# Profit-aware Resource Management for Edge Computing Systems

Cosimo Anglano
Computer Science Institute, DiSIT,
University of Piemonte Orientale
Italy
cosimo.anglano@uniupo.it

Massimo Canonico
Computer Science Institute, DiSIT,
University of Piemonte Orientale
Italy
massimo.canonico@uniupo.it

Marco Guazzone
Computer Science Institute, DiSIT,
University of Piemonte Orientale
Italy
marco.guazzone@uniupo.it

## ABSTRACT

Edge Computing (EC) represents the most promising solution to the real-time or near-real-time processing needs of the data generated by Internet of Things devices. The emergence of Edge Infrastructure Providers (EIPs) will bring the EC benefits to those enterprises that cannot afford to purchase, deploy, and manage their own edge infrastructures. The main goal of EIPs will be that of maximizing their profit, i.e. the difference of the revenues they make to host applications, and the cost they incur to run the infrastructure plus the penalty they have to pay when QoS requirements of hosted applications are not met. To maximize profit, an EIP must strike a balance between the above two factors.

In this paper we present the *Online Profit Maximization* (OPM) algorithm, an approximation algorithm that aims at increasing the profit of an EIP without a priori knowledge. We assess the performance of OPM by simulating its behavior for a variety of realistic scenarios, in which data are generated by a population of moving users, and by comparing the results it yields against those attained by an *oracle* (i.e., an unrealistic algorithm able to always make optimal decisions) and by a state-of-the-art alternative. Our results indicate that OPM is able to achieve results that are always within 1% of the optimal ones, and that always outperforms the alternative solution.

## CCS CONCEPTS

• **Networks → Cloud computing**; • **Computer systems organization → Cloud computing**;

## KEYWORDS

Edge computing, Profit maximization, Server consolidation, QoS

**ACM Reference Format:**
Cosimo Anglano, Massimo Canonico, and Marco Guazzone. 2018. Profit-aware Resource Management for Edge Computing Systems. In *EdgeSys '18: International Workshop on Edge Systems, Analytics and Networking, June 10–15, 2018, Munich, Germany*. ACM, New York, NY, USA, 6 pages.
https://doi.org/10.1145/3213344.3213349


*EdgeSys '18, June 10–15, 2018, Munich, Germany*

ACM ISBN 978-1-4503-5837-8/18/06…$15.00
https://doi.org/10.1145/3213344.3213349

## 1 INTRODUCTION

Large-scale *Internet of Things* (IoTs) services are becoming more and more commonplace. These services are based on the collection, storage, and processing of very large volumes of data, generated by an extremely large number of devices (e.g., sensors, smart personal devices, vehicles) which are located at the edge of the network and operating on a 24/7 basis.

These data often require real-time or near real-time processing (e.g., for augmented reality services or smart traffic light systems) [13]. Hence, the inherent high latency of the core network makes cloud computing unsuitable to meet the above requirements. Conversely, *Edge Computing* (EC) [15] whereby compute, storage, and network resources are provided by *Edge Nodes* (ENs) that are placed at the edge of the network, in close proximity to where data are generated, represents a very promising solution to the processing needs of these data.

To profitably implement the EC paradigm, a very large number of ENs must be typically placed at the end of the network to reduce latency [29], especially when several geographically dispersed areas need to be covered. Therefore, latency reduction would be prohibitively costly for an enterprise that had to purchase and deploy its own ENs. Furthermore, these costs would be hardly amortized, since the capacity of these ENs would be used only in part for most of the time, because of the variations in volume, variety, and velocity of generated data [13], especially when data are generated by a population of moving users.

The emergence of *Edge Infrastructure Providers* (EIPs) [29] will enable enterprises to cut these costs: EIPs will indeed provide individual enterprises with the computing and networking infrastructure needed to host their ENs in the edge tier on a pay-per-use basis, so that these enterprises do not need to purchase, deploy, and manage their own edge infrastructures. Moreover, by multiplexing the same physical infrastructure among multiple tenants, each EIPs can maximize the utilization of its resources, thus amortizing capital and operational costs and making profit.

An EIP runs the applications to process IoT data on their infrastructure by encapsulating each one of them into a set of *Virtual Machines* (VMs) that are hosted in its ENs. The owner of an application and the EIP that runs it are bound by a contractual agreement defining the amount of money that the owner of the application will correspond to the EIP for running its application and also the monetary penalty that the EIP will pay to the owner of the application when the agreed QoS is not met.

To improve its *net profit* (i.e., the difference between its revenues and costs), an EIP can thus reduce its costs by either switching off (part of) its infrastructure to save energy or by allocating as much

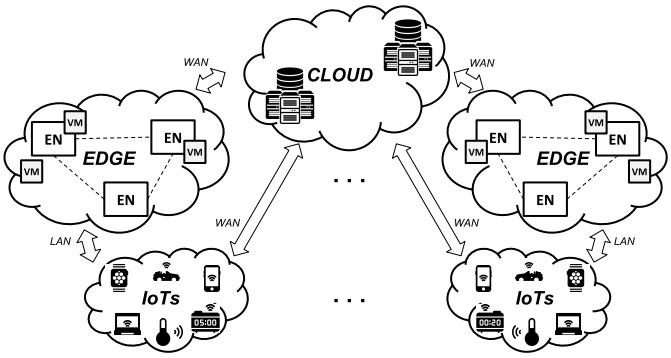


**Figure 1: System architecture.**

resources as required by the hosted application to avoid paying QoS penalties [16, 17]. To do so, an EIP can rely on *server consolidation* [28] to allocate several VMs on each EN, in the attempt to use as little ENs as possible. The maximization of the *consolidation level* that is achieved by allocating on each EN as many VMs as possible has thus become one of the major goals of an EIP, that however is a challenge to achieve because it requires to decide the best way to allocate the required VMs onto the EIP infrastructure in face of time-varying workload.

In this paper, we propose the *Online Profit Maximization* (OPM) algorithm, an approximation algorithm that aims at increasing the profit of an EIP without a priori knowledge. We assess the performance of OPM by simulating its behavior for a variety of realistic scenarios, in which data are generated by a population of moving users, and by comparing the results it yields against those attained by an *oracle* (i.e., an unrealistic algorithm able to always make optimal decisions) and by a state-of-the-art alternative. Our results indicate that OPM is able to achieve results that are always within 1% of the optimal ones, and that always outperforms the alternative solution.

The rest of the paper is organized as follows. Section 2 presents details of the system model, and Section 3 discusses the problem tackled in this paper. In Section 4, we present the algorithm we propose to tackle this problem, and in Section 5, we evaluate its performance via simulation. Finally, we end the paper with related works (in Section 6) and possible future extensions (in Section 7).

## 2 SYSTEM MODEL

We consider a system, whose architecture is outlined in Figure 1, where a population of geo-distributed IoT devices (either stationary or mobile) generates very large amounts of data that require real-time or near-real-time processing to deliver various types of data-analytics services. To accommodate these stringent requirements, a set of resource-rich ENs are deployed, either as cloudlets [27] or as micro data centers [2], in the edge tier (i.e., in the proximity of these data sources) to run a set of virtualized applications to process the data generated by the IoT devices.

Each EN $i \in \mathcal{E}$ is characterized by its CPU capacity $C_i$ which is measured by means of a suitable benchmark (e.g., [25]), and by its power consumption $w_i(u)$, which is computed as in [26]:

$$w_i(u) = W_i^{\min} + u \cdot \left(W_i^{\max} - W_i^{\min}\right) \tag{1}$$

where $u \in [0, 1]$ is the CPU utilization of the EN, and $W_i^{\min}$ and $W_i^{\max}$ denote its power consumption (in Watts) when its CPU is in the idle state and when it is fully utilized, respectively. We assume that ENs are provided by suitable dynamic resource allocation mechanisms (e.g., [4, 6–8]) enabling them to partition their physical capacity across the VMs they run.

Each application $j \in \mathcal{A}$ is characterized by its QoS target, which is quantified by the maximum value $Q_j$ that the mean request processing time $D_j$ can take (i.e., $D_j \leq Q_j$). To cover all the various geographic areas of its interest, application $j$ is deployed with one or more instances in the various areas, whereby instances located in a given area process all the data generated for $j$ by the devices located in that area. This model is representative of real-world IoT applications, arising in a variety of domains (e.g. health-care, smart cities, and agriculture monitoring [14]), where sensors generate data (the *IoT* clouds in Figure 1) that is stored and possibly processed at the edge of the network (the *EDGE* clouds in Figure 1).

The instances of any application are encapsulated in a set of identical VMs, each one running a single instance and hosted by an EN. Each VM of an application $j$ is cloned from a common *VM template* $k$ which is characterized by the request processing rate $\mu_{j,k}$ that, without loss of generality, it is assumed to be determined only by the amount of physical CPU capacity allocated to that VM, [1] and that this amount is the same for all its instances, and remains constant for all their lifetimes. For the same application $j$, we consider different classes $\mathcal{K}$ of VM templates, each one with different physical resource requirements, that an EIP can choose to achieve the maximum consolidation level on its ENs.

To ensure that all the clones of VM template $k$ of application $j$ exhibit the same value of $\mu_{j,k}$, we assume that each one of them receives, on the EN $i$ on which it runs, a suitable amount of CPU capacity $U_{i,k}$ computed as in [5]: $U_{i,k} = U_{x,j} \cdot C_x / C_i$, where $C_i$ and $C_x$ denote the physical CPU capacity of EN $i$ and of the reference EN $x$ used for profiling VM template $k$, respectively.

To achieve the QoS target of application $j$ in a given area, it is needed to suitably choose the number $N_{j,k}$ of class $k$ VMs allocated on ENs located in that area so as to ensure that $D_i \leq Q_i$, which however depends on the time-varying load intensity $\lambda_j(t)$ that application $j$ faces in that area. Likewise existing works (e.g., [9, 10], we assume that an EIP divides the time axis in equally spaced time intervals and that for each interval $\tau$ the EIP is able to estimate the load intensity $\lambda_j(\tau)$ for time interval $\tau$ at the end of the preceding time interval $(\tau - 1)$.

The values of $\lambda_j(\tau)$ are fed as input into an M/M/c-FCFS queueing model [12], with $c = N_{j,k}(\tau)$, representing the set of identical class $k$ VMs of $j$ allocated in a given area to process a stream of incoming requests which is fairly distributed among them. The solution of this model yields the minimum number $N_{j,k}(\tau)$ of class $k$ VMs in time interval $\tau$ that satisfies $D_j \leq Q_j$ as follows (for readability purposes, we drop the dependence on $\tau$):

$$D_j = \frac{G}{\mu_{j,k} N_{j,k} - \lambda_j} + \frac{1}{\mu_{j,k}}, \tag{2}$$

$$N_{j,k} \geq \frac{G}{Q_j - (1/\mu_{j,k})} + \frac{\lambda_j}{\mu_{j,k}}. \tag{3}$$

[1] The extension to multiple types of physical resources (e.g., RAM and storage) is straightforward (e.g., see [18]).

where $\rho = \frac{\lambda_j}{N_{j,k}\mu_{j,k}}$ and $G = [1+(1-\rho)(\frac{N_{j,k}!}{(\rho N_{j,k})^{N_{j,k}}}) \sum_{n=0}^{N_{j,k}-1} \frac{(N_{j,k}\rho)^n}{n!}]^{-1}$.

The owner of an application $j$ and the EIP that runs it are bound by a contractual agreement stating that the owner of $j$ will correspond to the EIP a certain amount of money per each unit of time, computed according to an agreed-upon *revenue rate* $R_j$. In addition, the EIP will pay the owner of $j$ an amount of money per each unit of time during which the QoS target of $j$ is not met, computed according to the agreed-upon *penalty rate* $L_j$.

## 3 PROBLEM STATEMENT

An EIP aims at increasing its net profit as much as possible, given the request for the allocation of all the applications it runs. We assume that in each area the EIP has enough resource capacity to meet the QoS requirements of all applications, so it never needs to allocate VMs on resources belonging to other areas. The overall net profit earned by EIP is therefore simply the sum of the net profits it earns in each single area.

The net profit rate $P$ (i.e., the profit an EIP makes per unit of time) in a given area is computed as the following difference (for readability purposes, we drop the dependency on time interval $\tau$):

$$P = \sum_{\substack{j\in\mathcal{A},\\ k\in\mathcal{K}}} R_j n_{j,k} - \Bigg(\sum_{i\in\mathcal{E}}\Big(x_i w_i(u_i)E + \sum_{j\in\mathcal{A}} \max\{0,\, y_{i,j,k} - a(i,j,k)\}M_{j,k} + \sum_{i\in\mathcal{E}}\Big[x_i\big(1-o(i)\big)\Big]A_i + \sum_{i\in\mathcal{E}}\Big[\big(1-x_i\big)o(i)\Big]S_i\Big) + \sum_{j\in\mathcal{A}} \mathbf{1}_{[0,N_{j,k})}(n_{j,k})L_j\Bigg) \tag{4}$$

where $x_i$ is 1 if EN $i$ is powered on and 0 otherwise, $y_{i,j,k}$ is the number of class $k$ VMs that are actually allocated for application $j$ on EN $i$, $n_{j,k} \leq N_{j,k}$ is defined as $n_{j,k} = \sum_{i\in\mathcal{E}} y_{i,j,k}$, $E$ is the electricity price (per unit of time), $u_i$ is the overall physical capacity of EN $i$ allocated to the VMs it hosts, $a(i,j,k)$ (where $a : \mathcal{E}\times\mathcal{A}\times\mathcal{K}\rightarrow\mathbb{N}$) is the number of class $k$ VM for $j$ that are already allocated on EN $i$, $M_{j,k}$ is the cost for cloning or reallocating a class $k$ VM of $j$, $o(i)$ (where $o : \mathcal{E}\rightarrow\{0,1\}$) is 1 if EN $i$ is already powered on and 0 otherwise, $A_i$ and $S_i$ are the costs for powering on and off EN $i$, respectively, and $\mathbf{1}_\Omega(x)$ is the *indicator function* which has value 1 if $x\in\Omega$ and 0 otherwise.

Eq.(4) has the following meaning: the first term of the difference is the sum of the revenue rates $R_j$ that EIP charges (per unit of time) to each application $j$ for hosting $n_{j,k}$ of its class $k$ VMs; while the second term of the difference represents the costs that EIP incurs (per unit of time) to run the above VMs. This cost, in turn, is given by the sum of five costs, namely (a) the energy cost rates resulting from the execution of overall CPU capacity allocated to all the VMs it hosts (see Eq. 1), (b) the cost rates for cloning or migrating its hosted VMs, (c) the energy cost rates for turning on the ENs where to allocate these VMs, (d) the energy cost rates for turning off unused ENs, and (e) the possible monetary penalty rates $L_j$ that EIP incurs when the QoS of some application $j$ is not met (i.e., when $n_{j,k} < N_{j,k}$).

Maximizing this profit rate is a challenging task which involves solving an optimization problem to find those values of $x_i$ and $y_{i,j,k}$ that maximizes $P$ for all time intervals $\tau$, and that takes into account the time-varying workload, the electricity price and the application penalties. Intuitively, for each time interval, when the number of VMs to allocate on a EN $i$ is so small that the resulting net profit is negative, EIP must decide whether it is more profitable to not allocate any VM on EN $i$ (thus opting to pay the monetary penalties for violating the QoS of the related applications), or it is instead better to allocate the VMs anyways to avoid paying high application penalties. Also, when the number of VMs is so large that it needs more than one EN to allocate them, EIP must decide whether it is more profitable to allocate all of them, or it is instead better to allocate only the ones that leads to a positive profit (thus paying the monetary penalties for those applications whose QoS is not met).

If all the system parameters were available in advance, Eq. (4) could be maximized by solving an *offline Mixed Integer Linear Program* (MILP). However, the time-varying nature of the workload makes this assumption unrealistic for many real-world scenarios.

To cope with this uncertainty, in this paper, we propose OPM, an *online optimization algorithm* that, for each time interval $\tau$, finds the values of $x_i$ and $y_{i,j,k}$ that maximize Eq. (4) in the interval $\tau$, without a priori knowledge and by only assuming that the EIP can estimate the $\lambda_j(\tau)$ at the end of the preceding interval $(\tau-1)$.

## 4 THE OPM ALGORITHM

In this section, we present the OPM algorithm we devised to maximize the net profit rate of an EIP. We assume that the time axis is divided in equally spaced time intervals and that, at the end of each interval $(\tau-1)$, the EIP invokes the OPM algorithm to find the VM allocation that maximizes the profit in the next interval $\tau$.

**Algorithm 1** The OPM algorithm (run for every time interval $\tau$).

1: **procedure** OPM
2: For each $j\in\mathcal{A}$, estimate $\lambda_j(\tau)$.
3: For each $j\in\mathcal{A}, i\in\mathcal{K}$, find $N_{j,k}$ by means of Eq. (3).
4: Find $\pi^* = \langle\{x_i^*|\forall i\in\mathcal{E}\}, \{y_{i,j,k}^*|\forall i\in\mathcal{E}, j\in\mathcal{A}, k\in\mathcal{K}\}\rangle$ by solving the MILP of Figure 2.
5: Apply $\pi^*$ to the system.
6: **end procedure**

More specifically, as reported in Algorithm 1, at the end of time interval $(\tau-1)$, EIP invokes OPM to estimate the maximal workload intensity $\lambda_j(\tau)$ for each application $j$ it hosts (line 2). Subsequently, with the just estimated $\lambda_j(\tau)$ and by means of Eq. (3), EIP finds the minimum number of VMs required to meet the QoS of $j$ (line 3). Then, EIP solves the optimization problem of Figure 2 to find the values of $x_i$ and $y_{i,j,k}$ decision variables that maximize its net profit as defined by Eq. (4) (line 4). Finally, EIP applies the solution of the MILP of Figure 2 (line 5).

Figure 2 shows the MILP used to maximize the profit in time interval $\tau$, where we use the same notation defined in Section 3, and, to ease readability, we drop from the model the dependence from $\tau$.

In the optimization model, $x_i$ are binary decision variables, $y_{i,j,k}$ and $n_{j,k}$ are non-negative integer decision variables, $u_j$ are non-negative real decision variables, and, finally, $s_j$ are binary decision variables indicating whether some VMs have been allocated for application $j$ ($s_j = 1$) or not ($s_j = 0$).

The objective function Eq. (5a) of the optimization model represents the overall net profit rate earned by EIP, which is defined as

$$\max \sum_{j\in\mathcal{A}}\sum_{k\in\mathcal{K}} R_j n_{j,k} - \Bigg[\sum_{i\in\mathcal{E}}\Big(x_i W_i^{\min} + (W_i^{\max} - W_i^{\min})u_i\Big)E + \sum_{i\in\mathcal{E}}\sum_{j\in\mathcal{A}}\sum_{k\in\mathcal{K}} \max\Big\{0,\, y_{i,j,k} - a(i,j,k)\Big\}M_{j,k} + \sum_{i\in\mathcal{E}}\Big[x_i\big(1-o(i)\big)\Big]A_i + \sum_{i\in\mathcal{E}}\Big[\big(1-x_i\big)o(i)\Big]S_i + \sum_{j\in\mathcal{A}}\Bigg[(\forall k\in\mathcal{K}: N_{j,k}>0)\wedge\Big(\big(s_j=0\big) \vee \Big(\exists k\in\mathcal{K}:\big(n_{j,k}>0\big)\wedge\big(n_{j,k}<N_{j,k}\big)\Big)\Big)\Bigg]P_j\Bigg] \tag{5a}$$

s.t.

$$n_{j,k} = \sum_{i\in\mathcal{E}} y_{i,j,k}, \quad \forall j\in\mathcal{A}, k\in\mathcal{K}, \tag{5b}$$

$$u_i = \sum_{j\in\mathcal{A}}\sum_{k\in\mathcal{K}} y_{i,j,k}B_{i,k}, \quad \forall i\in\mathcal{E}, \tag{5c}$$

$$s_j = \Big(\exists i\in\mathcal{E}, k\in\mathcal{K}: y_{i,j,k}>0\Big), \quad \forall j\in\mathcal{A}, \tag{5d}$$

$$\nexists k'\in\mathcal{K}, k'\neq k: \Big(n_{j,k}>0\Big)\wedge\Big(n_{j,k'}>0\Big), \ \forall j\in\mathcal{A}, k\in\mathcal{K}, \tag{5e}$$

$$u_i \le x_i, \quad \forall i\in\mathcal{E}, \tag{5f}$$

$$n_{j,k} \le N_{j,k}, \quad \forall j\in\mathcal{A}, k\in\mathcal{K}, \tag{5g}$$

$$x_i\in\{0,1\}, \quad \forall i\in\mathcal{E}, \tag{5h}$$

$$y_{i,j,k}\in\mathbb{N}, \quad \forall i\in\mathcal{E}, j\in\mathcal{A}, k\in\mathcal{K}, \tag{5i}$$

$$n_{j,k}\in\mathbb{N}, \quad \forall j\in\mathcal{A}, k\in\mathcal{K}, \tag{5j}$$

$$s_j\in\{0,1\}, \quad \forall i\in\mathcal{E}, \tag{5k}$$

$$u_i\in\mathbb{R}^*, \quad \forall i\in\mathcal{E}. \tag{5l}$$

**Figure 2: The profit maximization model.**

the difference between the revenues obtained by the allocation of VMs, and the costs due both to the electricity power absorbed by the powered-on ENs and to QoS violations (if any), as in Eq. (4).

The maximization of this objective function is bound to the following constraints: Eq. (5b) defines, for each application $j$, the value of $n_{j,k}$ as the sum of the number of class $k$ VMs allocated on the ENs; Eq. (5c) defines, for each EN $i$, the value of $u_i$ as the sum of the CPU capacity requirements $B_{i,k}$ of the VMs allocated on $i$, over all VM classes $k\in\mathcal{K}$; Eq. (5d) defines, for each application $j$, the value of $s_j$ as a boolean value indicating whether some VMs have been allocated to $j$ ($s_j=1$) or not ($s_j=0$); Eq. (5e) ensures that all VMs allocated for a given application belong to the same category; Eq. (5f) ensures that the allocated CPU capacity of a powered-on EN is not exceeded; Eq. (5g) ensures that for each application no more VMs are allocated than needed; Eq. (5h)–Eq. (5l) define the domain of decision variables $x_i, y_{i,j,k}, n_{j,k}, s_j$, and $u_i$, respectively.

In general, the profit resulting after iteratively applying OPM for each considered time interval is suboptimal because OPM has a limited view of the future. In Section 5, we experimentally evaluate this sub-optimality by comparing the results achieved by OPM with the ones computed by an *oracle* (optimal) algorithm which solves the offline MILP obtained by extending the model of Figure 2 to cover all time intervals.

# 5 EXPERIMENTAL EVALUATION

To assess the efficacy of the proposed approach in increasing the net profits for an EIP, we perform an experimental evaluation via simulation in which we run our algorithm for different scenarios. In these scenarios, we vary the workload of each application as well as other system parameters, and we assess the impact of them on the performance of the proposed algorithm. Also, to evaluate the effectiveness of the proposed approach with respect to the state-of-the-art, we compare the performance of our algorithm with another alternative approach.

The results we obtain from these experiments show the ability of our algorithm to increase the profit of EIP so as it is never less than 99% of the optimum (as computed by the oracle algorithm), and that it always outperforms the alternative solution.

In the rest of this section, we first provide the settings we use in our experimental scenarios (Section 5.1), and then we show the results we obtain by applying our proposed algorithm to these scenarios (Section 5.2).

## 5.1 Setup

Given a rectangular geographic area of size $100\times100$, we consider 2 applications, whereby $Q_1 = 0.5$ sec and $Q_2 = 1.0$ sec. For all applications, we consider 3 classes of VM such that $\mu_{1,k} = \mu_{2,k} = [\,2\ 4\ 6\,]$, $M_{1,k} = M_{2,k} = [\,0.001\ 0.002\ 0.003\,]$ req/sec, and $B_{i,k} = \begin{bmatrix} 0.05 & 0.10 & 0.20 \\ 0.085 & 0.17 & 0.34 \\ 0.1445 & 0.289 & 0.578 \end{bmatrix}$. Each application serves a population of 1000 users that moves inside the considered area according to the *random waypoint* model [21], with a minimum and maximum speed of 1 m/sec and 2 m/sec, respectively, and with a pause time of 1 sec. Each user issues requests to the associated application at a rate of 0.5 req/sec for the first application and of 1 req/sec for the second application.

We assume that the electricity price $E$ is charged hourly to EIP and we set it to 0.0001 \$/Wh. The edge infrastructure of EIP consists of 9 ENs (initially, all powered off) whereby $W_i^{\max} = 200$, for $i = 1,\ldots,9$, and: (i) $W_i^{\min} = 100$ W, $A_i = 0.002$ \$ and $S_i = 0.001$ \$, for $i = 1,\ldots,3$; (ii) $W_i^{\min} = 50$ W, $A_i = 0.001$ \$ and $S_i = 0.0005$ \$, for $i = 4,\ldots,6$; (iii) $W_i^{\min} = 25$ W, $A_i = 0.0005$ \$ and $S_i = 0.00025$ \$, for $i = 7,\ldots,9$. The considered infrastructure is powerful enough to meet the QoS of all applications at their maximum load intensity.

In our experiments we set the same revenue rate $R_j$ and QoS penalty rate $L_j$ for all applications $j$, and we vary them as a function of $E$ as follows: (i) $R_j = 50E$, $R_j = 100E$, and $R_j = 500E$, (ii) $P_j = 0$, $P_j = R_j$, and $P_j = 2R_j$.

Finally, we set the discretization time interval to 1 hour, so as the EIP aims at increasing its hourly profit.

To run our experiments, we develop an ad hoc discrete-event systems simulator in C++ where we use the *independent replications* as output analysis techniques [11] (where each independent replica is 24 hours long and where the whole simulation stops when the relative precision of the 95% confidence interval is $\le 4\%$) and we use the IBM ILOG CPLEX solver 12.8 [19] for solving the optimization problem discussed in Section 4.

## 5.2 Results

The results attained by our proposed solution are compared with *MCAPP-IM*, an alternative state-of-the-art approach proposed in [10]. The authors of [10] extend the classical Hungarian algorithm for solving the assignment problem and propose an approximation algorithm for the efficient placement of multi-component applications in edge computing systems. Since the authors of [10] do not specify if the servers where they place the application components are physical servers (i.e., ENs) or virtual servers (i.e., VMs), we implemented two variants of MCAPP-IM, one (called *MCAPP-IM (alt#1)*) that assigns one application component to a different EN, and another one (named *MCAPP-IM (alt#2)*), that assigns one application component to a different VM, and for both variants we select as class $k$ VMs the ones that minimize the total EN resource demands. Unlike MCAPP-IM (alt#1), MCAPP-IM (alt#2) can exploit server consolidation to allocate on the same ENs two or more VMs running different application components. Thus, we expect that the MCAPP-IM (alt#2) performs better than MCAPP-IM (alt#1) because it is able to achieve a greater consolidation level.

To compare the performance of the various algorithms, we use the *competitive ratio* $r$ defined as the ratio between the profit $P$ obtained by a specific (approximation) algorithm and the profit $P^*$ as computed by the oracle algorithm (i.e., by solving the offline profit maximization problem over all considered time interval), that is $r = P/P^*$. The nearer to 1 is $r$, the closer is the solution of the approximation algorithm to the optimal one. Thus, given two alternative approximation algorithms, the best one is the one with the greater value of $r$.

In Table 1, we report the results of the experimental simulations grouped by the different combinations of $\langle R_j, P_j \rangle$ considered in our evaluation and described in Section 5.1. In this table, the rows correspond to the results obtained by the various algorithms, and the columns report the name of the algorithm (column "Algorithm"), the profit obtained at the end of the simulation averaged over all the independent replicas along with its standard deviation (column "Mean Profit (s.d.)"), and the competitive ratio (column "$r$").

As shown in Table 1, OPM is always able to achieve a competitive ratio $r \geq 0.99$, meaning that, for the evaluated scenarios, the EIP obtains a profit which is at least 99% of the optimal one. Instead, MCAPP-IM is not able to do so, for both of its variants. Specifically, since MCAPP-IM (alt#1) cannot exploit server consolidation, the resulting VM allocation leads to higher energy consumption costs because it requires more powered on ENs than the other approaches. Also, MCAPP-IM (alt#1) is often unable to allocate all the VMs required to meet application QoS in face of the current workload, thus resulting in a lower revenue and in the payment of the monetary penalties for QoS violation. The performance attained by MCAPP-IM (alt#2) is better than the one achieved by MCAPP-IM (alt#1) because it can exploit server consolidation and thus can consume less energy and reduce QoS violations, but it is worse than the one of OPM because, in general, the resulting VM allocation in each time interval is not optimal for that interval as instead it is for the one computed by OPM (e.g., when deciding where to allocate the various VMs, MCAPP-IM (alt#2) does not take into account the possible monetary penalties that EIP will incur for not allocating all the required VMs for a given application).

Therefore, OPM always outperforms both algorithms because, for any time interval, it is able to find the VM allocation that maximizes the profit for that interval, by taking into account several factors like the current workload conditions, the current VM allocation, the current power status of ENs, and the revenues and the costs resulting from the VM allocation.

For the way the values of $R_j$ and $P_j$ affect the quality of the solution achieved by OPM, we note from Table 1 that an increment of $R_j$ leads to a nearly large increment in the resulting profit (e.g., compare "Case $\langle R_j = 50E, P_j = 0 \rangle$" with "Case $\langle R_j = 500E, P_j = 0 \rangle$", where the profit obtained in the former case is nearly half the one achieved in the latter case). Instead, the effects of a change in $P_j$ have a little impact on the resulting profit (e.g., compare "Case $\langle R_j = 500E, P_j = 0 \rangle$" with "Case $\langle R_j = 500E, P_j = 2R_j \rangle$", where the profits achieved in both cases are nearly equal). This is due to the way experimental scenarios have been set, whereby it is always possible to find an optimal VM allocation that it is able to accommodate the resource demands of all the VMs required to meet applications QoS, and to the ability of OPM to find this optimal VM allocation, thus enabling OPM to never violate any QoS.

Finally, we note that the execution time of OPM (on a server equipped with 2 Intel Xeon E5-2665 CPUs and 96 GiB RAM, and running Linux 4.15.14) is always $< 17$ seconds.

## 6 RELATED WORKS

In the last years, edge computing has attracted lot of interest in the scientific community.

In [3, 24], authors propose algorithms to share compute resources among ENs in order to satisfy users' compute demands. They define a utility metric that accounts for communication costs and resource sharing benefits, and design an algorithm to make resource sharing decisions according to this metric. In [1, 20], authors propose task scheduling algorithms for cellular networks, where ENs are located in the network cells to provide computing capabilities. Unfortunately, the above works do not scale properly considering that edge systems are expected to cover millions of IoT devices [22]. Conversely, OPM is able to manage a large number of components by dynamically provisioning the right number of VMs to meet QoS targets.

In [23, 30], authors propose a strategy to offload homogeneous devices at low transmission and energy costs, while ensuring a high utility and meeting the applications' deadline. Unlike these works, OPM is able to cope with heterogeneous devices.

Finally, in [10], the authors model the problem of efficiently placing multi-component applications in edge computing systems as an assignment problem and propose an extension of the Hungarian algorithm to reduce EIP costs by taking into account communication costs between application components. In contrast, as already discussed in Section 5, OPM is able to exploit server consolidation and resource heterogeneity to increase the EIP profit and by taking into account QoS targets.

## 7 CONCLUSIONS

In this paper, we propose OPM, an approximation algorithm for profit maximization that aims at increasing the profit of EIPs in face of time-varying workload, by taking into account energy costs

**Table 1: Experimental results.**

| Algorithm | Mean Profit (s.d.) | $r$ |
|---|---|---|
| *Case* $\langle R_j = 50E,\ P_J = 0 \rangle$ | | |
| **OPM** | **2.119 (0.113)** | **0.993** |
| MCAPP-IM (alt#1) | -0.903 (0.023) | -0.421 |
| MCAPP-IM (alt#2) | 1.159 (0.058) | 0.545 |
| *Case* $\langle R_j = 50E,\ P_j = R_j \rangle$ | | |
| **OPM** | **2.111 (0.110)** | **0.993** |
| MCAPP-IM (alt#1) | -1.000 (0.040) | -0.471 |
| MCAPP-IM (alt#2) | 1.162 (0.057) | 0.547 |
| *Case* $\langle R_j = 50E,\ P_j = 2R_j \rangle$ | | |
| **OPM** | **2.114 (0.108)** | **0.993** |
| MCAPP-IM (alt#1) | -1.107 (0.069) | -0.519 |
| MCAPP-IM (alt#2) | 1.160 (0.056) | 0.546 |
| *Case* $\langle R_j = 100E,\ P_J = 0 \rangle$ | | |
| **OPM** | **5.039 (0.264)** | **0.997** |
| MCAPP-IM (alt#1) | 0.158 (0.022) | 0.031 |
| MCAPP-IM (alt#2) | 2.359 (0.127) | 0.470 |
| *Case* $\langle R_j = 100E,\ P_j = R_j \rangle$ | | |
| **OPM** | **5.033 (0.254)** | **0.997** |
| MCAPP-IM (alt#1) | -0.075 (0.055) | -0.015 |
| MCAPP-IM (alt#2) | 2.364 (0.124) | 0.469 |
| *Case* $\langle R_j = 100E,\ P_j = 2R_j \rangle$ | | |
| **OPM** | **5.013 (0.260)** | **0.997** |
| MCAPP-IM (alt#1) | -0.310 (0.109) | -0.061 |
| MCAPP-IM (alt#2) | 2.364 (0.119) | 0.469 |
| *Case* $\langle R_j = 500E,\ P_J = 0 \rangle$ | | |
| **OPM** | **28.409 (1.513)** | **0.999** |
| MCAPP-IM (alt#1) | 8.516 (0.154) | 0.300 |
| MCAPP-IM (alt#2) | 11.980 (0.665) | 0.423 |
| *Case* $\langle R_j = 500E,\ P_j = R_j \rangle$ | | |
| **OPM** | **28.395 (1.453)** | **0.999** |
| MCAPP-IM (alt#1) | 7.472 (0.277) | 0.263 |
| MCAPP-IM (alt#2) | 12.012 (0.659) | 0.423 |
| *Case* $\langle R_j = 500E,\ P_j = 2R_j \rangle$ | | |
| **OPM** | **28.335 (1.414)** | **0.998** |
| MCAPP-IM (alt#1) | 6.292 (0.626) | 0.221 |
| MCAPP-IM (alt#2) | 11.977 (0.636) | 0.423 |

and applications' QoS, and by exploiting server consolidation. Experimental results show that OPM is able to achieve a profit that is very close to the optimum that can be earned, and it also outperforms alternative state-of-the-art solutions.

We plan to extend this works in several ways. First, we want to assess the impact of the lookahead on the performance of OPM, by enabling it to make workload predictions over multiple time intervals. Also, we want to evaluate the way the prediction error affects the achieved profit. Moreover, we intend to consider scenarios with limited resource capacity in each area that will require cross-area allocation of VMs. Finally, we consider to implement and validate OPM on a real testbed.

## ACKNOWLEDGMENTS

This research is original and has a financial support of the Università del Piemonte Orientale.

## REFERENCES

[1] M. Aazam et al. 2015. Dynamic resource provisioning through Fog micro datacenter. In *IEEE Int. Conf. on PerCom Workshops*. 105–110.
[2] M. Aazam et al. 2015. Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT. In 29$^{th}$ *Conf. on AINA*. 687–694.
[3] S.F. Abedin et al. 2015. A Fog based system model for cooperative IoT node pairing using matching theory. In 17$^{th}$ *Asia-Pacific APNOMS*. 309–314.
[4] L. Albano, C. Anglano, M. Canonico, and M. Guazzone. 2013. Fuzzy-Q&E: Achieving QoS Guarantees and Energy Savings for Cloud Applications with Fuzzy Control. In 3$^{rd}$ *Int. Cloud and Green Computing Conference (CGC'13)*. 159–166.
[5] C. Anglano, M. Canonico, P. Castagno, M. Guazzone, and M. Sereno. 2018. A Game-Theoretic Approach to Coalition Formation in Fog Provider Federations. In 3$^{rd}$ *IEEE Int. Conf. on Fog and Mobile Edge Computing (FMEC'18)*.
[6] C. Anglano, M. Canonico, and M. Guazzone. 2015. FC2Q: Exploiting fuzzy control in server consolidation for cloud applications with SLA constraints. *Concurrency and Computation: Practice and Experience* 27, 17 (2015), 4491–4514.
[7] C. Anglano, M. Canonico, and M. Guazzone. 2017. FCMS: A fuzzy controller for CPU and memory consolidation under SLA constraints. *Concurrency and Computation: Practice and Experience* 29, 5 (2017).
[8] C. Anglano, M. Canonico, and M. Guazzone. 2018. Prometheus: A flexible toolkit for the experimentation with virtualized infrastructures. *Concurrency and Computation: Practice and Experience* 30, 11 (2018), e4400.
[9] C. Anglano, M. Guazzone, and M. Sereno. 2014. Maximizing profit in green cellular networks through collaborative games. *Computer Networks* 75, Part A (2014), 260–275.
[10] T. Bahreini et al. 2017. Efficient Placement of Multi-component Applications in Edge Computing Systems. In 2$^{nd}$ *ACM/IEEE SEC*. ACM, Article 5, 11 pages.
[11] J. Banks et al. 2010. *Discrete-Event System Simulation* (5th ed.). Prentice Hall.
[12] G. Bolch et al. 2006. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications* (2nd ed.). Wiley.
[13] F. Bonomi et al. 2014. *Fog Computing: A Platform for Internet of Things and Analytics*. Springer, Cham, 169–186.
[14] Charles C Byers. 2017. Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks. *IEEE Commun. Mag.* 55, 8 (2017), 14–20.
[15] P. Garcia Lopez et al. 2015. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.* 45, 5 (2015), 37–42.
[16] M. Guazzone, C. Anglano, and M. Canonico. 2011. Energy-Efficient Resource Management for Cloud Computing Infrastructures. In 3$^{rd}$ *IEEE Int. Conf. on Cloud Computing Technology and Science (CloudCom'11)*. 424–431.
[17] M. Guazzone, C. Anglano, and M. Canonico. 2012. Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems. In 1$^{st}$ *Int. Workshop on Energy-Efficient Data Centres (E2DC)*. 81–92.
[18] M. Guazzone, C. Anglano, and M. Sereno. 2014. A Game-Theoretic Approach to Coalition Formation in Green Cloud Federations. In 14$^{th}$ *IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGRID'14)*. 618–625.
[19] IBM. 2018. ILOG CPLEX Optimization Studio. (2018).
[20] K. Intharawijitr et al. 2016. Analysis of fog model considering computing and communication latency in 5G cellular networks. In *IEEE Conf. on PerCom*. 1–4.
[21] D.B. Johnson et al. 1996. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Springer US, 153–181.
[22] C. Mouradian et al. 2017. A Comprehensive Survey on Fog Computing: State-of-the-art and Research Challenges. *IEEE Commun. Surveys Tuts.* (2017).
[23] S. Ningning et al. 2016. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Commun.* 13, 3 (2016), 156–164.
[24] J. Oueis et al. 2015. Small cell clustering for efficient distributed fog computing: A multi-user case. In *2015 IEEE* 82$^{nd}$ *VTC Fall*. 1–5.
[25] Primate Labs, Inc. 2018. GeekBench. (2018).
[26] S. Rivoire et al. 2008. A Comparison of High-level Full-system Power Models. In *2008 Conf. on HotPower*. USENIX, 3–3.
[27] M. Satyanarayanan et al. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Comput.* 8, 4 (2009), 14–23.
[28] W. Vogels. 2008. Beyond Server Consolidation. *Queue* 6, 1 (2008), 20–26.
[29] J. Weinman. 2017. The Economics of the Hybrid Multicloud Fog. *IEEE Cloud Computing* 4, 1 (2017), 16–21.
[30] D. Ye et al. 2016. Scalable fog computing with service offloading in bus networks. In *IEEE* 3$^{rd}$ *Int. Conf. on CSCloud*. 247–251.