we assume, here, that sensible ways of addressing VM density [10], given as constraint in Eq. (12), will not lead to overloading and SLA violations. In respect of (c), application modules are moved explicitly, as described later.

Every cloudlet has a processing capability (speed) of 3 heterogeneous servers, as shown in Table 4, that maps to the notion of millions of instructions per second (MIPS), for consistency with the iFogSim simulator, and is connected to the gateway (proxy server) through a link of bandwidth equal to 10 Mbps and latency of 4 ms (milliseconds). Moreover, the link between the gateway and the cloud has 10 Mbps bandwidth and 100 ms latency. We further assume that edge devices, such as mobile and camera, are connected to the cloudlets through a link of bandwidth equal to 10 Mbps and 2 ms latency. The maximum resource (CPU, RAM, and network bandwidth) requirements of each application's module are shown in Table 3. However, at scheduling time, prior to execution, each application's (module) resource requirements are unknown. Later on, resources could be predicted. Moreover, we assume that the application module workload (modelled as tuple) is dynamic that changes with time to time. Moreover, every tuple is assigned a particular task (i.e., fixed number of MIPS) which utilises the VM resources, using a normal distribution - most likely resource usage in Google cluster, that could create variations in runtimes.

Similarly, we account for resource contention or interference on various servers, that could, possibly, degrade applications performance. The cloud consists of 100 heterogeneous hosts that correspond to three different CPU platforms, as shown in Table 4. The idle (P_{idle}) and maximum power consumption (P_{max}) of hosts were taken from the SPECpower benchmarks.[5] To maintain consistency with the iFogSim simulator, speeds of the hosts are transformed to MIPS. Each host can run several VMs where each core of a particular host corresponds to a single vCPU of a VM instance. This implementational simplification is justified due to the facts that: (i) the dataset, used in this paper, comes from a single-core VMs service provider; and (ii) the notion of VM density [10]. We, further, assume that every module of the fog application runs in a VM instance and the speed of VM is exactly equal to the

TABLE 3
CPU, RAM, and Network Bandwidth (BW) Requirements (in MIPS, MB, and Mbps, Respectively) for Both Applications and Their Various Modules [9]

	VSOT				EEGTBG		
	Object detector	Motion detector	Object tracker	User interface	Client	Concentration calculator	Coordinator
CPU	550	300	300	200	200	350	100
RAM	30	25	25	20	50	60	30
BW	100	100	100	400	500	200	200

TABLE 4
Different Characteristics of Various Hosts Used in the Simulated MEC System [P_{idle} and P_{max} Denote the Host' Idle (0 percent Utilised) and Maximum (100 percent Utilised) Power Consumption, Respectively] - Performance Parameters of These Hosts are Described in [10]

CPU model	Speed (MHz)	No of cores	Memory (GB)	P_{idle} (Wh)	P_{max} (Wh)	Total amount of servers	
E5430	2830	8	16	166	265	34	
E5507	2533	8	8	67	218	33	100
E5645	2400	12	16	63.1	200	33	

CPU requirement of each module, as shown in Table 3. We also account for migration costs [24], resource heterogeneity (in terms of CPU architecture) and resource contention due to co-location i.e., when similar modules of the same applications are placed on the same host and compete for the same resources [27]. The resource contention and CPU heterogeneity parameters for various hosts, as shown in Table 4, were taken from our previous work [10]. Each migration degrades the VM performance by 10 percent, as investigated in [28]. Moreover, the energy cost of each VM migration E_m is modelled using Eq. (18)

$$E_m = 0.512 \times VM_{data} + 20.165, \tag{18}$$

where VM_{data} denotes the amount of memory (measured in MBs) allocated to the VM plus memory pages dirtied during the migration process [10]. Note that, speed of hosts and VMs are transformed into the notion of MIPS in order to keep consistency with the CloudSim and iFogSim simulators. Moreover, each service requirements (CPU) are according to the default setting of iFogSim (number of MIPS). We assume that each module of various application utilises its provisioned VM resources in whole. Furthermore, each provider's utility is computed using a particular cost model. In our experiments, there are 12 servers in cloud, 1 proxy server, and 3 servers per cloudlet. Each service is placed on the most energy, performance and cost-efficient host, using Algorithm 1, in such a way that resource requirements of each service are ensured [29], [30].

5.3 Evaluation Metrics

The metrics used to evaluate the energy, performance and cost-efficiency of the proposed resource allocation and migration policies are: (i) total energy consumed (E) measured in KWh; (ii) execution time (T) measured in seconds – as application's performance is inversely proportional to T; (iii) delay; (D) among various modules which is measured in milliseconds; (iv) the total number of migrations – fewer migrations may ensure higher performance levels and lower energy consumption; and (v) provider's utility. Note that, the providers' utilities actually describe the money (in US dollars - $) for energy consumption bills (IaaS), resources being sold (NaaS, IaaS) and instances being purchased (SaaS, IaaS). We assume the energy cost at the rate of 0.08$ per KWh, VM instances and bandwidth costs at 0.07$ and 0.02$ per hour, respectively [10]. These values reflect the real market prices in the United States.[6] We also assume that VMs running in cloudlets incurs an additional cost of approximately 20 percent greater than the cloud VMs [29]. Where needed, statistical tests (*t-test*) were performed to show significant differences among various techniques.

5. https://www.spec.org/power_ssj2008/
6. https://www.eia.gov/electricity/monthly/

TABLE 5
Experimental Results - Energy is the Sum of Both Dataceneter and Cloudlets Usage and R Means Total Execution Time [Without Migrations]

Allocation Policy	Providers utility ($)			Energy (KWh)	R (hours)
	IaaS	SaaS	NaaS		
Cloud only					
Random	279.3	1.82	0.48	3,491.7	26.12
FCFS	262.8	1.75	0.5	3,285.3	24.98
Delay-priority	239.7	1.68	0.5	2,996.0	23.93
epcAware-NC	225.8	1.68	0.52	2,822.1	23.87
epcAware-SC	223.0	1.68	0.53	2,787.9	23.51
Edge-ward					
Random	270.2	1.96	0.44	3,377.0	27.95
FCFS	239.3	1.89	0.47	2,991.5	26.69
Delay-priority	231.9	1.75	0.5	2,898.9	25.07
epcAware-NC	216.1	1.68	0.52	2,701.0	23.87
epcAware-SC	215.4	1.61	0.53	2,692.5	23.32

5.4 Experimental Results and Discussion

Table 5 describes experimental results for various resource placement, consolidation and management policies. The results show that various approaches to resource placement and placement methodologies (cloud-only, edge-ward) offer variations in energy consumption, and workload performance. Largely, our findings are consistent with previous outcomes [9] that the edge-ward approach is approximately 3.24 to 8.94 percent energy and ∼0.81% performance efficient than the cloud-only placement method. However, if resource contention (due to co-located VMs) and platform heterogeneities (due to CPU architectures) are considered at the cloudlets, then the edge-ward approach cannot ensure performance benefits. Therefore, performance gains are obtainable if certain performance-aware placement policies such as "epcAware-NC" or "epcAware-SC" are taken into account; where the words, NC and SC, indicate whether the problem is solved using a non-cooperative or a semi-cooperative game, respectively.

Migrations which may happen due to cooperation among various players (service providers) can increase energy, performance, therefore costs saving. However, if migration costs, resource heterogeneities and contention are considered, then migrations could potentially degrade applications' performance up to -10.97 percent. The figures, as described in previous paragraph, can further be improved i.e., ∼11.95% energy savings and ∼3.56% performance gains, if certain epcAware service migration techniques are considered, as shown in Table 6. The total number of migrations, as shown in Fig. 4, affect the overall performance degradation – a higher number of migrations potentially lead to applications lower performance. The migrations may happen either inter-fog nodes or/and intra-fog platform. Moreover, inter-datacentre migration and fog-datacentre migrations are also possible. Our investigation suggests that, in MEC, migrations may, largely, happen inter-datacentre which is justifiable due to the

TABLE 6
Experimental Results - Energy is the Sum of Both Dataceneter and Cloudlets Usage and R Means Total Execution Time [With Migrations]

Allocation Policy	Providers utility ($)			Energy (KWh)	R (hours)
	IaaS	SaaS	NaaS		
	Edge-ward				
Random	237.9	1.96	0.44	2,973.5	28.03
FCFS	219.6	1.89	0.45	2,745.2	27.4
Delay-priority	224.3	1.96	0.44	2,803.6	27.82
epcAware-NC	215.8	1.61	0.54	2,697.7	23.02
epcAware-SC	215.4	1.61	0.54	2,691.8	23.01

TABLE 7
Percentage Improvements of Using Game-Theoretic Methods to Placement [Base is the Delay-Priority Approach, E and P Denote Energy and Performance (%), Respectively]

Policy	Base		epcAware-NC		epcAware-SC	
	E	P	E	P	E	P
cloud-only	-	-	5.8	0.25	6.94	1.76
edge-ward	-	-	6.83	4.79	7.12	6.98
	-	-	3.78	17.25	3.98	17.29

increased number of nodes in datacentre. Moreover, since the Random policy puts modules scattered, therefore creating maximum opportunities for migrations, as shown in Fig. 4. However, if mobility is considered [9], then moving application modules across several cloudlets or between cloudlets and datacentre would be essential. This also demonstrates that migrations can be reduced up to 52.8 percent if providers cooperate and schedule workloads on appropriate resources.

Table 7 shows the percentage of improvements in energy efficiency and performance gains when resource allocation problem is modelled, using a game approach, among different service providers with conflicting goals. Our findings demonstrate that cooperative-based game approaches for resource allocation ensure higher efficiencies. To study the impact of long-running monitoring services and short-running game applications, we changed the runtimes of applications in the above experiment, accordingly [10]. We observed that monitoring applications (VSOT) modules are more cost-effective to migrate than game application (EEGTBG) modules in terms of the total number of migrations (reduced), and performance loss (reduced). Longer runtimes could guarantee recoverability of the migration costs [24] through subsequent running over the target and, therefore, cost-savings. This creates further gap for investigation of placing and running modules of monitoring services in VMs and game modules in containers. This will also ensure reduction in application migration times since container images are smaller than VM images.

The scheduling and migration decisions in fogs also affect the total network use, as shown in Fig. 5. For example, migrations increase the utilisation levels, that could be up to 9.65 percent, of the used bandwidth irrespective of the network capacity. Moreover, if network provider allocates their channels or bandwidth capacities in such a way that each application or user gets exactly what is needed for quality of service (QoS). Then the allocated bandwidth can be decreased, but, utilised more. When placement and migration decisions are taken based on the cooperation among various service providers, then approximately 3.83 to 12.86 percent network capacity could be saved which translate to cost-saving from NaaS perfective.

Table 8 shows average delays (measured in seconds) for various applications and resource placement and migration policies. The delay actually represents the time needed to complete a particular task (tuple). For example, in the case of the VSOT application, the delay represents the time between a sensor notices an object and PTZ controller to identify the object. The edge-ward approach is somehow offering lower delays than the remote cloud-only technique - which is justifiable due to short distances. Moreover, the application delay is affected through increasing the number of users (or application modules), network usage (available bandwidth), and the total number of migrations, which may produce contention if same modules are placed on same resources. We observed that the classical FCFS policy offers the least application delays since it prefers to put application modules on the remote cloud. Moreover, the delay-priority policy [9] offer acceptable delays, however, for a large number of users (concurrent application modules) it cannot guarantee consistent lower delays. Albeit, our proposed policies can ensure lower delays if migrations are not taken into account. However, since the proposed algorithms look for opportunities to reduce network (bandwidth) provisioning – the NaaS objective; therefore, combined with migrations, improvement in delays is not trivial and this needs further work.

We observed that IaaS energy costs are largely affected by the heterogeneity of resources and degree of co-location. If

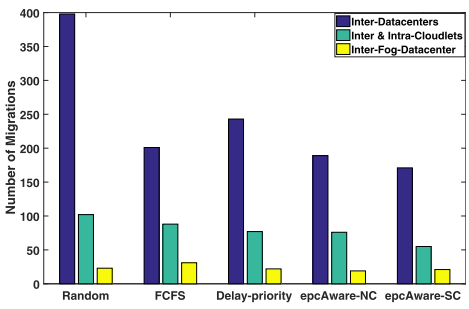

Fig. 4. Total number of migrations [including inter-datacentre, inter and intra-fog, and fog-datacentre].

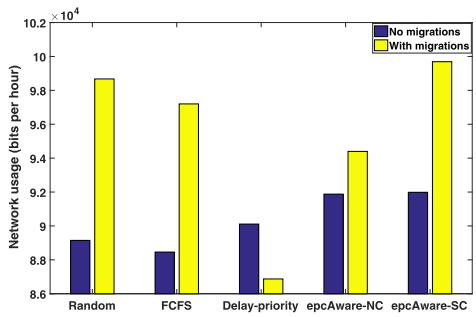

Fig. 5. Network usage for various policies [larger utilisation means more profit due to more number of customers; therefore, higher values are better than the lower values].

TABLE 8
Average Application Delays (in Seconds) the ± Denotes Standard Deviation Over the Average [Minimum Values are 'Best']

Policy	epcAware-NC		epcAware-SC	
	EEGTBG	VSOT	EEGTBG	VSOT
cloud-only	18.7±1.1	17.8±1.3	23.8±1.9	20.1±2.2
edge-ward	13.6±2.4	15.1±2.1	16.7±2.3	15.9±1.8
	30.0±3.3	29.9±2.8	24.6±2.6	23.2±2.6

TABLE 9
Generalisation of Experimental Results With Migrations [± Denotes Standard Deviation After the Mean Values]

Alloc. Policy	Providers utility ($) IaaS	SaaS	NaaS	Energy (KWh)	R (hours)
Edge-ward (200 servers - E5507, E5645, E5-2651), 150 Apps					
P_1	787.12	4.48	8.72	9,839.0±102	63.78±1.2
P_2	770.07	4.34	8.76	9,625.9±88.4	61.99±1.3
P_3	762.51	4.06	8.83	9,531.3±63.8	58.30±3.5
Edge-ward (800 servers - E5430, E5-2650, E5-2670), 300 Apps					
P_1	4,294.8	6.3	18.03	58,280.5±51.8	98.32±2.7
P_2	4,614.9	6.3	18.2	57,677.2±119	89.87±4.8
P_3	4,622.8	6.16	18.24	53,684.5±195	89.88±5.9

schedulers are made aware of the infrastructure heterogeneity, performance of the co-located applications; then it is possible to reduce up to 13.48 percent energy costs. Due to existing trade-off between energy consumption and workload performance, lower performance increases customers costs, utilities of IaaS and NaaS; however, it could decrease energy efficiency.

5.5 Results Generalisation

To demonstrate the applicability of the epcAware approach in a real MEC test-bed, we run several experiments with different MEC set-up, hosts, previous assumptions and application types. Various hosts, their characteristics and workload types were selected from [31]. We validated epcAware technique (P_1) to check whether generalisation of our findings is correct or not. The experimental details and results are shown in Table 9. On average, our approach outperforms other methods; however, we noticed some overlap between epcAware-NC (P_2) and epcAware-SC (P_3). Using the *t-test* analysis (95 percent confidence interval i.e., $p = 0.05$), we observed that both methods are significantly different; where P_3 largely outperforms P_2. As the number of applications grows, more migrations occur leading to rise in IaaS energy usage i.e., up to 6.85 percent at cost of approximately 0.01 percent performance loss. Albeit, runtimes of P_3 vary for 300 applications; that leads to lower average performance. However, we observed significant differences between P_2 and P_3 using the *t-test* analysis – the least values for p (*t-critical* = 2.374) [10].

6 Related Work

Gupta *et al.* [5] proposed two application placement policies: (i) cloud-only that places all modules of the application in cloud datacentre; and (ii) edge-ward that favours running modules of the application on edge/fog devices. However, in the case of (ii), if the allocation of a fog device is not appropriate, then resources from other fog devices or cloud datacentres could be provisioned. Moreover, a simulator "iFogSim" is developed to simulate mobile edge cloud platforms. Empirical evaluation of both allocation policies for two real-time applications suggests that the edge-ward policy significantly improves the application's performance and reduces the network traffic. Bittencourt *et al.* [9] investigated three different allocation policies; (i) concurrent – the requests at cloudlet are served in the same cloudlet; (ii) FCFS – requests are served in the order of their arrival (edge-ward); and (iii) delay-priority – the lowest delay applications are scheduled first and the remaining applications are placed according to edge-ward placement. Moreover, application (modules) migration is supported; and if a particular module is moved to a target device, all other modules are also moved. Their investigation suggests that if an application performance is the worst on a fog device, probably, due to a greater number of connected users, its migration to the cloud datacentre is performance efficient. Moreover, network traffic is reduced.

Guerrero *et al.* [32] suggested a decentralised placement policy that runs popular and most widely used applications closer to the end-users (closeness or proximity). The popularity of the applications is computed through statistical distributions of their access rate i.e., service request rate (denoted by λ). For each device, the algorithm analyses λ of every service and migrate the lower requested services to upper devices (in edge-ward fashion). Their experimental evaluation suggests that such a placement method significantly improves the network usage and service latency of the most widely used and popular applications. However, the existing trade-off between the network usage and applications' latencies (delays) is not investigated.

Taneja *et al.* [33] suggested an application placement policy that puts various modules of the fog application on suitable resources. The proposed policy first sorts all the network nodes and application modules in ascending order of their capacities and requirements, respectively. Then it searches for most efficient nodes that could meet the module requirements; and the module is run. Their research suggests that the proposed allocation scheme could significantly reduce the network usage and improve application latency as compared to traditional cloud allocation policy. Moreover, overall energy consumption varies with respect to the number of fog devices in the infrastructure. For a small number of fog devices, traditional cloud allocation policies could beat fog placement. However, energy consumption of the proposed placement strategy could be optimised for a large number of IoT and fog devices.

Skarlat *et al.* [34] have modelled the service placement problem in fog as integer linear programming – find the optimal mapping between services (applications) and computational resources; to optimise fog utilisation while meeting QoS requirement, particularly, deadlines. Moreover, services are prioritised based on their deadlines. When a service request is received at a particular node, the application is placed on it; and if cannot be accommodated there, then it is placed either: (i) in the same fog colony [34]; (ii) on the closest neighbouring nodes (fog colony); or (iii) on the cloud (in an edge-ward fashion). The experimental results show that the proposed method

could utilise the fog landscape for approximately 70 percent of services and could reduce the execution cost up to 35 percent as compared to execution in the cloud only approach.

Brogi *et al.* [35], [36] have proposed a software prototype "FogTorch" that could deploy applications over the fog infrastructure such that all hardware, software and QoS requirements (i.e., bandwidth, latency) are fulfilled. A smart agriculture application has been modelled [36]; and a 3-layer fog infrastructure has been suggested for its deployment. Empirical evaluation of the proposed prototype shows that it could successfully return all eligible deployments (resource provision) for several optimisation scenarios, requirements and needs. Tuli *et al.* [30] proposed HealthFog, a framework which integrates deep learning methods within the fog infrastructure to run health monitoring system i.e., heart disease analysis using IoT devices.

Plachy *et al.* [37] have discussed dynamic service placement in mobile edge clouds. Mobility of a fog user is predicted, and, instead of migrating the application (running inside a virtual machine), an alternative network path is selected to connect user at the target. Experimental results show a minimum 10 percent improvement in response time (delay). Furthermore, a service placement technique, based on predicted future costs of its placement, is also presented. To model user mobility, service migration between cloudlets, and, cloud datacentre is also investigated in [38]. Several prediction models are suggested to estimate the cost of running and migrating a particular service. Both, off-line and on-line service placement problems are solved using various placement algorithms.

Various techniques for migrating (live) services in fog infrastructure are proposed and evaluated in [39]. To minimise the migration time of an application, a three-layer architecture is presented; where an idle copy of the application is stored at an intermediate layer. Before migrating the memory states of the application from any source, the application idle copy is migrated first. Later on, only memory pages are copied, that could significantly reduce the migration time. Moreover, a comparison of VM and container-based service migration is also elaborated. Mahmud *et al.* [29] proposed a profit-aware service placement policy for resource provisioning in fog infrastructure. Their outcomes suggest that cloudlet instances are approximately 20 percent expensive than the cloud instances. However, service migrations and user mobility are not considered.

Urgaonkar *et al.* [40] have also discussed various strategies for migrating services, in mobile edge clouds, in order to minimise operational costs. The "never migrate" policy places each application at a particular cloudlet with no reconfiguration that may happen due to user mobility. User requests are always routed to the original location of the application. In the "always migrate" policy, user requests are routed to the closest cloudlet with reconfiguration, in such a way that queues with the largest backlogs are served first. Moreover, if the arrival rate of requests at a particular cloudlet exceed its capacity, they are routed towards the cloud (in an edge-ward fashion). The "myopic" policy accounts for reconfiguration, transmission and routing costs; and takes appropriate migration decision such that the sum of these costs could be minimised. The work presented in [40], assume the user mobility as a Markovian process that is solved using Markov Decision Process (MDP) technique. However, as mentioned in [41], users' mobility cannot be accurately predicted. A mobility-aware dynamic service placement technique (heuristic based on the Markov approximation) is presented in [41], that accounts for: (i) costs of migrating services; and (ii) the trade-off between performance and operational cost.

Gillam *et al.* [42] have also discussed VMs, containers and code/functions (Function as a Service – FaaS) in order to explore edge computing for on-vehicle and off-vehicle computation that will be needed to support connected and automated/autonomous driving. To minimise end-to-end latency, the authors suggest that it is essential that computation should be more local to vehicles. However, vehicle mobility will create opportunities for application/code migration, and with notable exception of [39], it is rarely discussed. Zafari *et al.* [7] modelled the resource allocation problem in MECs and, in particular, when various edge cloud service providers share their extra resources with each other. Moreover, various service providers have their own utility functions which they want to improve through coalition. To solve this multi-objective optimisation problem, a cooperative game theoretic approach is proposed which suggests that service providers can increase their utilities through resource sharing. The same idea has been implemented in clouds [11], where datacentre resources are shared among various IaaS providers that have their own objectives. In both cases, it is ensured that each provider allocates only required services to their native users first. After that, free and unused resources, if available, are shared with other IaaS providers. Compared to our approach, the players are always IaaS providers (therefore, same objectives), or multiple but solved individually, while we account for various kinds of players i.e., IaaS, SaaS and NaaS (therefore, multiple objectives which often conflict with each other).

Ahmed *et al.* [43] used game theory for scheduling tasks in a multi-core system such that energy consumption is minimised, and performance is ensured. Similarly, Khan *et al.* [19] studied various game-theoretical methods for resource allocation in multi-agent computational grids. The work in [12] extends [19] in order to optimise grid energy consumption and workload performance through game-based resource allocation techniques. All these proposals consider, only, a single service provider; and have ignored allocation when services are offered by various providers, in particular, having conflicting goals. Moreover, service migrations are not taken into account. Li *et al.* [8] formulated the task offloading problem in MEC system as a non-cooperative game; where each player can selfishly minimise his own pay-off through using an appropriate strategy. Moreover, they proposed various algorithms to find the Nash equilibrium. Table 10 describes summary of the related work. We believe, information in this table will help our readers to quickly identify gaps for further research, investigation and improvement.

7 Conclusions and Future Work

MEC is an evolving paradigm that combines computational, storage and communication (network) capacities at the edge of the network through datacentres, in an elastic infrastructure. MECs have potentials to accommodate and run various application types such as: (i) throughput-oriented that need huge computational capacity and network bandwidth; and (ii) latency-oriented applications that need low latency communication and computation in user' proximity. Usually, the

TABLE 10
Summary of the Related Work, Closest to epcAware, With Respect to Various Evaluation Criteria

Parameters		Related Work											
		[5]	[9]	[32]	[33]	[34]	[35]	[30]	[37]	[40]	[8]	[29]	epcAware
Platform	Cloud	✓	✓	✓	✓				✓			✓	✓
	Fog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	IoT	✓	✓		✓			✓		✓			✓
Provider	IaaS	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	SaaS					✓						✓	✓
	NaaS				✓			✓	✓	✓	✓		✓
Evaluation metrics	Energy	✓			✓			✓					✓
	Performance	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	Migration cost									✓			✓
	User costs					✓					✓	✓	✓
	Co-location		✓										✓
	Deadline					✓							
Placement	Single party	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
	Bi parties										✓		
	Multi parties												✓
Management policy	Allocation	✓			✓	✓	✓	✓		✓	✓	✓	✓
	Migration									✓			✓
	Allocation+Migration		✓	✓					✓				✓
	Game-theoretic										✓		✓

computational capacity in edge locations and the wireless access are managed either distinctly or by a single player. Nevertheless, bringing the full potential of MECs entails that edge locations, IaaS, and wireless networks are to be managed in concert. In this paper, we modelled the resource allocation problem in MECs as a non-cooperative game. Further, resource reallocation decisions were taken into account at the IaaS layer and modelled as semi-cooperative game. We, then, proposed a bidding-based resource allocation and consolidation technique "epcAware" in order to effectively place various applications across various service providers. Empirical evaluation of the proposed algorithm suggests that it is able to manage resources energy, performance and, therefore, cost-efficiently. Furthermore, our proposed approach could reduce up to 11.95 percent energy, and approximately 17.86 percent user costs at non-significant loss in performance. Moreover, IaaS can reduce up to 20.27 percent energy bills and NaaS can increase their costs savings up to 18.52 percent as compared to other methods.

This emerging technology has still a research gap for identifying the locations where such small datacentres should be installed. For example, potentially these can be deployed in universities, hospitals, mobile base stations and/or shopping malls – where their operation and management is more affective or possible. Once installed, what kind of services should they host and how the available resources should be allocated to customers' applications. Similarly, if mobility is involved, then how the available resources or running services should be managed or migrated among small datacentres (cloudlets), and geographic areas. The aim of our further research would be to investigate and answer these kinds of questions from a geographic area perfective – while accounting for resource and energy costs variations. We believe energy is often hard to quantify in MECs and needs further investigation. Albeit, iFogSim-based simulators make use of it, but in practice and reality assessing benefit in terms of energy is often very challenging. However, execution time and cost of resource use is much easier to quantify and measure. The first objective would be to investigate, where cloudlets should be deployed such that service agility and performance is guaranteed. Our second objective would be to study the resource allocation and placement policies in order to decrease energy and network usage, while performance and user costs are not affected. The third objective would be to propose a novel management framework, in order to efficiently manage resources using hierarchical or distributed schedulers instead of a single scheduler. Moreover, in the future, we will reconsider the aforementioned problem for further investigation (co-operative game), and mathematical proof for finding a Nash equilibrium [21].

The supposition that network is limited by the speed of the network interface at each device could be non-essentially tolerable, albeit bottlenecks can often exist elsewhere [14], [15]. More specifically, for NaaS providers the topology of the network may be very important as well, as building virtual networks, virtual network functions (VNFs), and running the SDN (software defined network) controls are complex tasks. We believe this modelling would be a different research topic for itself; and, therefore, leave it for future investigation and research.

ACKNOWLEDGMENTS

This work was supported, in part, by the Abdul Wali Khan University Mardan, Pakistan and, in part, by an Australian Research Council (ARC) Discovery Project. The authors are thankful to Dr. Luiz F. Bittencourt, from the University of Campinas (UNICAMP), Brazil, for helping us to set-up essential simulation platform. They also thank Prof. Erik Elmorth from Umea University, Sweden for his comments on the research problem. Hashim Ali contributed equally to this work.

REFERENCES

[1] A. Lebre, J. Pastor, A. Simonet, and F. Desprez, "Revising OpenStack to operate fog/edge computing infrastructures," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2017, pp. 138–148.

[2] M. Abderrahim, M. Ouzzif, K. Guillouard, J. Francois, and A. Lèbre, "A holistic monitoring service for fog/edge infrastructures: A foresight study," in *Proc. IEEE 5th Int. Conf. Future Internet Things Cloud*, 2017, pp. 337–344.

[3] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[4] M. Ali *et al.*, "Edge enhanced deep learning system for large-scale video stream analytics," in *Proc. IEEE 2nd Int. Conf. Fog Edge Comput.*, 2018, pp. 1–10.

[5] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Softw.: Practice Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[6] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format+ schema," Google Inc., Mountain View, CA, USA, Tech. Rep., 2011. [Online]. Available: https://github.com/google/cluster-data

[7] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, "A game-theoretic approach to multi-objective resource sharing and allocation in mobile edge," in *Proc. Technol. Wireless Edge Workshop*, 2018, pp. 9–13.

[8] K. Li, "A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2018.2868655.

[9] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar./Apr. 2017.

[10] M. Zakarya and L. Gillam, "Energy and performance aware resource management in heterogeneous cloud datacenters," PhD dissertation, University of Surrey, 2017. [Online]. Available: http://epubs.surrey.ac.uk/841959/

[11] F. Zafari, K. K. Leung, D. Towsley, P. Basu, and A. Swami, "A game-theoretic framework for resource sharing in clouds," 2019, *arXiv: 1904.00820*. [Online]. Available: https://arxiv.org/abs/1904.00820

[12] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 346–360, Mar. 2009.

[13] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.

[14] P. Costa, M. Migliavacca, P. Pietzuch, and A. L. Wolf, "NaaS: Network-as-a-service in the cloud," in *Proc. 2nd USENIX Workshop Hot Topics Manage. Internet Cloud Enterprise Netw. Serv.*, 2012. [Online]. Available: https://dl.acm.org/doi/proceedings/10.5555/2228283

[15] J. Moura and D. Hutchison, "Game theory for multi-access edge computing: Survey, use cases, and future trends," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 260–288, First Quarter 2019.

[16] A. A. Khan, M. Zakarya, and R. Khan, "H^2 – A hybrid heterogeneity aware resource orchestrator for cloud platforms," *IEEE Syst. J.*, vol. 13, no. 4, pp. 3873–3876, Dec. 2019.

[17] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[18] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 52–57, Aug. 2017.

[19] S. U. Khan and I. Ahmad, "Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation," in *Proc. 20th Int. Parallel Distrib. Process. Symp.*, 2006, pp. 10 pp.-.

[20] W. Cai, F. Chi, X. Wang, and V. C. Leung, "Toward multiplayer cooperative cloud gaming," *IEEE Cloud Comput.*, vol. 5, no. 5, pp. 70–80, Sep./Oct. 2018.

[21] P. S. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory," *IEEE Syst. J.*, vol. 10, no. 2, pp. 637–648, Jun. 2016.

[22] K. Metwally, A. Jarray, and A. Karmouch, "A distributed auction-based framework for scalable IaaS provisioning in geo-data centers," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2018.2808531.

[23] H. Zhang, F. Guo, H. Ji, and C. Zhu, "Combinational auction-based service provider selection in mobile edge computing networks," *IEEE Access*, vol. 5, pp. 13 455–13 464, 2017.

[24] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *Proc. Int. Conf. Econ. Grids Clouds Syst. Serv.*, 2016, pp. 175–190.

[25] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1027–1034, 2011.

[26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw.: Pract. Exper.*, vol. 41, no. 1, pp. 23–50, 2011.

[27] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2470–2483, Aug. 2016.

[28] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.

[29] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog–cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 135, pp. 177–190, 2020.

[30] S. Tuli *et al.*, "HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," *Future Gener. Comput. Syst.*, vol. 104, pp. 187–200, 2020.

[31] A. A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, and O. Rana, "An energy and performance aware consolidation technique for containerized datacenters," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2920914.

[32] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 6, pp. 2435–2452, 2019.

[33] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage.*, 2017, pp. 1222–1228.

[34] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput.*, 2017, pp. 89–96.

[35] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[36] A. Brogi, S. Forti, and A. Ibrahim, "How to best deploy your fog applications, probably," in *Proc. IEEE 1st Int. Conf. Fog Edge Comput.*, 2017, pp. 105–114.

[37] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2016, pp. 1–6.

[38] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Aug. 2017.

[39] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[40] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Perform. Eval.*, vol. 91, pp. 205–228, 2015.

[41] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018, doi: 10.1109/JSAC.2018.2869954.

[42] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2018, pp. 148–153.

[43] I. Ahmad, S. Ranka, and S. U. Khan, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," in *Proc. 22nd IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–6.