Energy Efficient Edge Computing Enabled by Satisfaction Games and Approximate Computing

Nafis Irtija, Iraklis Anagnostopoulos, *Member, IEEE*, Georgios Zervakis, Eirini Eleni Tsiropoulou, *Senior Member, IEEE*, Hussam Amrouch, *Member, IEEE*,, and Jörg Henkel, *Fellow, IEEE*

***Abstract*—In this paper, we introduce an energy efficient edge computing solution to collaboratively utilize Multi-access Edge Computing (MEC) and Fully Autonomous Aerial Systems (FAAS) to support the computing demands of the Internet of Things (IoT) nodes residing in Areas of Interest (AoIs) and executing machine learning tasks. The Satisfaction Games are adopted to determine whether the nodes' optimal partial task should be offloaded to the MEC server or to a hovering FAAS above the AoI. The decision is taken by considering IoT nodes' latency, energy consumption, and acceptable level of Deep Neural Network (DNN) inference accuracy drop constraints. We exploit the error resilience of DNNs and we enhance the FAAS with a heterogeneous approximate DNN accelerator that supports different computational precision and throughput, thus allowing to intelligently adapt to different computing demands. A reinforcement learning-based technique is introduced to enable the FAAS to autonomously optimize its trajectory, aiming at increasing the IoT nodes' satisfaction of their computing demands, while accounting for its flying and data processing energy cost. Our experimental results show the benefits of FAAS, MEC, and approximate computing in terms of increasing the number of satisfied users by 40% under a maximum accuracy drop of only 1%.**

***Index Terms*—Edge computing, energy efficiency, satisfaction games, reinforcement learning, deep neural networks accelerators, approximate computing.**

I. Introduction

With the emergence of diverse mobile services, such as safety protection and health monitoring, intelligent transportation, environmental monitoring, and the advent of Internet of Things (IoT), a variety of network services and applications has been introduced to support smart applications. Such modern applications are becoming more and more computationally demanding, requiring also low latency. However, IoT nodes are characterized by limited battery capacity and low computation capability. Thus, Multi-access Edge Computing (MEC) has been established as a promising solution to address these issues by bringing the computing functionalities closer to the IoT nodes [1]. Complimentary to the MEC technology, Unmanned Aerial Vehicles (UAVs) have been used to support both the increased communications and computing needs of IoT environments by acting as aerial mobile base stations and edge servers, respectively [2]. Also, the IoT nodes have different application-driven Quality of Service (QoS) requirements. Thus, satisfying the IoT nodes' minimum QoS requirements to efficiently support the corresponding IoT applications, without blindly maximizing their payoff, can conclude to major resources saving, both for the IoT system and the nodes.

Additionally, Deep Neural Networks (DNNs) have become the driving force for IoT nodes and particularly mobile devices. Many MEC services heavily rely on the utilization of DNNs, such as object detection, natural language processing, and virtual/augmented reality applications. Interestingly, previous research works [3]–[6] showed that DNNs feature increased error resilience making them an excellent candidate for approximation via computational precision reduction. Under the principle of approximate computing, DNNs can trade-off computation accuracy to further reduce IoT nodes' execution time and/or power consumption. To support such operations, heterogeneous DNN accelerators arise as the dominant computing architecture to balance the accuracy, throughput, and energy consumption. Such accelerators integrate different computing components tailored to the needs of the DNNs. We bridge these two concepts and we employ a heterogeneous approximate DNN accelerator to further boost the gains.

In this paper, we introduce an energy efficient edge computing solution enabled by the Satisfaction Games and Approximate Computing. Specifically, we utilize a Heterogeneous Approximate DNN accelerator (HADA), in order to support an IoT smart environment by exploiting the MEC technology and the Fully Autonomous Aerial Systems (FAAS).

A. Related Work & Background

Several recent research works exploit the joint combination of UAV and MEC technology to address the ground

Manuscript received May 21, 2021; revised September 27, 2021; accepted October 18, 2021. Date of publication October 26, 2021; date of current version February 16, 2022. This work was supported in part by the German Research Foundation (DFG) through the Project "ACCROSS: Approximate Computing Across the System Stack." The work of Nafis Irtija and Eirini Eleni Tsiropoulou was supported by NSF under Grant CRII-1849739. *(Corresponding author: Eirini Eleni Tsiropoulou.)*

Nafis Irtija and Eirini Eleni Tsiropoulou are with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131 USA (e-mail: nafis@unm.edu; eirini@unm.edu).

Iraklis Anagnostopoulos is with the School of Electrical, Computer and Biomedical Engineering, Southern Illinois University, Carbondale, IL 62901 USA (e-mail: iraklis.anagno@siu.edu).

Georgios Zervakis and Jörg Henkel are with the Chair for Embedded System, Department of Computer Science, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: georgios.zervakis@kit.edu; henkel@kit.edu).

Hussam Amrouch is with the Chair for Semiconductor Test and Reliability, Computer Science, Electrical Engineering Faculty, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: amrouch@iti.uni-stuttgart.de).

Digital Object Identifier 10.1109/TGCN.2021.3122911

users'/devices' computing demands. In [7], a third party agent is deployed, which learns the optimal task offloading decision of a user to a UAV and/or a ground MEC server to minimize the task execution delay and the users' energy consumption, based on a combination of machine learning and deep learning algorithms. The agent exchanges information with the users, the UAV, and the MEC server to learn the optimal strategy, which is finally communicated to the user. A multi-agent reinforcement learning approach is introduced in [8] to determine the users' optimal task offloading to a UAV cloudlet, while jointly considering the computing tasks' inter-dependencies, the communication conditions with the UAV cloudlet, and energy constraints of the latter one. The authors design an on-policy algorithm to perform the decision-making of the task offloading and an off-policy algorithm to reduce the cost.

The problem of optimal UAV's trajectory planning to serve the ground users, while considering several physical constraints, such as UAVs' energy consumption, users' mobility, and computing priority, has attracted significant interest by the recent literature. In [9], the trajectory learning and the users' association to UAVs is performed via maximizing the users' sum log-rate by mainly focusing on the communications performance improvement and considering static users. This framework has been extended for mobile users by adopting a stochastic gradient ascent algorithm to determine the UAV's trajectory. A novel approach of determining the UAV's optimal trajectory is proposed in [10] by minimizing the average age of information of the data collected by the users. The authors consider as constraint the time that is needed for the users' devices to harvest energy from the UAV. Also, the authors in [11] introduce a reinforcement leaning-based coalition formation mechanism, among the ground users, to determine the UAV's optimal trajectory by solving the optimization problem of maximizing the coalition heads' total energy availability.

The problem of energy efficient edge computing becomes even more challenging, while jointly considering the UAV and MEC technologies, and the UAV's trajectory planning. In [12], the authors consider mobile users and UAV-mounted MEC servers, where the latter ones execute a double deep Q-network algorithm to determine the UAV trajectory according to the users' location and QoS constraints. The optimal partial computing task offloading is studied in [13], where the users offload part of their computing tasks to the UAV and process the rest locally. To determine the users' optimal task offloading strategies, the authors solve a minimization problem of the users' total energy consumption, while jointly optimizing the data allocation, the resource partitioning, and the UAV trajectory. In [14], the authors determine the optimal offloading of computing tasks and the UAV's trajectory by minimizing the users' and UAV's average weighted energy consumption, while considering the system's physical constraints. The authors decompose the joint optimization problem into three subproblems (tasks offloading, resource allocation, and flying trajectory) and they adopt a Lyapunov-based approach to solve them. In [15], the authors formulate a minimization problem of the total users' energy consumption, under the constraints of users' QoS, latency, and UAV's energy availability. The successive convex approximation technique is adopted to solve the constrained optimization problem and determine the task offloading and the UAV's trajectory. In [16], reinforcement learning is used to optimize the users' achieved QoS, by offloading the tasks to the UAV, and the UAV's path planning. The main novelty of [16] is the adoption of a sigmoidal function to depict users' QoS demands and drive the reinforcement learning algorithm to maximize users' QoS.

Hardware accelerators, and particularly Neural Processing Units (NPUs), have been incorporated to support the execution of compute-intensive Deep Neural Networks (DNNs) at the edge. The heart of an edge-based NPU is an array of 8-bit of multiply-accumulate (MAC) units that profoundly accelerates the computations required by various phases throughout the execution of DNN inference (e.g., Google Edge TPU [17] and Samsung embedded-oriented NPU [18]). In [19], the authors use approximate computing and they propose an accuracy configurable approximate DNN accelerator that employs run-time reconfigurable MAC units. Based on the weights, they decide the approximation level that should be introduced in the performed multiplications to satisfy an accuracy threshold, while maximizing the energy gains. This time-consuming method does not scale and requires additional circuitry, inducing an area overhead. Similarly, the authors in [20] replace the accurate MAC units with more power-efficient approximate ones and control the introduced error with layer-based pattern matching. In [21], the authors also employ approximate multipliers, however the design is not reconfigurable, thus accurate operations are not supported and accuracy constraints might not always be satisfied. Another approach for HADAs and approximate computing, is to reduce the number of bits that NPUs operate on. In [22], [23], low precision for approximate DNNs are used, but such reconfigurable approaches do not scale with the number of MAC units.

Limited research work has been performed so far in the joint exploitation of UAVs, MEC and Approximate Computing technologies to provide energy efficient edge computing solutions, while accounting for the QoS requirements of the IoT nodes. Recently, the Fully Autonomous Aerial Systems (FAAS) have been introduced in the literature. The FAAS executes intelligent trajectory planning algorithms without exchanging any control information with ground controllers, as in the case of UAVs [24]. Additionally, it determines its optimal trajectory by usually running computationally-light reinforcement learning algorithms to support the data collection from the IoT nodes, their computing demand, charge them, and act as a relay [25]. Furthermore, the majority of the existing literature targets at the maximization of the IoT nodes' achieved QoS. However, several applications, e.g., augmented reality, autonomous driving, smart agriculture, have specific minimum QoS requirements, and the "blind" QoS maximization leads to energy inefficiencies, increased cost, and resource starvation of the system. Thus, the novel concept of Satisfaction Games has been introduced, where an equilibrium solution, i.e., Satisfaction Equilibrium (SE), is determined in an autonomous manner among the IoT nodes. At the SE, all the IoT nodes satisfy their minimum QoS constraints, and the system benefits in terms of saving resources to serve additional IoT nodes. Therefore, ultimately, the

system increases its scalability and robustness [26]. Limited research effort has been devoted so far in exploiting the benefits of Satisfaction Games in communications and computing environments, while emphasis has been placed on examining traditional power control problems at the access layer [27], [28].

B. Contributions and Outline

Despite the efforts made in the previous works to provide edge computing solutions to IoT nodes, the issue of jointly exploiting the benefits of FAAS, MEC, and Approximate Computing in a unified holistic framework still remains open. An even more challenging problem is to provide an energy efficient edge computing solution and intelligently use the system's resources, by incorporating the IoT nodes' specific minimum QoS requirements and satisfy them via an intelligent resource orchestration framework.

In this paper, we consider a smart IoT environment consisting of Areas of Interest (AoIs), where IoT devices reside, executing DNN-based applications. A macro base station (MBS) equipped with a MEC server resides in the smart IoT environment, and a FAAS hovers autonomously over the area to additionally serve the IoT nodes' computing demand. The FAAS is equipped with an HADA, providing different levels of DNN latency and approximation (in terms of computational precision), while the MEC server performs exact computing functionalities. The IoT nodes have personalized minimum QoS prerequisites, i.e., latency, energy consumption, and acceptable level of DNN inference accuracy drop. A Satisfaction Game is formulated to determine the IoT nodes' optimal partial task offloading to the MEC server or the FAAS, if the latter one hovers above the IoT nodes' AoI. The Satisfaction Equilibrium is determined to ensure the IoT nodes' satisfaction of their minimum QoS requirements, if this is feasible based on the overall system's available computing resources and nodes' error tolerance. Alternatively, the concept of Generalized Satisfaction Equilibrium is introduced, where some IoT nodes satisfy their minimum QoS prerequisites, while some other nodes fail to achieve their satisfaction, due to the limited computing resources. The latter outcome feeds a set of gradient ascent reinforcement learning algorithms that follow the theory of Learning Automata, to determine the optimal trajectory of the FAAs aiming at increasing the IoT nodes' satisfaction by providing supplementary computing functionalities and capacity.

The main contributions of our work are summarized below:

1) A smart IoT environment is introduced consisting of multiple Areas of Interest (AoIs), a MEC server residing at the MBS, and a FAAS, equipped with a HADA that flies among the AoIs to complement the system's computing capabilities. The HADA employs varying NPUs that feature different computational precision and throughput, allowing intelligent adaptation to the IoT nodes computing demands, under specific accuracy requirements.
2) An energy efficient edge computing solution is introduced by jointly examining (i) the MEC server's and the FAAS computing capabilities, (ii) the HADA's accuracy-throughput trade-off, and (iii) the nodes' minimum QoS prerequisites. Specifically, a Satisfaction Game is formulated among the nodes to determine their optimal partial computing task offloading decisions to the MEC server or to the FAAS, if the latter one hovers above their AoI. A distributed learning algorithm determines whether the Satisfaction Equilibrium and the Generalized Satisfaction Equilibrium of the nodes satisfy, partially or fully, their minimum latency and energy consumption constraints.
3) The FAAS's optimal trajectory is determined by a set of gradient ascent reinforcement learning algorithms. The proposed algorithms consider its energy consumption (both for navigating and processing the IoT nodes' computing tasks), as well as the satisfaction of the IoT nodes' computing demands.
4) A detailed set of numerical and comparative results is presented, highlighting the benefits of jointly exploiting the FAAS, the MEC and Approximate Computing technologies, along with considering the IoT nodes' minimum QoS prerequisites and error tolerance. The results reveal that an increase of 35% satisfied IoT nodes can be achieved by the FAAS computing contribution, with a negligible accuracy drop of only 0.5% at the IoT nodes. When the accepted accuracy drop increases to 1%, the number of satisfied IoT nodes also increases approximately to 40%.

The remainder of this research paper is organized as follows. In Section II, the overall system model and the concept of approximate computing via bit precision are introduced. In Section III-A, the IoT nodes' utility functions are defined capturing their minimum QoS requirements, and the problem of determining the IoT nodes' optimal partial computing tasks offloading is formulated in Section III-B. In Section III-C, the Satisfaction Equilibrium is determined via introducing a distributed learning Satisfaction Equilibrium algorithm. The FAAS optimal trajectory is determined in Section IV. Finally, a detailed experimental evaluation and comparative results are presented in Section V, while Section VI concludes the paper.

II. System Model & Approximate Computing

In this section, the system model adopted in this research work is presented by jointly exploiting the benefits of the FAAS, MEC, and Approximate Computing technologies. Particularly, Approximate Computing is supported by the utilization of an HADA that employs varying NPUs with different computational precision (e.g., 8-, 7-, and 6-bit), thus resulting in small accuracy drop, but with significant gains in latency and energy as trade-offs.

A. System Model

A smart IoT environment is considered consisting of $|A|$ Areas of Interest (AoIs), as shown in Fig. 1, and their set is denoted as $A = \{1, \ldots, a, \ldots, |A|\}$. At each AoI a, a number of IoT nodes $|U_a|$ resides and their set is denoted as $U_a = \{1, \ldots, u, \ldots, |U_a|\}$. The overall set of IoT nodes

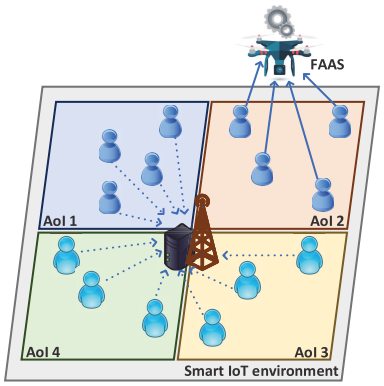

Fig. 1. Network Topology.

in the examined smart IoT environment is $U = \bigcup_{\forall a \in A} U_a = \{1, \ldots, u, \ldots, |U|\}$. A MEC server resides at the center of the overall topology serving the IoT nodes' computing demands, and a FAAS hovers above the AoIs to provide complementary computing capacity, acting as a FAAS-mounted computing server. The overall system is studied for a number of time slots $|T|$, where $T = \{1, \ldots, t, \ldots, |T|\}$ denotes their set.

Each IoT node has a DNN-based computing task $\mathcal{T} = \{B_{ua}^{(t)}, \phi_{ua}^{(t)}\}$ per time slot t that needs to process. The computing task is defined by the amount of data $B_{ua}^{(t)}$ [frames] that should be processed, and its computing intensity $\phi_{ua}^{(t)}$ [Computing Cycles] depending on the DNN-based application that the IoT node serves. Further, each IoT node has a minimum latency requirement $l_{ua}^{(t)}$ [sec] and energy consumption constraint $e_{ua}^{(t)}$ [J] depending on the IoT application that the node supports. Thus, if the IoT node experiences a latency and energy consumption greater than its minimum constraints, it is characterized as *dissatisfied*, because it cannot successfully support the IoT application. Every IoT node can offload part of its computing task's data to the MEC server or to the FAAS, if the latter one hovers above the node's AoI. The percentage of offloaded data is denoted as $\mathcal{A}_{ua}^{(t)} \in [0, 1]$, and the choices are discretized, e.g., $[0\%, 10\%, 20\%, \ldots, 100\%]$. Each IoT node makes a data offloading decision $d_{ua}^{(t)} = \{\delta_{ua}^{(t)}, \mathcal{A}_{ua}^{(t)}\}$ at each time slot t regarding the amount of data $\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}$ that it will offload to the MEC server ($\delta_{ua}^{(t)} = 0$) or to the FAAS ($\delta_{ua}^{(t)} = 1$), if the latter one is at the AoI that the IoT node resides. The IoT node processes $\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}$ data at the edge (i.e., MEC or FAAS), while the rest amount of data, i.e., $(1 - \mathcal{A}_{ua}^{(t)}) B_{ua}^{(t)}$ is processed locally at the IoT node. In the following, we present the IoT nodes' experienced latency and energy consumption, as an outcome of their data offloading decisions considering a FAAS with our HADA. The impact of our employed HADA on the accuracy, latency, and energy consumption of NN-based workloads is analyzed in the following subsection.

Initially, focusing on the *communications characteristics* of the considered smart IoT environment, the non-orthogonal multiple access (NOMA) technique is adopted by the IoT nodes to offload their data to the MEC server or the FAAS. Each IoT node transmits with power $P_u^{(t)}$ [W] its data during the data offloading to its receiver (i.e., MEC server or FAAS) and its corresponding channel gain is denoted as $g_u^{(t)}$. Given that the power control problem is not the main focus of our paper, we have considered that each IoT node u transmits with a power $P_u^{(t)} \in [0, 1]$ [W]. Specifically, the users that are characterized with good and bad channel gains transmit with relatively lower power, compared to the users that are characterized with an intermediate value of their channel gain. The latter formulation follows the principles of the power-domain NOMA in the uplink communication. Without loss of generality and for presentation purposes, we assume that the channel gains are sorted by the receiver, i.e., $g_1^{(t)} > \cdots > g_{|U|}^{(t)}$, hence, the sensed interference by each IoT node is $I_u^{(t)} = \sum_{u' \geq u+1}^{|U|} P_{u'}^{(t)} g_{u'}^{(t)}$ when offloading its data to the MEC server, i.e., $u = \{u \in U | \delta_{ua}^{(t)} = 0\}$, and $I_u^{(t)} = \sum_{u' \geq u+1}^{|U_a|} P_{u'}^{(t)} g_{u'}^{(t)}$, when the IoT node offloads its data to the FAAS, i.e., $u = \{u \in U_a | \delta_{ua}^{(t)} = 1\}$. The above analysis holds true given that the Successive Interference Cancellation (SIC) technique is implemented at the receiver [29]. The IoT node's achievable data rate to offload its data to the receiver (MEC server or FAAS) is:

$$R_u^{(t)} = W \cdot \log_2 \left(1 + \frac{P_u^{(t)} g_u^{(t)}}{I_u^{(t)} + \sigma^2} \right) \tag{1}$$

where W [Hz] denotes the IoT system's bandwidth, and σ^2 is the power of zero-mean Additive White Gaussian Noise (AWGN). Thus, the IoT node's experienced latency due to data offloading is $\frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}}{R_{ua}^{(t)}}$ [sec], and the corresponding energy consumption is $P_u^{(t)} \cdot \frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}}{R_{ua}^{(t)}}$ [J].

Furthermore, focusing on the *computing characteristics* of the examined smart IoT environment, the data processing frequencies of the IoT node, the MEC server, and the FAAS are denoted as f_u, f_M, f_F [Computing Cycles/sec], respectively. The IoT nodes employ small embedded NPUs, while the MEC server utilizes a strong and power-hungry DNN accelerator with computing capacities ϕ_u and ϕ_M [Computing Cycles] accordingly. Both the IoT nodes and the MEC sever execute the DNN at full precision (exact computing). On the contrary, the FAAS is equipped with an HADA which comprises varying NPUs that process the DNNs with different computational precision, i.e., 8-, 7-, or 6-bit. More details are provided in Section II-B. Thus, based on computational precision utilized on the FAAS, its corresponding computing capacity is $\phi_F = \{\phi_{8-bit}, \phi_{7-bit}, \phi_{6-bit}\}$ [Computing Cycles], where $\phi_{8-bit} > \phi_{7-bit} > \phi_{6-bit}$. Moreover, each node has the option to declare the acceptable level of computing from the FAAS (maximum accuracy drop), while offloading its data. Each node can declare its preference for no approximation (exact computing), 0.5%, or 1% maximum accuracy drop. For example, an IoT node that supports an IoT application that can tolerate an error in its computations, but has stringent latency constraints, it can select higher approximation (i.e., execution with lower precision) in order to process its data faster. Therefore, the computing capacity that is allocated to each IoT node in order to process its offloaded data at the

edge, is given as follows:

$$\Phi_{ua}^{(t)} = \frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \phi_{ua}^{(t)}}{\sum\limits_{\substack{u' \in U_a \\ j=\{8-bit,7-bit,6-bit\} \\ \delta_{u'a}^{(t)}=\delta_{ua}^{(t)}}} \mathcal{A}_{u'a}^{(t)} B_{u'a}^{(t)} \phi_{u'a}^{(t)}} \left[\left(1-\delta_{ua}^{(t)}\right)\phi_M + \delta_{ua}^{(t)}\phi_j \right] \tag{2}$$

The physical meaning of Eq. 2 follows the concept of proportional fairness, i.e., the IoT nodes share the edge computing capacity in a fair manner among them, without discriminating or prioritizing the computing tasks of the nodes. Specifically, the IoT nodes that offload part of their data to the MEC server, i.e., $\delta_{ua}^{(t)} = 0$, they proportionally share the MEC server's computing capacity ϕ_M based on the amount of their offloaded data, i.e., $\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}$, and their computing intensity $\phi_{ua}^{(t)}$. Also, in the case that the IoT nodes offload their data to the FAAS, i.e., $\delta_{ua}^{(t)} = 1$, the nodes that request the same level of computing (i.e., $j = \{8-bit, 7-bit, 6-bit\}$), they proportionally share the FAAS corresponding computing capacity ϕ_j.

Also, the device of each IoT node has a local computing capacity $\phi_{ua}^{(t)}$ [Computing Cycles] to process locally the remaining data $(1-\mathcal{A}_{ua}^{(t)})B_{ua}^{(t)}$. The local processing delay for each IoT node is $\frac{(1-\mathcal{A}_{ua}^{(t)})B_{ua}^{(t)}}{f_u}$ [sec], while the corresponding energy consumption to process the remaining data locally is $(1-\mathcal{A}_{ua}^{(t)})B_{ua}^{(t)}\epsilon_{ua}$ [J], where ϵ_{ua} [J/Computing Cycles] is the IoT node's local energy consumption per Computing Cycle.

The data processing delay that each IoT node experiences is $\frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \Phi_{ua}^{(t)}}{f_F}$ at the FAAS, and $\frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \Phi_{ua}^{(t)}}{f_M}$ at the MEC server. Thus, considering that the IoT node's data are processed in parallel locally and at the edge, the overall experienced latency is given as follows.

$$L_{ua}^{(t)} = \max\left\{ \frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}}{R_{ua}^{(t)}} + \mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \Phi_{ua}^{(t)} \left[\frac{\delta_{ua}^{(t)}}{f_F} + \frac{1-\delta_{ua}^{(t)}}{f_M} \right], \frac{\left(1-\mathcal{A}_{ua}^{(t)}\right) B_{ua}^{(t)} \phi_{ua}^{l}}{f_u} \right\} \text{[sec]} \tag{3}$$

Also, the overall energy consumption of each IoT node is given as follows:

$$E_{ua}^{(t)} = P_u^{(t)} \cdot \frac{\mathcal{A}_{ua}^{(t)} B_{ua}^{(t)}}{R_{ua}^{(t)}} + \left(1-\mathcal{A}_{ua}^{(t)}\right) B_{ua}^{(t)} \cdot \phi_{ua}^{l} \cdot \epsilon_{ua} \text{[J]} \tag{4}$$

Each IoT node will be satisfied and successfully serve its IoT application, if its latency, i.e., $L_{ua}^{(t)} \leq l_{ua}^{(t)}$, and energy constraints, i.e., $E_{ua}^{(t)} \leq e_{ua}^{(t)}$, are satisfied per time slot *t*.

B. Computing Architecture for Using Approximate Computing

In this section, we present the Heterogeneous Approximate DNN Accelerator (HADA) architecture utilized on the FAAS in order to support approximate computing of the offloaded DNNs. As aforementioned, NPUs are ubiquitous in modern edge systems to accelerate the increased DNN-based workloads. In our work, we consider a microarchitecture similar to Google Edge TPU [30] that comprises a 64 × 64 systolic multiply-accumulate (MAC) array, able to perform millions of operations per inference. In this paper, we propose an HADA based on the numerical precision of the NPUs (e.g., 7-, 6-bit).

The strong advantage of reducing the bit-width is the fact that the area and the power consumption of the NPU are significantly reduced. Contrary to previous methods that utilize bit-precision [22], we do not deactivate bits, but we design the whole MAC array with lower precision. Consequently, we leverage more the obtained power and area gain from approximation (w.r.t. the 8-bit baseline) to incorporate a larger number of MAC units. This, in turn, provides more acceleration to the deployed DNNs. To that end, we design and synthesize NPUs operating at 7- and 6-bit. In order to conduct a fair evaluation, our developed NPUs have almost the same area (negligible size overhead of only 5%) compared to the baseline 8-bit 64 × 64 systolic MAC array. However, reducing the number of bits can result in significant accuracy loss. Thus, in order to utilize the NPUs on the FAAS and still keep DNN accuracy high, we employed run-time low bit-width post-training quantization based on the method presented in [31]. Originally, this method does not quantize the first, the last, and the pooling layers. To that end, we significantly modified it in order to still employ the concept of analytical clipping method, introduced in [31], but for all layers. For our IoT deployment, this is important as all layers of a DNN will be executed on the same NPU on the FAAS to avoid resource congestion and run-time overheads.

To demonstrate the benefits of the low-precision NPUs mounted on the FAAS and their importance on our IoT deployment, Table I shows the average power gain, the size overhead, the raw throughput gains, and the corresponding accuracy drop (error due to approximation) for multiple different state-of-the-art DNNs. The baseline NPU for our comparison is the 8-bit 64 × 64 systolic MAC array presented in [30]. As raw throughput we define the maximum number of computational instructions that can be executed per cycle. Synopsys EDA tools and the 14nm technology node [6] are used to obtain the area and power of the NPUs, while SCALE-Sim, cycle-accurate CNN simulator from ARM [32], was used to capture the Computing Cycles. The reported accuracy is for the state-of-art ImageNet dataset [33]. As Table I presents, for 7- and 6-bit NPUs, the number of integrated MAC units has increased, while at the same time the power gains are significant. The increased number of MAC units in the approximate NPUs, increases the attained throughput and decreases the computing cycles. The average gain due to reduced bit precision is on average 7.5% and 17.2% for 7- and 6-bit respectively. Overall, Table I shows our strong motivation for utilizing approximate computing on the FAAS.

III. Energy Efficient Data Offloading & Processing

In this section, we initially define the IoT nodes' utility function to capture their minimum QoS requirements, i.e., latency and energy consumption constraints, following the paradigm of Satisfaction Games. Then, the IoT nodes' optimal data offloading problem to the MEC server or the FAAS is

TABLE I
POWER, SIZE, THROUGHPUT, AND ACCURACY ANALYSIS OF LOW BIT-WIDTH NPUS COMPARED TO THE BASELINE NPU (8-BIT 64 × 64)

	NPU Design Characteristics				Accuracy drop of multiple DNNs under low precision			
bit-width	**# MAC units**	**Power gain**	**NPU size overhead**	**Throughput gain**	**ResNet34**	**ResNet50**	**VGG16**	**WideresNet50**
7-bit	72×72	28%	5%	27%	0.04%	0.20%	0.04%	0.12%
6-bit	80×80	24%	5%	56%	0.70%	0.90%	0.53%	0.90%

captured as a Satisfaction Game among the IoT nodes and the Satisfaction Equilibrium (SE) and Generalized SE (GSE) are determined via a distributed learning algorithm.

A. IoT Node's Utility Function

Each IoT node has some personalized latency $l_{ua}^{(t)}$ and energy consumption $e_{ua}^{(t)}$ requirements, which jointly define its minimum QoS prerequisites. Both QoS requirements must be simultaneously satisfied in order for the IoT node to enjoy a non-negative utility (Eq. 5c) and support its IoT application. In the opposite case, the IoT node remains dissatisfied if both (Eq. 5b) or even one of its QoS requirements is not fulfilled, and it experiences a negative utility (Eq. 5c). Based on the above analysis, the nodes utility function is defined as follows.

$$U_{ua}^{(t)}(d_{ua}^{(t)}, \mathbf{d}_{-ua}^{(t)}) = \begin{cases} -\left(\dfrac{l_{ua}^{(t)} - L_{ua}^{(t)}}{l_{ua}^{(t)}}\right)\left(\dfrac{e_{ua}^{(t)} - E_{ua}^{(t)}}{e_{ua}^{(t)}}\right), & \text{(5a)} \\ \text{if } l_{ua}^{(t)} \leq L_{ua}^{(t)} \text{ and } e_{ua}^{(t)} \leq E_{ua}^{(t)} & \text{(5b)} \\ \left(\dfrac{l_{ua}^{(t)} - L_{ua}^{(t)}}{l_{ua}^{(t)}}\right)\left(\dfrac{e_{ua}^{(t)} - E_{ua}^{(t)}}{e_{ua}^{(t)}}\right), \text{else} & \text{(5c)} \end{cases}$$

It is noted that $d_{ua}^{(t)} = \{\delta_{ua}^{(t)}, \mathcal{A}_{ua}^{(t)}\}$ denotes the IoT node's u decision to offload its data, and $\mathbf{d}_{-ua}^{(t)}$ denotes the decision vector of all the other IoT nodes except for node u. The decision set of each IoT node is denoted as $D_{ua}^{(t)}$, where $d_{ua}^{(t)} \in D_{ua}^{(t)}$. If the FAAS is in another AoI other than the AoI a, then the strategy set is defined as $D_{ua}^{(t)} = \{(0, \mathcal{A}_{ua}^{(t)}|_1), \ldots, (0, \mathcal{A}_{ua}^{(t)}|_n), \ldots, (0, \mathcal{A}_{ua}^{(t)}|_N)\}$, where N denotes the levels of discretization of the data offloading percentages. In the opposite case, the IoT nodes can either offload part of their data to the MEC server or the FAAS, thus, their strategy set is defined as $D_{ua}^{(t)} = \{(0, \mathcal{A}_{ua}^{(t)}|_1), \ldots, (0, \mathcal{A}_{ua}^{(t)}|_N), (1, \mathcal{A}_{ua}^{(t)}|_1), \ldots, (1, \mathcal{A}_{ua}^{(t)}|_N)\}$.

B. Problem Formulation

Given the IoT nodes' utility function, our goal is to determine each IoT node's optimal partial data offloading strategy $d_{ua}^{(t)*}$ to the edge computing available options to satisfy its minimum QoS prerequisites. Also, given that the edge computing options are shared among the IoT nodes, we capture their interactions via a non-cooperative satisfaction game, defined as $G = [U, \{D_{ua}^{(t)}\}_{\substack{\forall u \in U \\ \forall a \in A}}, \{U_{ua}^{(t)}\}_{\substack{\forall u \in U \\ \forall a \in A}}, \{S_{ua}(\mathbf{d}_{-ua}^{(t)})\}_{\substack{\forall u \in U \\ \forall a \in A}}]$, where U is the set of all the IoT nodes, $D_{ua}^{(t)}$ denotes each node's strategy set, $U_{ua}^{(t)}$ is its utility, as defined in Eq. 5, and $S_{ua}(\mathbf{d}_{-ua}^{(t)}) = \{d_{ua}^{(t)} \in D_{ua}^{(t)} | U_{ua}^{(t)} \geq 0\}$ expresses the satisfaction correspondence.

The IoT nodes' goal is to determine a strategy that will satisfy their minimum QoS prerequisites. This strategy is the Satisfaction Equilibrium, as defined below.

Definition 1 [Satisfaction Equilibrium (SE)]: The strategy vector $\mathbf{d}^* = [d_{1a}^*, \ldots, d_{ua}^*, \ldots, d_{|U|a}^*], \forall a \in A$ is a Satisfaction Equilibrium of the non-cooperative game G, if $\forall u \in U, \mathbf{d}^* \in S_{ua}(\mathbf{d}_{-ua}), \forall a \in A$.

It is noted that the game G may have multiple SEs or an SE may not exist due to the limited available computing resources. In the latter case, some of the IoT nodes may satisfy their minimum QoS prerequisites and some of them may not. In this case, a Generalized Satisfaction Equilibrium (GSE) should be determined. It is noted that in our analysis, we refer to the SE and the GSE as the optimal strategy that the IoT nodes can perform in order for all the nodes or part of them, respectively, to satisfy their minimum QoS prerequisites. It is highlighted that the paradigm of Satisfaction Games aims to satisfy a minimum QoS constraint, and not to maximize the user's utility or the system's welfare, as compared to the Network Utility Maximization (NUM) theory [26].

C. Problem Solution

In this section, we present the detailed theoretical steps in the format of a distributed learning algorithm to determine an SE for all the IoT nodes, if it exists, or a Generalized SE, if the computing resources are not sufficient to satisfy the minimum QoS requirements of all the nodes. The distributed learning algorithm is executed at each time slot t to determine the SE or the GSE.

Distributed Learning Algorithm (DLA):

1) At the first iteration $i = 0$, each IoT node selects a strategy $d_{ua}^{(t)}|_{i=0}$ with equal probability $P_{ua}^{(t)}|_{i=0} = \frac{1}{2N}$ if the FAAS flies above its AoI, or $P_{ua}^{(t)}|_{i=0} = \frac{1}{N}$, if not. Then, it determines its achieved utility $U_{ua}^{(t)}$, based on Eq. 5, by receiving the information of the allocated computing capacity $\Phi_{ua}^{(t)}$ (Eq. 2) and the sensed interference $I_u^{(t)}$ from the FAAS or the MEC server, depending on where it offloads its data. If $U_{ua}^{(t)} \geq 0$, then $d_{ua}^{(t)} \in S_{ua}(\mathbf{d}_{-ua})$, thus, $d_{ua}^{(t)} = d_{ua}^{(t)*}$ is a satisfactory strategy, and the IoT node keeps it for the rest of the iterations of the algorithm. If $U_{ua}^{(t)} < 0$, continue to the next step.
2) The IoT nodes that have not already selected a strategy which satisfies their minimum QoS requirements, update their probabilities of selecting an alternative strategy as

follows.

$$P_{ua}^{(t)}\left(d_{ua}^{(t)}|_i\right) = P_{ua}^{(t)}\left(d_{ua}^{(t)}|_{i-1}\right) - \frac{1}{i+1} \cdot b \cdot P_{ua}^{(t)}\left(d_{ua}^{(t)}|_{i-1}\right), \text{ if } d_{ua}^{(t)}|_i = d_{ua}^{(t)}|_{i-1} \quad (6a)$$

$$P_{ua}^{(t)}\left(d_{ua}^{(t)}|_i\right) = P_{ua}^{(t)}\left(d_{ua}^{(t)}|_{i-1}\right) + \frac{1}{i+1} \cdot b\left(\frac{1}{N-1} - d_{ua}^{(t)}|_{i-1}\right), \text{ if } d_{ua}^{(t)}|_i \neq d_{ua}^{(t)}|_{i-1}, |D_{ua}^{(t)}| = N \quad (6b)$$

$$P_{ua}^{(t)}\left(d_{ua}^{(t)}|_i\right) = P_{ua}^{(t)}\left(d_{ua}^{(t)}|_{i-1}\right) + \frac{1}{i+1} \cdot b\left(\frac{1}{2N-1} - d_{ua}^{(t)}|_{i-1}\right), \text{ if } d_{ua}^{(t)}|_i \neq d_{ua}^{(t)}|_{i-1}, |D_{ua}^{(t)}| = 2N \quad (6c)$$

and perform a new strategy selection.

The physical meaning of the above probabilities' update rule is that the learning rate $\frac{1}{1+i}$ will enable the IoT nodes to update their probabilities over the iterations of the distributed learning algorithm, towards obtaining higher probability to select a strategy that has the potential to provide a higher utility.

3) If convergence is not achieved, then return to step 2.

Based on the above description, it is obvious that if there exists at least one satisfactory strategy $d_{ua}^{(t)*}$ for each IoT node $u \in U$, the proposed distributed learning algorithm will eventually determine it after performing exploration. However, there can be the case that the proposed algorithm cannot converge to an SE either due to the limited computing resources of the overall system or due to the selected strategies of the IoT nodes. In order to explore this case, let us provide the definition of the clipping action.

Definition 2 (Clipping Action): At the non-cooperative game *G*, an IoT node has a clipping action $d_{ua}^{(t)c} \in D_{ua}^{(t)}$ if $\forall \mathbf{d}_{-ua}, d_{ua}^{(t)c} \in S_{ua}$.

Based on the above definition, if the distributed learning algorithm described above, concludes at a state (step 2) that at least one IoT node has a clipping action and one IoT node has satisfaction correspondence $S_{ua}(\mathbf{d}_{-ua}) = \varnothing, \forall \mathbf{d}_{-ua}$, then the nodes with a clipping action will select it, as the selection probability of this action is strictly increasing. Thus, the IoT node with satisfaction correspondence $S_{ua}(\mathbf{d}_{-ua}) = \varnothing$ will never be satisfied. Thus, the distributed learning algorithm will converge to a Generalized Satisfaction Equilibrium (GSE) and its definition is provided below.

Definition 3 (Generalized Satisfaction Equilibrium): A strategy vector $\mathbf{d}_{-ua}, \forall u \in U, \forall a \in A$ is a GSE for the non-cooperative gave *G*, if two sets U^{sat} and U^{unsat} exist, such that $d_{ua}^{(t)} \in S_{ua}(\mathbf{d}_{-ua}), \forall u \in U^{sat}$ and $S_{u'a}(\mathbf{d}_{-u'a}) = \varnothing, \forall u' \in U^{unsat}$.

Concluding the above analysis, it is observed that if there is no clipping action, the distributed learning algorithm will converge to an SE. On the other hand, if at least one IoT node has a clipping action, the algorithm will converge to a GSE. The complexity of the Distributed Learning Algorithm (DLA) is $O(|U| \cdot Ite)$, where *Ite* is the total number of iterations in order the algorithm to converge to the SE or to the GSE.

IV. Fully Autonomous Aerial System's Movement Based on Reinforcement Learning

In this section, we exploit the set of gradient ascent reinforcement learning algorithms in order to introduce a fully autonomous trajectory planning to serve the IoT nodes' computing needs and sparingly use its limited available energy. Based on the introduced gradient ascent reinforcement learning, the FAAS interacts with the environment and learns which AoI to visit per time slot. Following the principles of reinforcement learning, the FAAS acts as an agent, it takes an action regarding the AoI that should visit, it performs the action, and receives a corresponding reward from the environment that interacts with as IoT nodes process their computing tasks. The FAAS learns its most beneficial actions in order to optimize its long-term reward by interacting with the IoT environment and supporting the IoT nodes computing demand. For practical purposes, we consider that the time slot duration τ is sufficient to allow the FAAS to move from one AoI to another based on its velocity *v* [m/sec]. Following the theory of the gradient ascent reinforcement learning, the FAAS acts as a Learning Automaton, and at each time slot updates its probability to visit an AoI based on the reward received by the environment. The reward of the FAAS by visiting an AoI *a* is defined as follows,

$$r_a^{(ite)} = \frac{\frac{|U_a^{sat}|^{(ite)}}{|U_a|}\left[\sum_{\substack{\forall u \in U_a \\ \delta_{ua}^{(t)}=1 \\ \forall j \in \{ex,0.5,1\}}} \mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \phi_j / \sum_{\forall u \in U_a} \mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \phi_{ua}^{(t)}\right]^{(ite)}}{w_1 E_m^{(ite)} + w_2 E_{proc}^{(ite)}} \quad (7)$$

where *ite* denotes the iteration of the gradient ascent algorithm, $E_m^{(ite)} = E_M \frac{d_{aa'}}{v}$ [J] is the FAAS consumed energy to fly from AoI *a* to a', E_M [J/s] is the FAAS energy consumption when it flies with constant velocity *v* [m/s] and $d_{aa'}$ is the distance among the AoI *a* and a'. Also, $E_{proc}^{(ite)}$ [J] denotes the FAAS energy consumption to process the IoT nodes' offloaded data, as defined by its approximate computing characteristics, and $w_1, w_2 \in [0, 1]$ denotes the weighting factors of the FAAS energy consumption to fly among the AoIs and process the IoT nodes' offloaded data, respectively.

The physical meaning of Eq. 7 is that the FAAS experiences greater reward $r_a^{(ite)} \in [0, 1]$, when it visits an area and achieves to contribute to the satisfaction of a large portion of IoT nodes, i.e., $\frac{|U_a^{sat}|^{(ite)}}{|U_a|}$, via processing a large portion of their data, i.e.,

$$\left[\sum_{\substack{\forall u \in U_a \\ \delta_{ua}^{(t)}=1 \\ \forall j \in \{ex,0.5,1\}}} \mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \phi_j / \sum_{\forall u \in U_a} \mathcal{A}_{ua}^{(t)} B_{ua}^{(t)} \phi_{ua}^{(t)}\right]^{(ite)}.$$

On the other hand, the FAAS experiences low reward if it travels a large distance to visit an AoI, i.e., $w_1 E_m^{(ite)}$, and if it spends a large amount of energy to process the IoT nodes' offloaded data, i.e., $w_2 E_{proc}^{(ite)}$. Furthermore, we normalize the reward from each AoI *a*, as $\hat{r}_a^{(ite)} = \frac{r_a^{(ite)}}{\sum_{\forall a \in A} r_a^{(ite)}}$ to express

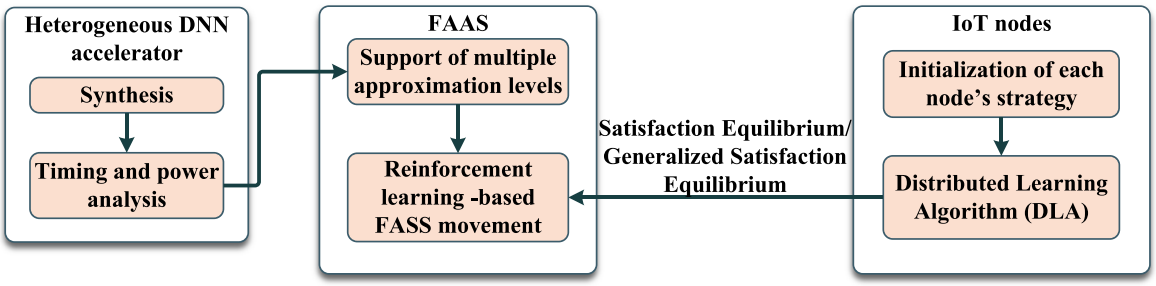

Fig. 2. Overview: Energy Efficient Edge Computing Enabled by Satisfaction Games and Approximate Computing.

the relative reward of the FAAS by visiting an AoI as compared to all the available AoIs that it can potentially visit. The FAAS updates its probability to visit the same (Eq. 8a) or a different AoI (Eq. 8b) at each iteration of the gradient ascent reinforcement learning algorithm based on the following action probability rules.

$$P_a^{(ite+1)} = P_a^{(ite)} + \lambda_1 \hat{r}_a^{(ite)} \left(1 - P_a^{(ite)}\right) - \lambda_2 \left(1 - \hat{r}_a^{(ite)}\right) P_a^{(ite)}, \text{ if } a^{(ite+1)} = a^{(ite)} \quad (8a)$$

$$P_a^{(ite+1)} = P_a^{(ite)} - \lambda_1 \hat{r}_a^{(ite)} P_a^{(ite)} + \lambda_2 \left(1 - \hat{r}_a^{(ite)}\right) \left(\frac{1}{|A| - 1} - P_a^{(ite)}\right), \text{ if } a^{(ite+1)} \neq a^{(ite)} \quad (8b)$$

The action probabilities updating rule presented in Eq. 8a and Eq. 8b allows the FAAS to thoroughly explore all the available AoIs for $\lambda_1 >> \lambda_2, \lambda_1, \lambda_2 \in [0, 1]$ to select the most beneficial one in terms of improved received reward. For $\lambda_1 >> \lambda_2$, the RL algorithm is called Linear Reward-ϵ Penalty (LR-ϵP). In the other special case that $\lambda_2 = 0$, the FAAS always selects probabilistically the AoI a that provides the highest reward, while performing very limited exploration of other alternative AoIs. The RL algorithm is called Linear Reward Inaction (LRI) for $\lambda_2 = 0$. It is noted that both the LRI and LR-ϵP run iteratively per time slot at the FAAS to determine the optimal AoI to be visited at the next time slot. The number of satisfied IoT nodes $|U_a^{sat}|^{(ite)}$ is the only information that needs to be collected from the field and can be retrieved with a single bit information message from the IoT nodes. The complexity of the gradient ascent reinforcement learning algorithms is $O(ITE)$, where ITE is the total number of iterations for the algorithm to converge to the a stable AoI selection. Detailed numerical results showing the FAAS trajectory planning and a detailed comparison among the LRI and LR-ϵP reinforcement learning algorithms are presented in Section V. The overview of the overall proposed framework, and the relationship and interaction among the different modules, are presented in Fig. 2.

V. Experimental Evaluation

In this section, a detailed numerical evaluation is presented to study the performance and the inherent characteristics of the proposed energy efficient edge computing framework enabled by the satisfaction games and the approximate computing. Initially, in Section V-A, we present the benefits of approximation and the utilization of the proposed HADA on the FAAS. Then, the pure operation performance of the proposed framework is demonstrated in an IoT deployment, in Section V-B, both from the IoT nodes' and the FAAS perspective. Also, the benefits of the reinforcement learning-based trajectory planning are demonstrated in Section V-C. Finally, a scalability analysis is provided to show the efficiency and robustness of the proposed framework in Section V-D, as well as a detailed comparative analysis among multiple different DNNs, showing the benefits of our approach in a variety of different workloads.

A. Benefits of Approximation

In this section, we present the benefits of the proposed HADA for DNN execution on the FAAS and the added value that approximate computing brings. To that end, we consider multiple state-of-art DNNs and a single IoT node in four cases: (1) the DNN is executed on the IoT node's device; (2) the DNN is executed on the FAAS, requiring no approximation; (3) the DNN is executed on the FAAS, having 0.5% maximum accuracy drop tolerance; and (4) the DNN is executed on the FAAS, having 1% maximum accuracy drop tolerance. All the selected DNNs follow different algorithmic approaches. In that way, we managed to evaluate our approach on different scenarios. Additionally, we considered that the edge IoT node is equipped with a 32x32 NPU (i.e., similar computing power to the Samsung mobile NPUs [18]). Moreover, the FAAS is equipped with an accelerator that integrates three NPUs: (1) an 8-bit NPU, with 64 $\times$ 64 MAC units, utilized for exact computations; (2) a 7-bit NPU, with 72 $\times$ 72 MAC units, utilized when the IoT node can tolerate 0.5% accuracy drop; and (3) a 6-bit NPU, with 80 $\times$ 80 MAC units, utilized when the IoT node can tolerate 1.0% accuracy drop.

Figs. 3a–3b show the normalized execution time and processing energy consumption respectively during inference, of five different DNNs, under the four aforementioned cases. The inference execution time was extracted from the SCALE-Sim cycle-accurate CNN simulator [32], while the energy consumption was calculated after a chip design using Synopsys EDA tools at 14nm technology. Clearly, the accelerator mounted on the FAAS significantly improves the execution time of all DNNs by an average of 50%. However, the computation energy consumption overhead for DNN processing on the FAAS is greater by approximately 43%, if no accuracy drop can be tolerated (exact computation). If the IoT node can

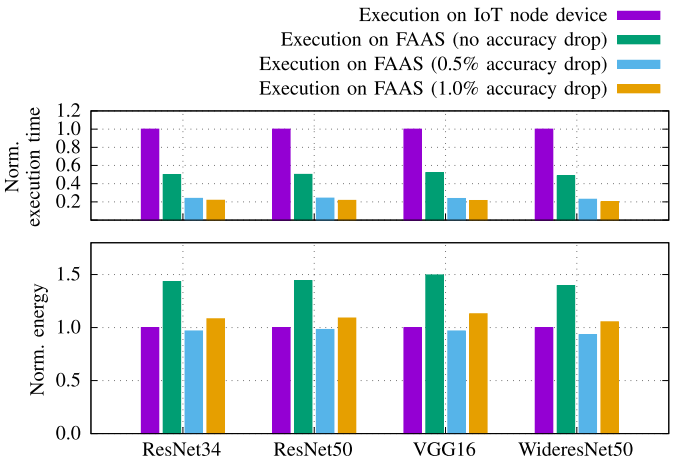

Fig. 3. (a) Normalized computing cycles of different DNNs for the IoT node's device and the NPUs employed on the FAAS. (b) Corresponding normalized energy consumption. The values in both figures are normalized over the IoT node's device.

tolerate a negligible accuracy drop of 0.5%, then the execution of the DNN on the FAAS is completed 77% faster, while the processing energy consumption on the FAAS is 5% lower than the IoT node's device. Furthermore, if the IoT node can tolerate 1% accuracy drop, then the gains in execution time increases to 79%, while the energy consumption is almost similar with the IoT node's device. Overall, in this section we presented the benefits of utilizing approximation and how we can achieve significant gains if we introduce negligible error in the calculation.

B. Pure Operation Performance: IoT Nodes and FAAS Perspective

In this section, we aim to elucidate the operational characteristics of the proposed energy efficient edge computing framework based on satisfaction games and approximate computing. Our use case evaluation is based on the ResNet50 DNN. We selected it as it constitutes the core DNN for many applications [34].

Particularly, we consider an IoT environment with $|A| = 8$ AoIs, and $|U_a| = 15$ IoT nodes in each AoI. Also, we have $B_{ua}^{(t)} = [25, 30]$ [frames], $\phi_{ua}^{(t)} = 6007234$ [Computing Cycles], the distance among the IoT nodes and the MEC server $d_{Mu} \in [1900, 3000]$ [m], the distance among the IoT nodes and the FAAS $d_{Fu} \in [120, 180]$ [m], $g_u = \frac{1}{d_{F/Mu}^2}$, $\sigma^2 = 10^{-13}$, $P_u^{(t)} \in [0, 1]$ [W], $f_u = 500 \times 10^6$ [Computing Cycles/sec], $f_M = 1000 \times 10^6$ [Computing Cycles/sec], $f_F = 700 \times 10^6$ [Computing Cycles/sec], $\phi_M = 897881$ [Computing Cycles], $E_m^{(ite)} = 1.55 \times 10^{-8}$, $w_1 = 1$, $w_2 = 1$, $\lambda_1 = 0.7$, $\lambda_2 = 0.005$ for LR-ϵP algorithm and $\lambda_1 = 0.7$ for LRI algorithm, unless otherwise explicitly stated. For nodes with strict QoS constraints, we have $l_{ua}^{(t)} = 8 \times 10^{-2}$ [sec] and $e_{ua}^{(t)} = 2 \times 10^{-2}$ [J], for mild QoS constraints, $l_{ua}^{(t)} = 12 \times 10^{-2}$ [sec] and $e_{ua}^{(t)} = 2.1 \times 10^{-2}$ [J], and for weak QoS constraints, $l_{ua}^{(t)} = 16 \times 10^{-2}$ [sec] and $e_{ua}^{(t)} = 2.2 \times 10^{-2}$ [J]. Finally, the 15 IoT nodes reside at each AoI are organized in groups of 5 nodes in terms of requesting exact computing, 0.5%, and 1% maximum accuracy drop, respectively. The proposed framework's evaluation was conducted in an Asus Laptop with 2.10GHz AMD Ryzen 5 3550H processor and 8GB available RAM.

Figs. 4(a)–4(c) present the IoT nodes' percentage of offloaded data to the edge computing options (MEC server or FAAS), the normalized experienced latency, and the normalized energy consumption, respectively, as a function of the distributed Learning algorithm's (DLA) iterations. It is noted that the normalization of the latency and the energy consumption has been performed with respect to the corresponding value derived if the whole amount of the IoT nodes' data is processed locally on their devices. Additionally, the size of transmitted data is the same w.r.t. to the computing precision. The quantization is performed on the FAAS, thus the size of transmitted data is not affected. Each curve in Fig. 4(a)–4(c) presents the average values of each group of IoT nodes with homogeneous computing requests, i.e., exact computing, 0.5%, and 1% maximum accuracy drop (i.e., approximate computing).

Focusing on the IoT nodes' perspective, the results reveal that the more flexible the IoT nodes are in terms of dropping the computing accuracy, the more data they offload to the edge computing options (Fig. 4(a)), which support the fast (Fig. 4(b)) and energy efficient (Fig. 4(c)) data processing. In contrast, the IoT nodes that accept only exact computing of their data, they tend to keep approximately 40% of their data locally, to process them on their devices (Fig. 4(a)), as the trade-off of the data transmission time and processing time at the edge options is not favorable given the strict and time-consuming computing requirements. Thus, the IoT nodes, requesting exact computing for their computing tasks, experience a higher latency (Fig. 4(b)) and consume a larger amount of energy (Fig. 4(c)) to process their data. Furthermore, Fig. 4(d) presents the percentage of satisfied IoT nodes under the different computing requirements in terms of exact and approximate computing. Towards deriving those results, a Monte Carlo analysis has been executed for 1.000 executions of the overall framework. The results show that the stricter the IoT nodes' computing requirements are in terms of computing approximation, the less percentage of satisfied IoT nodes is achieved, given that it becomes more difficult for the smart IoT environment to satisfy their latency and energy constraints.

Continuing our analysis and focusing on the FAAS perspective, we consider four comparative scenarios for the same smart IoT environment, where all the IoT nodes request: 1) exact computing; 2) 0.5%; and 3) 1% maximum accuracy drop, or (iv) process their data locally to their devices. The results are derived by performing a Monte Carlo analysis with 1.000 executions of the overall framework, where at each execution the FAAS may hover above a different AoI. Figs. 5(a)–5(b) present the average normalized experienced latency and normalized energy consumption of the IoT nodes, respectively, for all the comparative scenarios. Also, Fig. 5(c) illustrates the percentage of satisfied IoT nodes for all the comparative scenarios. It is observed that the IoT nodes' worst latency (Fig. 5) and energy consumption (Fig. 5(b)) is

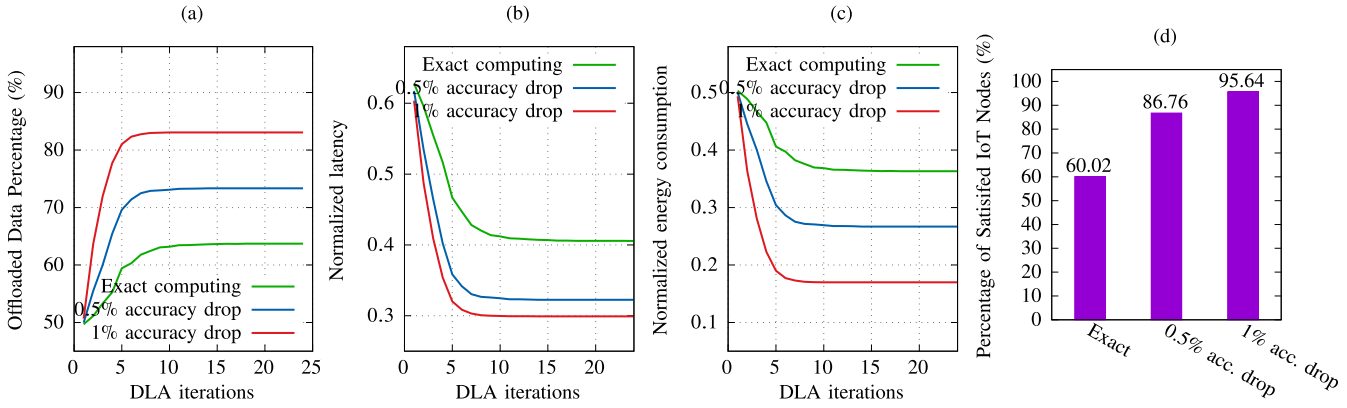

Fig. 4. IoT nodes' (a) percentage of offloaded data, (b) normalized experienced latency, (c) normalized energy consumption, and (d) percentage of satisfied IoT nodes considering different levels of computing approximation for the ResNet50 DNN.

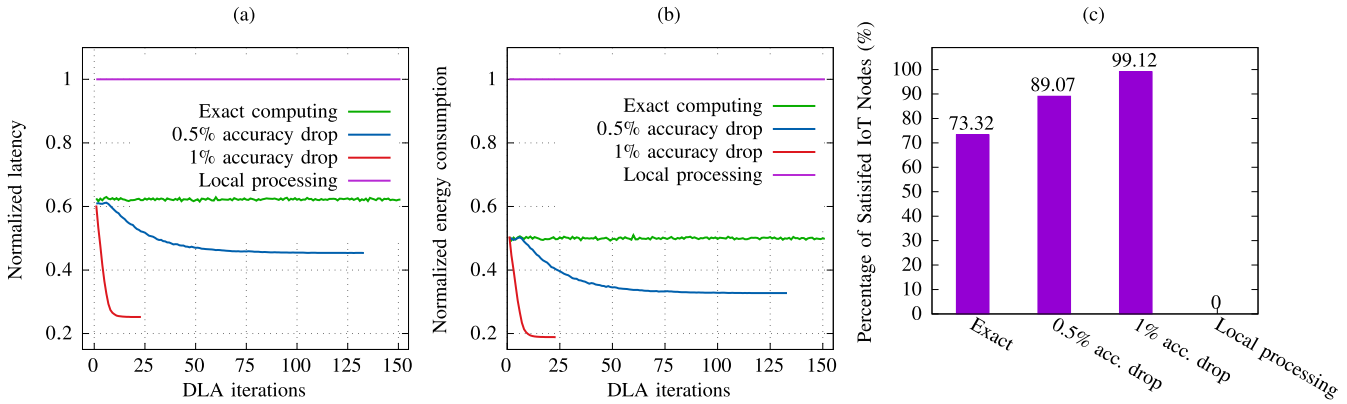

Fig. 5. IoT nodes' normalized experienced (a) latency and (b) energy consumption, and (c) percentage of satisfied IoT nodes under four comparative scenarios of computing approximation for the ResNet50 DNN.

experienced, when the nodes process all of their data locally, thus, they remain dissatisfied in terms of their latency and energy consumption requirements (Fig. 5(c)). Also, the results show that the more relaxed are the IoT nodes' computing approximation requirements, the less latency (Fig. 5(a)) and energy consumption (Fig. 5(b)) they experience, and the smart IoT environment achieves to satisfy a higher percentage of them (Fig. 5(c)). Also, focusing on the real time execution of the Distributed Learning Algorithm (DLA) that enables the IoT nodes' autonomous decision making in terms of data offloading and processing, we observe that few milliseconds are required to converge (Figs. 4(a)–4(c)). Thus, the proposed energy efficient edge computing framework can be implemented in a real time manner in a smart IoT environment.

C. Reinforcement Learning-Based FAAS Trajectory Planning

In this section, the benefits of adopting a reinforcement learning mechanism to enable the FAAS to fully autonomously determine its optimal trajectory, are presented. The two introduced reinforcement learning algorithms introduced in this paper, i.e., Linear Reward-ϵ Penalty (LR-ϵP) and Linear Reward Inaction (LRI), are also compared in terms of their time execution efficiency. The same topology of the smart IoT environment is adopted as described above, and we consider fixed computing requirements for the IoT nodes and no mobility in order to evaluate the pure benefits of the reinforcement learning mechanism.

Figs. 6(a)–6(b) present the convergence of the LRI and LR-ϵP algorithms to a stable selection of an AoI and the corresponding elapsed time, respectively. It is noted that at each iteration of the reinforcement learning mechanism, the FAAS makes an optimal AoI selection to visit and offer its computing services, it receives a reward (Eq. 7) and updates its probability to visit another AoI in the next iteration based on Eq. 8a and Eq. 8a. It is observed that if the smart IoT environment remains static, the FAAS ultimately will learn the optimal AoI selection after performing a thorough exploration and exploitation of its available choices. Additionally, it is highlighted that at each iteration, the FAAS makes an optimal choice of an AoI to visit based on the knowledge that it has gained so far by interacting with the smart IoT environment.

The results reveal that the LR-ϵP algorithm needs more time to converge to a stable AoI selection compared to the LRI one. This outcome is derived from the fact that the LR-ϵP performs a more thorough exploration of the available AoI choices, while the LRI enables the FAAS to probabilistically select the AoI that provides the highest reward. Thus, the LRI algorithm performs very limited exploration of the alternative AoIs and converges faster to a stable selection.

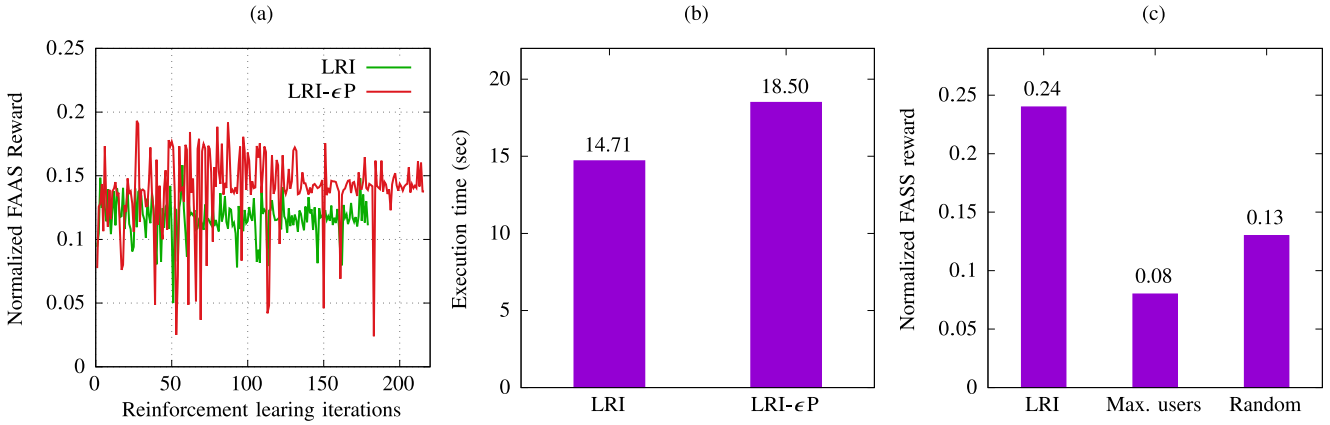

Fig. 6. Reinforcement Learning-based FAAS trajectory planning - A Comparative Evaluation.

Furthermore, Fig. 6(c) presents a comparative analysis among different AoI selection mechanisms implemented at the FAAS, where the FAAS selects an AoI based on: (i) the LRI algorithm, (ii) the maximum number of the IoT nodes residing in an AoI, and (iii) a random selection. A smart IoT environment is considered with 8 AoIs and [9, 16, 26, 10, 25, 24, 14, 20] IoT nodes residing at each AoI. A Monte Carlo analysis is performed with 1.000 executions of the overall framework. The results reveal that the reinforcement learning-based AoI selection by the FAAS concludes to higher achieved reward (Eq. 7) by the FAAS compared to all other scenarios. The detailed results showed that based on the reinforcement learning-enabled FAAS trajectory planning, the FAAS achieves to satisfy a greater portion of the IoT nodes by processing more of their data, while satisfying their latency and energy consumption requirements. Also, the reinforcement learning-based trajectory planning enables the FAAS energy saving in terms of moving among the AoIs to serve the IoT nodes' computing demands. In contrast, when the FAAS chooses to always visit the AoI with the maximum number of IoT nodes, the FAAS achieves the lowest reward, as a large number of IoT nodes remains dissatisfied in the rest of the AoIs.

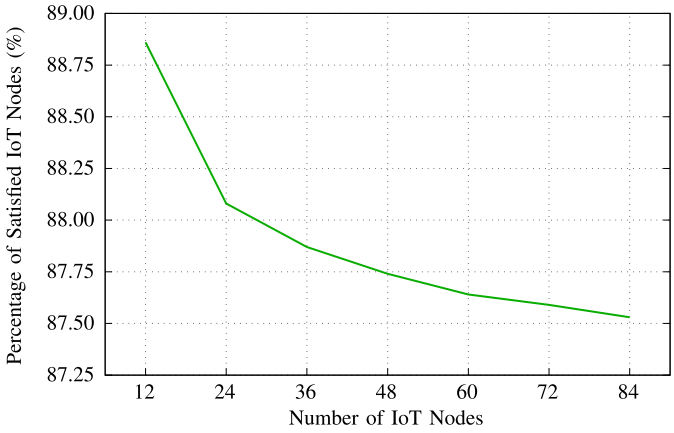

Fig. 7. Scalability Analysis for the ResNet50 DNN.

TABLE II
DIFFERENCE IN NUMBER OF PERFORMED COMPUTATIONS AND NUMBER OF SATISFIED USERS FOR OTHER DNNS. THE RESNET50 IS USED AS THE BASELINE

	DNNs		
	ResNet34	WideresNet50	VGG16
% difference in MAC operations	-10%	+179%	+278%
% difference in satisfied users	-2.3%	+2.0%	+3.6%

D. Scalability Analysis and Comparative Evaluation

In this section, a scalability analysis of the proposed energy efficient edge computing framework is provided to show its efficiency and robustness in large-scale IoT environments. A smart IoT environment is simulated with 8 AoIs, where $12, 24, \ldots, 84$ IoT nodes reside in each AoI at each simulation, i.e., the total number of IoT nodes are $96, 192, \ldots, 672$ at each simulation. Fig. 7 presents the total percentage of satisfied IoT nodes in the examined smart IoT environment. The IoT nodes' computing tasks requests are considered homogeneous for fairness purposes in the scalability analysis. The results reveal that the total percentage of satisfied IoT nodes drops by approximately 2% when an eight-fold increase of the number of IoT nodes occurs. This outcome confirms the robustness of the proposed framework in large-scale IoT environments.

Additionally, we present comparative results for other state-of-art DNNs with varying computational complexity and algorithmic design, in order to demonstrate the benefits of our approach under different workloads. Table II depicts the differences in the numbers of satisfied users if we replace the ResNet50 with other DNNs. The table also depicts the differences in the number of actual computations (MAC operations) for each DNN as a metric of computational complexity. WideresNet50 and VGG16 are more computational intensive than ResNet50 and the number of satisfied users at a large scale is increased by 2% and 3.6% respectively. This verifies our approach that by utilizing approximation, we can meet the requirements of more users having a robust IoT deployment. On the other end, ResNet34 is less computational intensive than ResNet50, thus users can meet their requirements locally easier and they offload less data. For that reason the numbers of satisfied users by the FAAS drops by 2.3%.

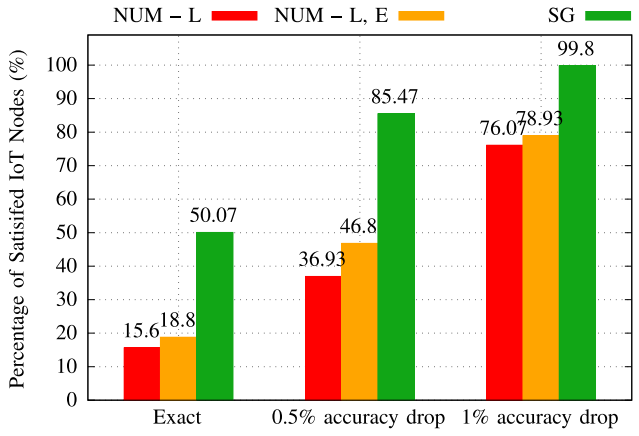

Fig. 8. Satisfaction Games versus Network Utility Maximization.

Towards showing the superiority of the theory of Satisfaction Games in terms of dealing with resource management problems, we compare our proposed computing task offloading framework to the state of the art. Specifically, the most common theory that is used nowadays to perform the resource management in communications and computing systems is the Network Utility Maximization (NUM) theory [35]. The main principle under the NUM theory is that every user in a complex system aims at maximizing its utility or equivalently, minimizing its experienced cost. We consider three comparative scenarios: 1) SG: our proposed satisfaction games-based computing task offloading model; 2) NUM – L, E: the IoT nodes decide their optimal amount of offloaded data by solving the optimization problem $\min\{w_1 L_{ua}^{(t)} + w_2 E_{ua}^{(t)}\}, w_1 + w_2 = 1$; and 3) NUM – L: the optimal amount of offloaded data is determined by solving the optimization problem $\min\{L_{ua}^{(t)}\}$. Fig. 8 presents the percentage of satisfied IoT nodes under the three comparative scenarios and considering the three different computing levels requested by the nodes. Specifically, for the purposes of the comparison, we have focused on one AoI with 45 IoT nodes, and the FAAS hovering above this area. The results reveal that the theory of satisfaction games enables the superior satisfaction of the IoT nodes' latency and energy consumption constraints by sparingly using the overall system's available resources. Also, the NUM – L scenario presents the worst results, as it simply tries to minimize the nodes' experienced latency without considering their energy constraints, and the shared nature of the computing resources with the other nodes. The NUM – L, E scenario presents an intermediate behavior in terms of the percentage of satisfied nodes, as even if it jointly tries to minimize the nodes' experienced latency and energy consumption, it still allows the nodes to act in a selfish manner in terms of the usage of the shared computing resources.

VI. Conclusion and Future Work

In this paper, an energy efficient edge computing framework is introduced enabled by the theory of satisfaction games and approximate computing. A smart Internet of Things (IoT) environment is considered consisting of multiple areas of interest (AoIs), where several IoT nodes reside per AoI, a multi-access edge computing server is attached to the macro-cell base station serving the examined IoT environment and a fully autonomous aerial system (FAAS) providing computing support on-demand hovers above the AoIs to accommodate the IoT nodes' computing demand. Based on the IoT service that the nodes serve, they are characterized by specific latency and energy consumption constraints, as well as acceptable flexibility level of maximum accuracy drop during the processing of their computing tasks. A data offloading problem is formulated to guarantee the IoT nodes' minimum latency and energy consumption requirements and it is formulated as a satisfaction game. A distributed learning approach is introduced to determine the Satisfaction Equilibrium of the system, where all the IoT nodes satisfy their latency and energy consumption constraints. The proposed distributed learning algorithm also determines the Generalized Satisfaction Equilibrium, where part of the nodes satisfy their constraints, while the rest of them are unable to do so due to the limited computing resources. A reinforcement learning-based algorithm is proposed to enable the FAAS to autonomously determine its optimal trajectory within the smart IoT environment to serve the computing demand of the IoT nodes. A detailed set of numerical and comparative results show the pure operation performance of the proposed framework, the benefits of adopting the reinforcement learning approach to autonomously perform the FAAS optimal path planning, and a scalability analysis demonstrates the robustness of the proposed framework across multiple nodes as well as different DNN-based applications. Part of our current and future work is to examine the proposed model in an integrated access and backhaul (IAB) framework, where the IoT nodes' data follow a multi-hop transmission, and computing capabilities can be offered in different levels, such as multi-access edge computing, fog computing, and cloud computing via exploiting the benefits of approximate computing. Also, part of our future work includes the study of the edge-based federated learning problem, where the edge server (or the FAAS) "recruits" the IoT nodes to perform the edge-based federated learning towards enabling it to learn a global optimal strategy, e.g., service recommendations.

References

[1] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, 4th Quart., 2018.

[2] Y. Li and L. Cai, "UAV-assisted dynamic coverage in a heterogeneous cellular system," *IEEE Netw.*, vol. 31, no. 4, pp. 56–61, Jul./Aug. 2017.

[3] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 2, pp. 1–23, 2018.

[4] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Austin, TX, USA, 2016, pp. 1–7.

[5] G. Zervakis, H. Amrouch, and J. Henkel, "Design automation of approximate circuits with runtime reconfigurable accuracy," *IEEE Access*, vol. 8, pp. 53522–53538, 2020.

[6] H. Amrouch, G. Zervakis, S. Salamin, H. Kattan, I. Anagnostopoulos, and J. Henkel, "NPU thermal management," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3842–3855, Nov. 2020.

[7] R. Wang, Y. Cao, A. Noor, T. A. Alamoudi, and R. Nour, "Agent-enabled task offloading in UAV-aided mobile edge computing," *Comput. Commun.*, vol. 149, pp. 324–331, Jan. 2020.

[8] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang, and X. Lang, "Learning-based computation offloading approaches in UAVs-assisted edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 928–944, Jan. 2021.

[9] Y.-S. Wang, Y.-W. P. Hong, and W.-T. Chen, "Trajectory learning, clustering, and user association for dynamically connectable UAV base stations," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 4, pp. 1091–1105, Dec. 2020.

[10] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 1211–1223, Jan. 2021.

[11] D. Sikeridis, E. E. Tsiropoulou, M. Devetsikiotis, and S. Papavassiliou, "Wireless powered public safety IoT: A UAV-assisted adaptive-learning approach towards energy efficiency," *J. Netw. Comput. Appl.*, vol. 123, pp. 69–79, Dec. 2018.

[12] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, May 2020.

[13] M. Hua, Y. Wang, C. Li, Y. Huang, and L. Yang, "UAV-aided mobile edge computing systems with one by one access scheme," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 664–678, Sep. 2019.

[14] J. Zhang *et al.*, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.

[15] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[16] H. Chang, Y. Chen, B. Zhang, and D. Doermann, "Multi-UAV mobile edge computing and path planning platform based on reinforcement learning," 2021, *arXiv:2102.02078*.

[17] S. Cass, "Taking AI to the edge: Google's TPU now comes in a maker-friendly package," *IEEE Spectr.*, vol. 56, no. 5, pp. 16–17, May 2019.

[18] J. Song *et al.*, "7.1 an 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2019, pp. 130–132.

[19] Z.-G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-oriented approximation for energy-efficient neural network inference accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 12, pp. 4670–4683, Dec. 2020.

[20] X. Jiao, V. Akhlaghi, Y. Jiang, and R. K. Gupta, "Energy-efficient neural networks using approximate computation reuse," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Dresden, Germany, 2018, pp. 1223–1228.

[21] G. Zervakis, O. Spantidi, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Control variate approximation for DNN accelerators," 2021, *arXiv:2102.09642*.

[22] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *Proc. IEEE Int. Solid-State Circuits Conf.*, San Francisco, CA, USA, 2018, pp. 218–220.

[23] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2017, pp. 240–241.

[24] J. G. Boubin, N. T. R. Babu, C. Stewart, J. Chumley, and S. Zhang, "Managing edge resources for fully autonomous aerial systems," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, 2019, pp. 74–87.

[25] P. A. Apostolopoulos, M. Torres, and E. E. Tsiropoulou, "Satisfaction-aware data offloading in surveillance systems," in *Proc. 14th Workshop Challenged Netw.*, 2019, pp. 21–26.

[26] S. Papavassiliou, E. E. Tsiropoulou, P. Promponas, and P. Vamvakas, "A paradigm shift toward satisfaction, realism and efficiency in wireless networks resource sharing," *IEEE Netw.*, vol. 35, no. 1, pp. 348–355, Jan./Feb. 2021.

[27] S. M. Perlaza, H. Tembine, S. Lasaulce, and M. Debbah, "Quality-of-service provisioning in decentralized networks: A satisfaction equilibrium approach," *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 2, pp. 104–116, Apr. 2012.

[28] M. Goonewardena, S. M. Perlaza, A. Yadav, and W. Ajib, "Generalized satisfaction equilibrium: A model for service-level provisioning in networks," in *Proc. Eur. Wireless 22nd Eur. Wireless Conf.*, Oulu, Finland, 2016, pp. 1–5.

[29] M. Diamanti, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Unified user association and contract-theoretic resource orchestration in NOMA heterogeneous wireless networks," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1485–1502, 2020.

[30] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.

[31] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2019, pp. 7950–7958.

[32] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "SCALE-sim: Systolic CNN accelerator simulator," 2018, *arXiv:1811.02883*.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, 2009, pp. 248–255.

[34] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv: 1804.02767*.

[35] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

Nafis Irtija received the bachelor's and master's degrees in electrical and electrical engineering from the University of Dhaka, Dhaka. He is currently pursuing the Ph.D. degree and is an Research Assistant with the Department of Electrical and Computer Engineering, University of New Mexico. His main research interests include quantum sensing, distributed decision making, reinforcement learning, game theory, optimization, contract theory, and prospect theory.

Iraklis Anagnostopoulos (Member, IEEE) received the Ph.D. degree from the Microprocessors and Digital Systems Laboratory, National Technical University of Athens. He is an Assistant Professor with the Electrical and Computer Engineering Department, Southern Illinois University Carbondale. He is the Director of the Embedded Systems Software Lab, which works on run-time resource management of modern and heterogeneous embedded many-core architectures, and he is also affiliated with the Center for Embedded Systems. His research interests lie in the area of constrained application mapping for many-core systems, design, and exploration of heterogeneous platforms, resource contention minimization, and power-aware design of embedded systems.

Georgios Zervakis received the Diploma and Ph.D. degrees from the Department of Electrical and Computer Engineering, National Technical University of Athens, Greece, in 2012 and 2018, respectively. He is a Research Group Leader with the Chair for Embedded Systems, Karlsruhe Institute of Technology (KIT), Germany. Before joining KIT, he worked as a Primary Researcher in several EU-funded projects and as a Member of the Institute of Communication and Computer Systems, Athens, Greece. His research interests include approximate computing, low power design, design automation, and integration of hardware acceleration in cloud.

Eirini Eleni Tsiropoulou (Senior Member, IEEE) is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of New Mexico. Her main research interests lie in the area of cyber-physical social systems and wireless heterogeneous networks, with emphasis on network modeling and optimization, resource orchestration in interdependent systems, reinforcement learning, game theory, network economics, and Internet of Things. Four of her papers received the Best Paper Award at IEEE WCNC in 2012, ADHOCNETS in 2015, IEEE/IFIP WMNC 2019, and INFOCOM 2019 by the IEEE ComSoc Technical Committee on Communications Systems Integration and Modeling. She received the NSF CRII Award in 2019 and the Early Career Award by the IEEE Communications Society Internet Technical Committee in 2019. She was selected by the IEEE Communication Society—N2Women—as one of the top ten Rising Stars of 2017 in the communications and networking field.

Hussam Amrouch (Member, IEEE) received the Ph.D. degree (*summa cum laude*, with Distinction) from the Karlsruhe Institute of Technology (KIT), Germany, in 2015. He is a Junior Professor with the Semiconductor Test and Reliability Chair, Computer Science, Electrical Engineering Faculty, University of Stuttgart, as well as a Research Group Leader with the KIT. He has over 100 publications in multidisciplinary research areas across the entire computing stack. His main research interests are design for reliability and testing from device physics to systems, machine learning, security, approximate computing, and emerging technologies with a special focus on ferroelectric devices. He holds seven HiPEAC Paper Awards and three best paper nominations at top EDA conferences: DAC'16, DAC'17, and DATE'17 for his work on reliability. He currently serves as an Associate Editor of *Integration, the VLSI Journal*.

Jörg Henkel (Fellow, IEEE) received the diploma and Ph.D. degrees *(summa cum laude)* from the Technical University of Braunschweig. He is the Chair Professor of Embedded Systems with Karlsruhe Institute of Technology. He was a Research Staff Member with NEC Laboratories, Princeton, NJ, USA. His research work is focused on co-design for embedded hardware/software systems with respect to power, thermal, and reliability aspects. He has received six best paper awards throughout his career from, among others, ICCAD, ESWeek, and DATE. He has led several conferences as the General Chair, including ICCAD, ESWeek, and serves as a steering committee chair/member for leading conferences and journals for embedded and cyber-physical systems. For two consecutive terms he served as the Editor-in-Chief for the *ACM Transactions on Embedded Computing Systems*. He is currently the Editor-in-Chief of the *IEEE Design & Test* and is/has been an associate editor for major ACM and IEEE Journals. He coordinates the DFG Program SPP 1500 "Dependable Embedded Systems" and is a Site Coordinator of the DFG TR89 collaborative research center on "Invasive Computing." He is the Chairman of the IEEE Computer Society and Germany Chapter.