Multi-Objective Reverse Offloading in Edge Computing for AI Tasks

Amanatidis, Petros; Michailidis, George; Karampatzakis, Dimitris; Kalenteridis, Vasileios; Iosifidis, George; Lagkas, Thomas

DOI
10.1109/OJCOMS.2025.3555947

Publication date
2025

Document Version
Final published version

Published in
IEEE Open Journal of the Communications Society

Received 27 January 2025; revised 4 March 2025; accepted 25 March 2025. Date of publication 28 March 2025; date of current version 10 April 2025.

Digital Object Identifier 10.1109/OJCOMS.2025.3555947

Multi-Objective Reverse Offloading in Edge Computing for AI Tasks

PETROS AMANATIDIS [1], GEORGE MICHAILIDIS [1], DIMITRIS KARAMPATZAKIS [1], VASILEIOS KALENTERIDIS [1], GEORGE IOSIFIDIS [2], AND THOMAS LAGKAS [1] (Senior Member, IEEE)

[1]Department of Informatics, Democritus University of Thrace, 65404 Kavala, Greece
[2]Department of Software Technology, Delft University of Technology, 2600 AA Delft, The Netherlands

CORRESPONDING AUTHOR: T. LAGKAS (e-mail: tlagkas@cs.duth.gr)

This work was supported by the European Union's Horizon Europe Research and Innovation Programme under Grant 101070181.

ABSTRACT Offloading tasks between edge nodes is a subject that has drawn a lot of attention since edge computing first emerged. A large number of edge IoT devices utilizing increased computing resources such as autonomous vehicles and UAVs can be used to execute AI tasks close to users. We present a novel approach that deviates from the conventional edge computing offloading concept namely offloading computationally intensive tasks from cloudlets to nearby end nodes. Specifically, we enhance a scenario where end nodes assist more powerful nodes (like cloudlets) in executing AI inference tasks. In edge computing networks, as end nodes grow in number, they build an idle computing capacity which can solve and provide efficient solutions. Our goal is to solve a defined Multi-Objective optimization problem with three objectives namely the overall execution time (slowest substasks), the execution accuracy, and the total energy consumption. We address this challenging optimization problem using a novel method with our released Multi-Objective Edge AI-Adaptive Reverse Offloading, or MOEAI-ARO, algorithm. Using an edge computing testbed and a representative AI service, we demonstrate the effectiveness of our reverse offloading proposal and method. The results indicate that our method further optimizes the system's performance compared to baseline algorithms.

INDEX TERMS AI task offloading, edge computing, multi-objective optimization, resource allocation.

I. INTRODUCTION

EDGE computing is becoming a necessary component of our lives and is constantly developing. Enabling end nodes (such as mobile phones, Internet of Things devices, etc.) to execute computational tasks closer to the data source is considered a crucial component of future networks. These tasks are often offloaded to more powerful edge devices like cloudlets which enhances the low-latency applications in IoT [1], [2], [3]. Edge computing improves the performance of resource-intensive tasks by bringing computation and storage closer to the user. This reduces the latency caused by sending data to distant cloud servers. On the other hand, communication costs in terms of latency and energy consumption must be taken into account for end devices that are powered by batteries.

Interestingly, there is a growing opportunity to investigate the opposite direction: allowing end nodes to perform computational tasks that are assigned to them by nearby edge servers or cloudlets. Through the utilization of numerous end nodes located close to the edge servers, this new architecture has the potential to provide even greater advantages by decreasing response times and improving real-time applications. This potential architecture can be implemented particularly in edge computing applications such as smart cities or in industry, where many end devices either remain idle or execute tasks that require small processing power, thereby enabling them to assist the edge servers in the data processing created within the edge computing network.

In the last few years, the research community has thoroughly researched the "traditional" task offloading, from the end devices to the edge servers. Indicating the powerful hardware of the end devices, this new reverse task offloading architecture can prove even more beneficial for processing AI tasks. It is true that as end devices increase in edge

computing networks, they create an idle computing capacity. This capacity can be useful if it is combined and utilized methodically.

The use of Machine Learning (ML) inference from data samples collected at the cloudlets has increased recently. Pre-trained Deep Neural Networks (DNNs) are deployed on the end nodes in large numbers thanks to improvements in end nodes' hardware and the creation of DNN models that require less processing power and storage at the expense of lower inference accuracy [4], [5], [6]. The deployment of AI models on end nodes is enabled by dedicated AI hardware integrated into current processors. Several AI models of different computational characteristics are being used at the edge. The inference time, accuracy, and energy consumption are directly related to the model's characteristics. In more detail, larger models achieve more accurate results or predictions but need longer processing time and more energy.

Demonstrating their crucial role in reducing task execution delays and energy consumption of end nodes, task offloading has gained significant attention in recent years [7], [8], [9]. Furthermore, the implementation of different kinds of optimization algorithms for task allocation guarantees in most cases the optimal use of the resources in IoT networks which significantly reduces latency and energy usage. The research community is continuously focusing on improving such task allocation methods to handle real-time applications. This enhances the opportunity to investigate the implementation of novel algorithms suited for our proposed task offloading architecture.

In this work, we use the reverse offloading approach and solve a multi-objective optimization problem with a newly introduced optimization algorithm. We consider, as a use case, an exemplary AI inference service, namely, an object detection process on images that are located in an edge server. Our goal in our defined multi-objective optimization problem is to find the optimal distribution of the images among the available edge devices (edge server, end devices) and also find the optimal deep learning model selection which takes into account all objectives simultaneously. In our optimization problem, we consider three objectives: the AI inference latency which is determined by the slowest execution of any subtask assigned to some end node, the accuracy of the object detection process, and the total energy consumed for processing the images.

Since the above objectives are competitive, there is not a single solution that optimizes the three objectives simultaneously. For this reason, we use the notion of Pareto-optimal solutions [10], [11]. We propose a novel methodology for the automatic selection of a solution that belongs to the Pareto front.

To tackle the challenging mixed continuous-discrete optimization problem we introduce a novel adaptive task offloading method that combines a Linear Programming (LP) [12], [13] algorithm with a stochastic Non-dominated Sorting Genetic Algorithm (NSGA) [14]. The results show considerable performance gains and cost savings.

A. METHODOLOGY AND CONTRIBUTIONS

We explore a scenario of a wireless edge computing network with one access point, an edge server, and end nodes with multiple implemented deep learning models. All end devices are connected via links of different capacities with the edge server.

We search for a solution that optimizes simultaneously three objectives, namely the processing speed of the workload, the execution accuracy, and the total energy consumption by splitting the workload (images) among available edge devices via the edge server and additionally selecting the optimal deep learning model for each edge node. The workload consists in processing images (such as videos with a specified duration) that are distributed from the edge server to the edge nodes (including the edge server) where an object detector is used to locate objects of interest. Since each edge device disposes a set of object detectors with a specific neural network size, each combination of device and object detector model essentially has different performance and efficiency. Furthermore, since the devices are connected over wireless links with varying capacities, the transmission time from the edge server to each device differs. The distribution or splitting of the images and the selection of the object detector for the edge nodes are determined by the edge server's algorithm. For the evaluation of the performance, gains are expressed in terms of all objectives.

We provide a complete solution employed as a reverse offloading method. Using a wireless testbed consisting of three Raspberry Pis (RPis) of which one serves as an edge server and two Nvidia Jetsons, this method is evaluated through a series of experiments utilizing the YOLO object detector [15]. When compared to different task offloading baseline methods, the findings demonstrate that our implemented strategy provides an improved overall performance. Furthermore, our offloading technique which we refer to as Multi-Objective Edge AI-Adaptive Reverse Offloading (MOEAI-ARO), tailors its operation by considering resource availability.

Consequently, the following contributions are provided by this work:

- We enhance the concept that involves edge nodes splitting up their tasks and sending the subtasks to far-edge nodes (end devices). This is known as reverse offloading, which is of high importance for edge computing services and IoT networks.
- Formulation of a Multi-Objective optimization problem that optimizes simultaneously three crucial objectives, namely the latency, the total energy consumption, and the execution accuracy for edge computing applications.
- Consideration of the different object detector models as an optimization variable.
- Development of an adaptive reverse offloading algorithm (MOEAI-ARO) that can be tailored to different uniform AI tasks as well as different end devices with

different hardware processing capacities and resource constraints.
- Thorough performance evaluation by comparing the proposed MOEAI-ARO algorithm with various baseline algorithms in a wireless edge computing testbed, which consists of end devices with different processing capacities.

II. RELATED WORKS

Reverse task offloading. The optimization of the system's latency for vehicular edge computing using a reverse offloading framework was presented in [16]. Resource management and task allocation were optimized via a greedy-based efficient searching (GES) method for binary reverse offloading strategies and the joint alternative optimization-based bi-section searching strategy for partial reverse offloading. Another reverse offloading framework that used a GES algorithm was presented in [17] optimizing the system's services capacity in a cooperative fashion. Moreover, in [18] the authors developed a reverse offloading strategy for MEC (Multi-access Edge Computing). In more detail, they implemented a heuristic-based and machine learning method, such as Deep Reinforcement Learning, to optimize reverse offloading decisions, effectively reducing latency and energy consumption. In this paper, we focus particularly on handling resource-intensive tasks, such as computer vision tasks, using additional degrees of freedom (DoF) corresponding to different object detector models.

AI task offloading. In recent years much work from the scientific community has been devoted to the optimization of AI inference tasks in edge computing networks. Researchers in [19] formulated a Mixed-Integer Nonlinear Programming (MINLP) problem to optimize the latency, accuracy, and energy consumption of AI inference in edge computing networks for videos using the so-called Channel-Aware heuristic algorithm. Offloading of Machine Learning tasks in edge computing networks was examined in [20]. An Integer Linear Programming problem was defined, aiming to maximize the processing accuracy of AI inference tasks. In more detail, the authors' purpose was to find the optimal trade-off between the model size and the processing accuracy using a Dynamic Programming algorithm. In addition, an Automated Machine Learning framework was proposed in [21] that optimizes the inference accuracy of AI tasks while adhering to the minimum frame-rate constraint. The introduced online optimization algorithm improves the performance of the edge computing systems in real time without violating the constraints. All the aforementioned methods only employ continuous or integer optimization variables. Our method shows how to treat discrete and continuous optimization variables efficiently in a model where an edge device (cloudlet) divides and offloads the tasks to multiple smaller devices.

Multi-objective optimization. The most reasonable optimization problems in edge computing are those that optimize multiple objectives. In [22] heuristic-based solutions are used to demonstrate significant gains in computational time and energy efficiency. The proposed system, named RAMOS, utilizes different kinds of edge nodes and addresses a multi-objective resource-aware task assignment and scheduling problem with modes for energy efficiency and latency minimization. An interesting work is presented in [23] where the authors tried to identify the optimal trade-off accuracy and latency for deep learning tasks. They formulated an Integer Linear Programming optimization problem to optimize the user's satisfaction. Additionally, a polynomial constant-time greedy algorithm was presented to get the neat optimal solution. Another work [24] introduced a Constrained Multi-Objective Decomposition Evolutionary Algorithm to optimize energy consumption and latency. This algorithm was implemented in a UAV-assisted MEC network. Another paper [25] attempted to challenge the optimal management of available resources (edge devices) in IoV networks. The NSGA-III algorithm was implemented to optimize objectives such as latency, energy consumption, and load balancing. Authors in [26] used a Genetic Algorithm (GA) [27] to optimize the task processing latency while adhering to several constraints. The presented approach attempts to find the optimal resource allocation in the MEC network. In our proposal, we explore further the implementation of an evolutionary algorithm (NSGA) for our multi-objective optimization problem with the difference that our constraints are handled by an LP algorithm which results in a highly efficient optimization algorithm. This novel strategy efficiently optimizes our three objectives using both discrete and continuous optimization variables. The discrete variables are handled by the NSGA algorithm while the continuous ones by the LP algorithm.

III. MODEL AND PROBLEM FORMULATION

We present our system model and the corresponding mathematical formulation. A batch of images B, or else the Batch step, is distributed by the edge server to a set $\mathcal{N}$ of N devices including the edge server for processing. We consider a wireless IoT network represented by the set $\mathcal{N}$ connected over wireless links with different capacities, all operating with standard IoT technologies such as Wi-Fi. The devices execute an AI task, or more specifically, an object detection task. For object detection, each device in our system implements a deep learning model (neural network). Because the models may vary in size, they may also differ in terms of energy consumption, mean accuracy, and computational performance. A set of M neural networks is represented by the index $y_i \in \mathcal{M}$ for each device $i \in \mathcal{N}$ where $\mathcal{M}=\{1, 2, \ldots, M\}$. We set the index $i = 1$ for the edge server.

We introduce an optimization variable $x_i \in [0, 1]$ for each edge device i. This variable denotes the part of B distributed to the edge device $i \in [1, N]$. All of these variables are collected into a vector $\mathbf{x} = [x_1, \ldots, x_N] \in [0, 1]^N$ and satisfy the constraint $\sum_{i=1}^{N} x_i = 1$. We also add a discrete optimization variable $\mathbf{y} = [y_1, \ldots, y_N]$ which denotes the

TABLE 1. Key parameters, functions and variables.

Description	Parameters/variables
x_i	Percentage of the total number of images sent to device i.
y_i	Object detector choice for every device i.
B	Batch of images for processing (Batch step).
T_{tx}^i	Time to send a single image to device i.
$T_{dl}^i(y_i)$	Time per image for the device i to execute the object detector using the neural network y_i.
$L_i(x_i, y_i)$	Time for the server communication and object detection execution of device i.
$mAP(\mathbf{x}, \mathbf{y})$	Mean accuracy of the overall object detection process.
$mAP(y_i)$	Mean accuracy per image of the object detection process using the neural network y_i.
$E(\mathbf{x}, \mathbf{y})$	Total energy consumed for the data transmission, data reception and object detection process.
$E_i(x_i, y_i)$	Energy consumed by device i for data reception and task execution.
$E_i^{exec}(y_i)$	Energy consumed by device i for the object detection process of one image using neural network y_i.
E_i^{rec}	Energy consumed by device i for the reception of one image.
E_i^{tr}	Energy consumed by the edge server to transmit one image to device i.
$E_i^{available}$	Energy available for device i.
x_0	Auxiliary variable that serves simultaneously as cost function and uniform bound for the latency of each device i.
$\mathbf{w}$	Vector of weight coefficients.
P	Number of Pareto optimal solutions.
f^{CoG}	The Center of Gravity (CoG) of the solutions on the Pareto front.
f_i	Value of objective i.
f_i^{norm}	Normalized value of objective i.
$f_{i,j}^{norm}$	The i^{th} normalized objective of the j^{th} solution on the Pareto front.
$distance(x_j, x_k)$	Ellipsoidal distance function which computes a distance between two solutions x_j,x_k.
x_{def}	The solution on the Pareto front that has the minimum distance from the CoG.

neural network selection utilized for object detection at each edge device.

Remark: The number of the images distributed from the edge server to the end device $i \in [1, N]$ equals $x_i B$ and is rounded to the nearest integer value. As long as a significant number of tasks is distributed at each batch step, this rounding has a negligeable impact on the optimization solution.

A. FORMULATION OF THE OPTIMIZATION PROBLEM

In this section, we provide the analytical formulation of the optimization problem. All key parameters, functions, and decision variables used in our formulation are summarized in Table 1.

Our first objective is to minimize the maximum latency. Here, latency is defined for each device as the total amount of time required for both, the transmission time of images from the server to the end device i, denoted as T_{tx}^i, and the time to execute the object detection process by end device i using the neural network y_i, denoted as $T_{dl}^i(y_i)$, i.e.,

$$L_i(x_i, y_i) = x_i B T_{tx}^i + x_i B T_{dl}^i(y_i) = C_i x_i \tag{1}$$

where:

$$C_i = B\Big(T_{tx}^i + T_{dl}^i(y_i)\Big) \tag{2}$$

$$T_{tx}^i = \frac{Datasize}{DataRate}. \tag{3}$$

We denote the maximum latency as:

$$L_{max}(\mathbf{x}, \mathbf{y}) = \max_i L_i(x_i, y_i) \tag{4}$$

The second objective corresponds to maximizing the mean accuracy of the total object detection process, denoted as $mAP(\mathbf{x}, \mathbf{y})$, which is computed as:

$$mAP(\mathbf{x}, \mathbf{y}) = \sum_i x_i mAP(y_i), \tag{5}$$

where $mAP(y_i)$ stands for the mean accuracy of the object detection process for one image using the neural network y_i. Finally, the third objective is to minimize the total energy consumption ($E(\mathbf{x}, \mathbf{y})$), which corresponds to the energy cost for data transmission (E_i^{tr}), data reception (E_i^{rec}) and task execution (E_i^{exec}), i.e.:

$$E(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} x_i B E_i^{exec}(y_i) + \sum_{i=2}^{N} x_i B\Big(E_i^{rec} + E_i^{tr}\Big) \tag{6}$$

Remark: In equation (6), we have considered that $E_1^{rec} = 0$, since the edge server does not consume any energy for data reception, and $E_1^{tr} = 0$, since it does not transmit data to itself.

Thus, the objective function of the multi-objective optimization problem can be written in vector form as:

$$\min_{\mathbf{x},\mathbf{y}} [L_{max}(\mathbf{x}, \mathbf{y}), -mAP(\mathbf{x}, \mathbf{y}), E(\mathbf{x}, \mathbf{y})] \tag{7}$$

In real-world applications, it is also necessary to consider the energy availability to ensure that devices are truly capable of completing the assigned tasks. For every device i, we need to include an additional constraint, s.t.

$$E_i(x_i, y_i) \leq E_i^{available}, \tag{8}$$

where $E_i(x_i, y_i)$ corresponds to the energy consumed by the device i for data transmission, data reception, and task execution, and $E_i^{available}$ represents the energy available in

the end device i. Depending on the device index i, the term $E_i(x_i, y_i)$ is further analysed as:

$$E_1(x_1, y_1) = \sum_{i=2}^{N} x_i BE_i^{tr} + x_1 BE_1^{exec}(y_1) \quad (9)$$

$$E_i(x_i, y_i) = x_i BE_i^{rec} + x_i BE_i^{exec}(y_i), \quad i = 2, \ldots, N \quad (10)$$

Therefore, the complete optimization problem reads:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \ & [L_{max}(\mathbf{x}, \mathbf{y}), -mAP(\mathbf{x}, \mathbf{y}), E(\mathbf{x}, \mathbf{y})] \quad (11.1) \\ \text{s.t.} \ & E_i(x_i, y_i) \leq E_i^{available}, \quad i = 1, \ldots, N \quad (11.2) \\ & \sum_{i=1}^{N} x_i = 1 \quad (11.3) \\ & x_i \in [0, 1], \quad i = 1, \ldots, N \quad (11.4) \end{aligned} \quad (11)$$

To provide a detailed explanation of the multi-objective optimization problem formulated in (11), the objective in (11.1) is to simultaneously optimize the three objectives, namely to minimize the maximum latency, maximize the mean accuracy, and minimize the total energy consumption. Regarding the constraints of the optimization problem, equation (11.2) ensures that end device i has the required available energy ($E_i^{available}$) to execute the distributed task. According to equation (11.3), the sum of the portion $\{x_i\}_{i=1,\ldots,N}$ of B transmitted to all end devices must be equal to 1. Lastly, the bound constraints for the optimization variables $\{x_i\}_{i=1,\ldots,N}$ are stated in equation (11.4).

As in our previous work [28], to avoid the non-differentiability of the *max* operator in $L_{max}(\mathbf{x}, \mathbf{y})$, we adopt the approach in [29] and introduce an auxiliary optimization variable, denoted $x_0 \in [0, \infty)$ which serves both as a cost function and as a uniform bound for the latency of each device, i.e., we use the equivalence between the following optimization problems:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \ & \max_i L_i(x_i, y_i) \ \text{ and } \ \min_{\mathbf{x},\mathbf{y}} x_0 \\ \text{s.t.} \ & L_i(x_i, y_i) \leq x_0, i = 1, \ldots, N \end{aligned}$$

Finally, the multi-objective optimization problem reads:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \ & [x_0, -mAP(\mathbf{x}, \mathbf{y}), E(\mathbf{x}, \mathbf{y})] \\ \text{s.t.} \ & C_i x_i \leq x_0, \quad i = 1, \ldots, N \\ & E_i(x_i, y_i) \leq E_i^{available}, \quad i = 1, \ldots, N \\ & \sum_{i=1}^{N} x_i = 1 \\ & x_0 \geq 0 \\ & x_i \in [0, 1], \quad i = 1, \ldots, N \end{aligned} \quad (12)$$

The main challenges in efficiently solving the optimization problem (12) and providing a solution that offers a well balanced trade-off in terms of all three objectives are explained in Section III-B and III-C correspondingly.

B. SOLUTION OF MULTI-OBJECTIVE OPTIMIZATION PROBLEM

The above optimization problem can be written in compact form as:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{x_0} \in \boldsymbol{F}} \{f_1, f_2, f_3\} \quad (13)$$

where F denotes the feasible set defined by the constraints of problem (12), while f_1, f_2, f_3 denote the corresponding objectives. Using a classical Genetic Algorithm for this problem would be too costly for a practical implementation of a task-offloading problem, where decisions need to be taken in very short time. More specifically, GA algorithms are not very efficient in handling several constraints simultaneously, in particular for problems as (12) where continuous variables appear in the constraints formulation. This difficulty becomes even more evident as the number of devices increases, requiring many iterations for the algorithm to converge.

For this reason, as in [28], we search to tailor a Genetic Algorithm with an LP algorithm. The former aims to find the optimal choice of neural networks while the latter is occupied with choosing the optimal task distributions and satisfying all the set of constraints for each choice of neural networks.

To formulate such an LP problem, we need to construct a single-objective optimization problem. Since all our constraints are linear (for fixed y), the objective function shall also be chosen to be linear. An evident choice is to construct a weighted sum of f_1, f_2, f_3, i.e.,

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{x_0} \in \boldsymbol{F}} w_1 f_1 + w_2 f_2 + (1 - w_1 - w_2) f_3 \quad (14)$$

where $w_1, w_2, w_3 \geq 0$ are positive weight coefficients.

However, the choice of the above weights is not straightforward. It seems more natural to produce a front of Pareto-optimal solutions for different values of weights, and then choose accordingly. However, it would be too costly to solve the above optimization problem for a great number of weight combinations. Another idea, adopted in this work, consists in adding $\mathbf{w} = [w_1, w_2, w_3]$ as an optimization variable in the Genetic Algorithm, searching at the same time for combinations that create Pareto-optimal solutions. Therefore, the NSGA-II algorithm creates Pareto-optimal solutions for the optimization problem:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{w},x_0 \in \boldsymbol{F}} w_1 f_1 + w_2 f_2 + (1 - w_1 - w_2) f_3 \quad (15)$$

by evolving a population of solutions $\{\mathbf{y}, \mathbf{w}\}$ while for each member of the population the optimization variables $\mathbf{x}, x_0$ are determined via the solution of the LP-problem:

$$\min_{\boldsymbol{x},x_0 \in \boldsymbol{F}} w_1 f_1 + w_2 f_2 + (1 - w_1 - w_2) f_3 \quad (16)$$

To achieve a better scaling for the three objectives in the weighted sum (16), all values are normalized such that $f_i^{norm} \in [0, 1]$, $i = 1, 2, 3$, using the formula:

$$f_i^{norm} = \frac{f_i - f_i^{min}}{f_i^{max} - f_i^{min}} \quad (17)$$

where f_i^{max}, f_i^{min} are the maximum and minimum possible values for each objective.

Remark: Let us emphasize that several Evolutionary Algorithms could be used instead of the proposed Genetic Algorithm, without expecting any significant influence on the performance of the method. The central idea of the methodology lies in the combination of any Evolutionary Algorithm with an LP algorithm. Our choice of GA lies purely in its simplicity of implementation.

C. DEFAULT SELECTION AMONG PARETO-OPTIMAL SOLUTIONS

By definition, all Pareto-optimal solutions are equivalent. Without additional specifications, we propose a methodology for default selection among Pareto-optimal solutions, based on a distance function, which ensures a proper equilibrium with respect to all objectives.

First, we compute the Center of Gravity (CoG) of the solutions on the Pareto front. Each coordinate of the **CoG** vector reads:

$$f_i^{CoG} = \frac{\sum_{j=1}^{P=100} f_{i,j}^{norm}}{P}, i = 1, 2, 3 \tag{18}$$

where P is the number of Pareto optimal solutions on the front and $f_{i,j}^{norm}$ denotes the i^{th} normalized objective of the j^{th} solution on the front.

Then, we define an ellipsoidal distance function which computes a distance between two points $\mathbf{x}_j$,$\mathbf{x}_k$ on the Pareto front as:

$$distance(x_j, x_k) = \sqrt{\sum_{i=1}^{3}\left(\frac{f_{i,j}^{norm} - f_{i,k}^{norm}}{\max_l f_{i,l}^{norm} - \min_l f_{i,l}^{norm}}\right)^2} \tag{19}$$

In the above definition, all three objectives are re-normalized based on their maximum and minimum values among solutions of the front ($\max f_{i,l}^{norm}$, $\min f_{i,l}^{norm}$) which provides a better scaling among objectives of different scales and thus an improved equilibrium.

Finally, the default selection is found as:

$$\mathbf{x}_{def} = \arg\min_{\mathbf{x}_j} distance(\mathbf{CoG}, \mathbf{x}_j) \tag{20}$$

i.e., we choose the solution on the Pareto front that lies closer to the **CoG** in terms of the distance function (19).

IV. PROPOSED ALGORITHM

First, the initial population P_0 is created. It consists of Z **y** vectors that combine different neural network indices along with different weight vectors **w**. Every individual in P_0 has to solve a Linear Programming (LP) problem to evaluate its fitness function. Subsequently, the generation counter t is set to zero. In every iteration, operators for crossover and mutation are applied to P_t in order to produce an offspring population Q_t, for which we evaluate its fitness function. $R_t = P_t \cup Q_t$ is the result of combining the populations of the parents and offspring. Non-dominated sorting is performed on R_t, and crowding distances are computed to measure the density of solutions surrounding a specific individual. Based on the rank and crowding distance, the best Z individuals from R_t are chosen to form the new population P_{t+1}. Once the termination condition is satisfied (e.g., maximum number of iterations), the algorithm stops and the default Pareto-optimal solution is computed.

Algorithm 1 NSGA-II-LP (MOEAI-ARO) Algorithm

1: Initialize population P_0 of size Z.
2: Solve the LP problem and evaluate the fitness for each individual in P_0.
3: $t \leftarrow 0$.
4: **while** termination condition not met **do**
5: Apply crossover and mutation operators on P_t to generate offspring population Q_t.
6: Solve the LP problem and evaluate the fitness for each individual in Q_t.
7: Combine parent and offspring populations: $R_t = P_t \cup Q_t$.
8: Perform non-dominated sorting on R_t.
9: Calculate crowding distance for each individual in each front.
10: Form new population P_{t+1} by selecting the best Z individuals from R_t based on rank and crowding distance.
11: $t \leftarrow t + 1$
12: **end while**
13: Select the default Pareto optimal solution.

TABLE 2. NGSA-II algorithm parameters.

Optimization parameters	Values
population size	100
number of parents (μ)	100
number of children (λ)	100
number of mutants	5
max number of iterations	100

Our selection of parameters for Algorithm 1 can be found in Table 2 following an extensive amount of numerical testing.

A. COMPLEXITY ANALYSIS

The computational complexity of the proposed MOEAI-ARO method can be computed as a combination of the corresponding complexities of the NSGA-II [30] and SIMPLEX algorithms [31]. First, the complexity of the classical NSGA-II algorithm for a population of size Z and K objectives is known to be $O(ZK^2)$. Then, although the SIMPLEX method has exponential worst-case complexity, its average complexity is polynomial and approximately $O(N^2)$ where N is the number of variables. This holds because the number of variables may grow significantly, but the number of inequalities remains very limited. Therefore, the total average complexity of the MOEAI-ARO algorithm is estimated as $O(ZN^2K^2)$.

B. SCALABILITY

In our previous research [28] we proposed a clustering methodology for handling large-scale testbeds using the EAI-ARO method. The same strategy could be applied to its multi-objective version, i.e., the proposed MOEAI-ARO method, solving one multi-objective optimization problem per cluster of devices.

Since the complexity of the MOEAI-ARO method is K^2 higher compared to EAI-ARO, one should expect that the size of clusters are smaller in size. However, the exact optimal size depends highly on the corresponding implementation of the MOEAI-ARO method. Using programming languages that are suitable for high performance computing, then exploiting the possibility of massive parallelism, allows to reduce significantly the solution time of the optimization problem and thus the impact on the total latency.

Furthermore, in our current implementation the actual dynamic problem is approximated by a series of static problems and the distribution of images occurs after the optimization problem is resolved. This is an unfavourable scenario for the scalability of the method, since the optimization problem could be solved continuously, updating the dynamic parameters at every iteration. Once the tasks have been executed, the current optimal solution could be used for the next distribution. This strategy could improve significantly the scalability of the method.

All of the above directions have not been examined in this work and are addressed as future work.

V. RESULTS

In this section, we demonstrate numerical and experimental findings utilizing the suggested adaptive offloading method. We introduce our testbed and compare the MOEAI-ARO with different baseline methods.

For all baseline methods, the same methodology is used to choose a default solution on the Pareto front, illustrated in Section III-C. More specifically, for each baseline method, we compute the first Pareto front of the solutions across all neural network (NN) model combinations (see Figure 5). Then, we select the solution closest to the center of gravity of this front. The evaluation process takes into account all three objectives.

The baseline methods used to verify the effectiveness of our proposed method are the following:

- *Local:* Every task is executed locally at the edge server. This can be considered as a greedy method [32] that minimizes the network delay [33].
- *First Available (FA):* The edge server initially assigns an equal amount of tasks to the end devices and to itself. Once a device completes its assigned tasks and becomes idle, the server reallocates new tasks to it, as described in [22]. More specifically, the total batch of images is divided into equal smaller batches. When a device becomes idle the server immediately sends the next batch of images for processing. It is important to highlight that for this method, the TCP socket connection is maintained open while the devices are processing images. This choice results in lower latency, since we need no additional time to create the connection, but higher power consumption (4.5 Watt for the testbed considered herein).
- *Random:* The edge server simultaneously assigns tasks to all end devices according to a random distribution.
- *Round Robin:* The edge server distributes the tasks uniformly across all edge devices. In more detail, the distribution for this baseline is $\mathbf{x} = [0.2, 0.2, 0.2, 0.2, 0.2]$ which corresponds to 20% of the batch distributed to the five devices. This method is a commonly used baseline method [22], that ensures that the workload is equally distributed, regardless of the system's network condition.

All the above baseline methods are parameter-free, which permits one to avoid any unintentional bias during the comparison that may occur from parameters' calibration.

A. TESTBED DESCRIPTION

Figure 1 shows our heterogeneous edge computing setup, containing four end devices, an edge server, and a Wi-Fi access point. The four end devices in the illustration are two Raspberry Pi's 4 and two Nvidia Jetson's, each having 4GB of RAM and varying capacities of wireless connectivity to the edge server (see Figure 2). The edge server is also a Raspberry Pi's 4 with 8 GB RAM. All edge devices including the edge server are equipped with various YOLO object detectors. According to Table 3, each end device may have many object detection models, each implemented with a different input image size, energy consumption value, and mAP value. Regarding the energy consumption, the Nvidia jetson ($E_{Nvidia}(y_i)$) values are much lower than those of the Raspberry pi's ($E_{Rasp}(y_i)$) because of the fast processing time of each image (T_{dl}).

Batches of images selected from the COCO dataset [34] for object detection are distributed to the available edge devices via the edge server which is a typical edge device (Raspberry Pi's 4). The edge server sends the images via sockets. The latency of the device that processes the last image is considered as the latency objective. The edge server sends the images via TCP sockets, each managed by a dedicated thread. The results which are labels for each processed image and its related bounding boxes, are subsequently sent back by the end devices to the edge server.

Regarding the energy characteristics of our testbed, all devices are connected to a battery with a capacity of 20000mAh (20A for 1 hour). To provide a clear understanding of our system's energy consumption for example we assume that the edge server sends 100 images to the Raspberry Pi that consumes 200mA for one minute, or 1/60th of an hour, and the consumption of the device comes to 3.3mAh. This indicates that the battery capacity dropped from 20000mAh to 19996.7mAh, i.e., 0.0165% [35]. It is noteworthy that temperature affects battery life and that supply voltage may decrease with increasing load capacity

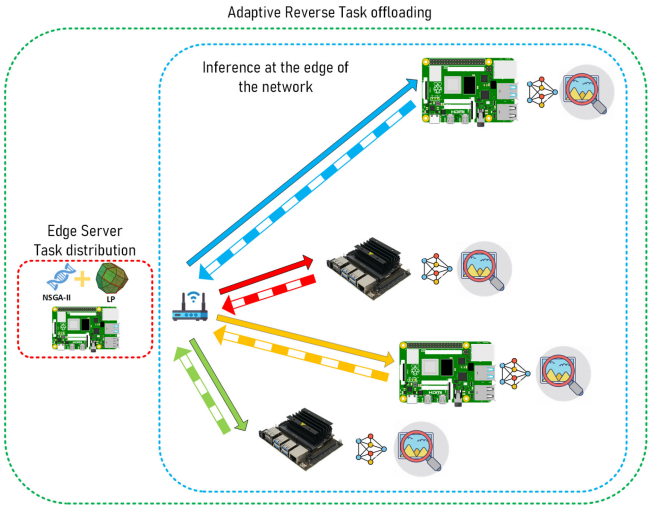

FIGURE 1. Testbed topology.

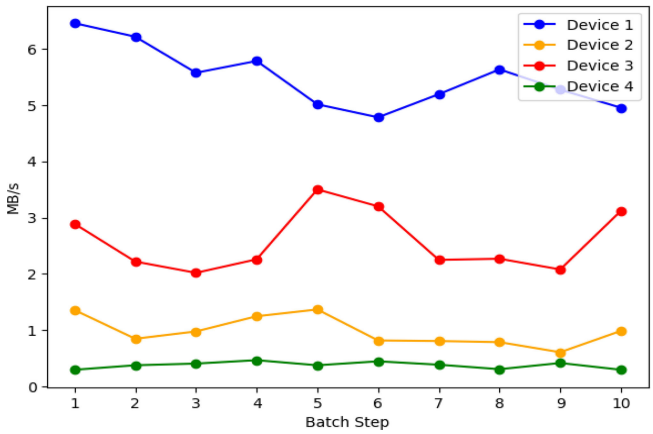

FIGURE 2. Example of network status for every batch step.

TABLE 3. Characteristics of object detection models.

y_i index	Input Image Size	$mAP(y_i)$	$E_{Rasp}(y_i)$ (Joule/image)	$E_{Nvidia}(y_i)$ (Joule/image)
1	224×224×3	0.33	0.6	0.15
2	320×320×3	0.438	1.22	0.195
3	480×480×3	0.497	2.16	0.305
4	640×640×3	0.541	4.8	0.55

over time, both of which may affect the performance of the device, but we simplify our modelling by neglecting the above factors.

In our experimental process, we use a total batch of 20000 images with an average size of 150 KB per image. After 10 batch steps of 2000 images, the images located at the edge server are processed. The current network status is considered to solve the defined multi-objective optimization problem, i.e., a set of static problems is used to approximate the actual dynamic problem. Python is used to code the entire distribution method.

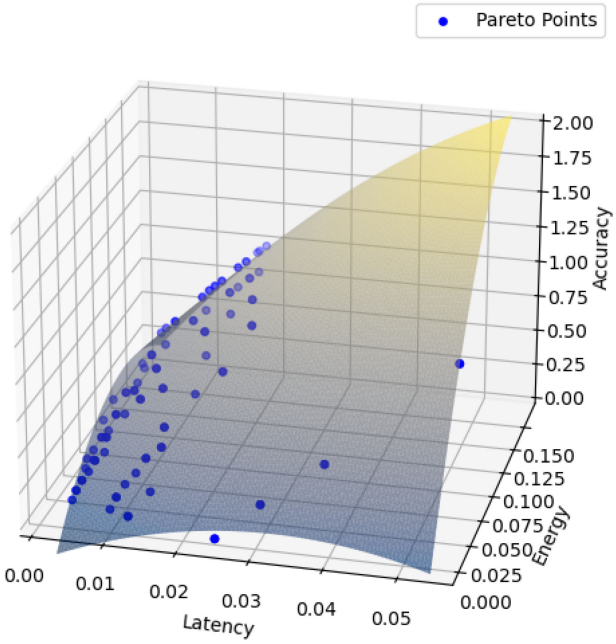

FIGURE 3. Pareto front surface.

B. TESTBED SIMULATION RESULTS

In our previous work [28] we proposed a reverse task offloading algorithm that optimizes the task distribution to minimize the latency of the slowest subtask while adhering to energy and accuracy constraints. In this work, we take our method one step further and solve a multi-objective optimization problem to provide our proposed (default) Pareto optimal solution for the three objectives.

We illustrate the Pareto front with 100 optimal solutions in Figure 3 for one batch step (2000 images) along with a fitting surface to enhance the visualization of the front. Every solution in the Pareto front corresponds to a vector $\mathbf{x} = [x_1, \ldots, x_N] \in [0, 1]^N$, a vector $\mathbf{y} = [y_1, \ldots, y_N]$, and a vector $\mathbf{w} = [w_1, w_2, w_3]$ where $\mathbf{x}$ represents the distribution percentage, $\mathbf{y}$ the selected object detection models for each device and $\mathbf{w}$ the three weight values. As expected, when the accuracy increases more resource-intensive object detection models are selected which results in longer processing time and consequently higher energy consumption.

In Figure 4, we illustrate the process of computation described in Section III-C. The Center of Gravity (CoG) of the Pareto front corresponds to the point 'X' in green color. Then, using the defined distance metric, the Nearest Point to CoG is shown in red dot and corresponds to the decision variables $\mathbf{x} = [0.271, 0.323, 0.052, 0.304, 0.050]$, $\mathbf{y} = [1, 4, 3, 3, 4]$ and $\mathbf{w} = (0.846, 0.095, 0.059)$. The same procedure for each baseline method is shown in Figure 5. The decision variables for all methods for this typical batch step are shown in Table 4.

1) COMPARISON WITH BASELINE METHODS

To provide an extensive evaluation of our proposed method, we compare it with all previously defined baseline methods.

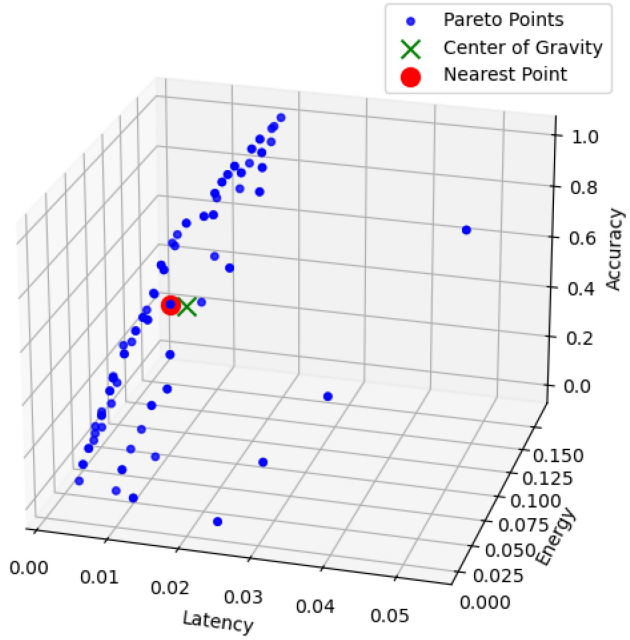

FIGURE 4. Default solution selection of MOEAI-ARO method (normalized values).

We execute all methods 100 times, under dynamic network conditions (10 batch steps), and compute the average values for all objectives. The results are summarized in Table 5 and detailed in the sequel:

- *Local:* The MOEAI-ARO method outperforms the Local execution solution in two objectives, namely 95.5% for the latency and 1.5% for the accuracy objective. This significant difference in latency is mainly due to the limited processing power of the edge server used. Regarding the energy objective the Local method outperforms the MOEAI-ARO by a difference of 47.1%, which is expected since the Local baseline method does not consume energy for data transmission and reception.
- *First Available (FA):* Our proposed method outperforms the FA method in two objectives, namely 18.5% for the latency and 39.2% for the energy objective. However, the FA method default solution exhibits better performance in accuracy by 1.2%.
- *Random:* The Random method outperforms our proposed method in terms of accuracy by 2% but is significantly worse in terms of latency 94.2% and energy consumption 45.1%. The difference in latency derives from distributing an important part of images to devices with low processing power and slow network connections. Regarding the accuracy objective, the difference of 2% is due to the fact that images were processed using neural networks with higher accuracy.
- *Round Robin:* MOEAI-ARO outperforms Round Robin in all objectives, namely 88.2% for the latency, 48.2% for the energy, and 2.6% for the accuracy objective. This method is very sensitive to the existence of some slow device and/or network connection and is mainly used when the network conditions are unknown.

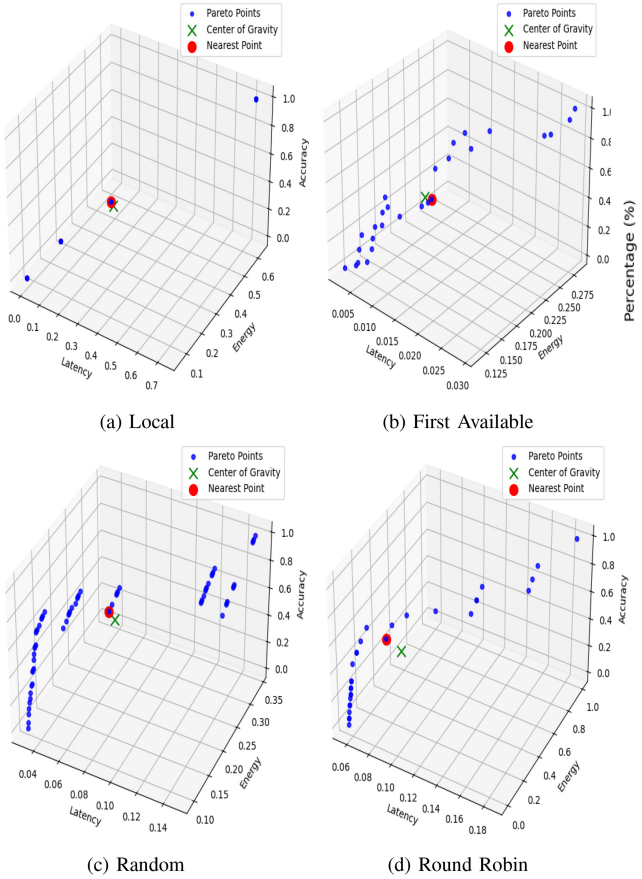

FIGURE 5. Selection of the default solution among the Pareto front for all baseline methods (normalized values).

TABLE 4. Decision variables for the default solution of each method in Figures 4 and 5.

Methods	x	y
MOEAI-ARO	[0.271, 0.323, 0.052, 0.304, 0.050]	[1, 4, 4, 3, 4]
Local	[1, 0, 0, 0, 0]	[3, 1, 1, 1, 1]
First Available	[0.10 0.40, 0.05, 0.40, 0.05]	[2, 3, 1, 4, 3]
Random	[0.20 0.54, 0.11, 0.15, 0.10]	[3, 4, 2, 4, 2]
Round Robin	[0.2, 0.2, 0.2, 0.2, 0.2]	[2, 4, 2, 4, 2]

To enhance the visualization of the results, we provide in Figure 6 a bar chart that corresponds to Table 5. The above findings show that our method considerably improves the average system performance compared to all tested baseline methods. Even when some baseline method outperforms the MOEAI-ARO solution for some objective, the average gain for all objectives is significantly higher, providing a better equilibrium for the overall performance of the system.

Remark: Note that the gain in accuracy shall not be compared in absolute values to the corresponding gains in latency or energy, since the different neural networks

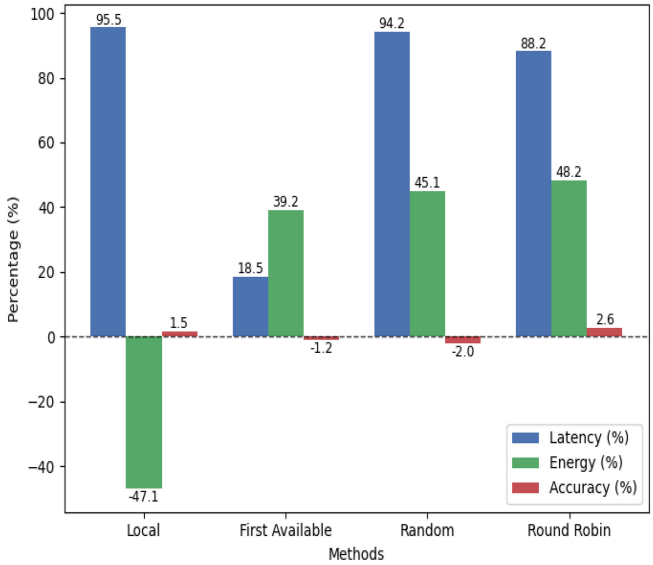

FIGURE 6. Average gain for each objective comparing the MOEAI-ARO with all baselines (simulation).

TABLE 5. Improvement for every objective using the MOEAI-ARO method compared to all baseline solutions, using numerical simulation.

Baseline method	Latency (%)	Energy(%)	Accuracy (%)
Local	95.5	-47.1	1.5
First Available	18.5	39.2	-1.2
Random	94.2	45.1	-2
Round Robin	88.2	48.2	2.6

considered present significantly lower variation in accuracy compared to the rest of the objectives.

C. TESTBED EXPERIMENTAL RESULTS

In Table 6 we illustrate the experimental results obtained after implementing all the methods on our testbed which correspond to the results illustrated in Figure 7. Same as for the simulation results, we execute all the methods 100 times under dynamic network conditions (10 batch steps) and compute the average values for all objectives. For the baseline methods, instead of testing all possible combinations of neural networks, we utilize the **y** vectors obtained from the default solutions of the baseline methods from the simulation results.

One can observe that numerical and experimental results are in good accordance, validating the numerical observations. The slight deviation is mostly due to variations in the network conditions.

Remark: Every edge device in our testbed can be used as an edge server. Depending on this choice, the results change accordingly but the general conclusions are still valid. For example, we illustrate in Figure 8 the results of our method for a virtual testbed, in which all edge devices (the edge server included) are Nvidia Jetsons. Once more, we verify that the overall performance of the system is better using our proposed methodology.

TABLE 6. Improvement for every objective using the MOEAI-ARO method compared to all baseline solutions, using experimental validation.

Baseline method	Latency (%)	Energy(%)	Accuracy (%)
Local	94.2	-45.2	1,2
First Available	10.8	34.5	-1.1
Random	84.2	41.2	-1.7
Round Robin	81.7	49.6	2.7

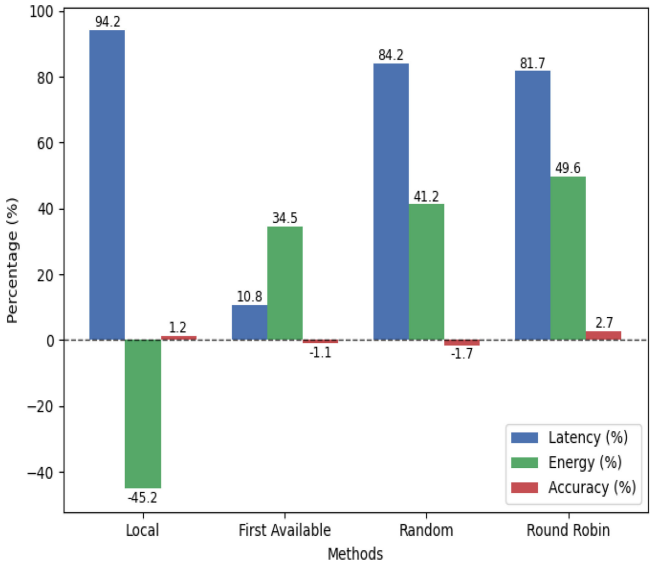

FIGURE 7. Average gain for each objective comparing the MOEAI-ARO with all baselines (experimental).

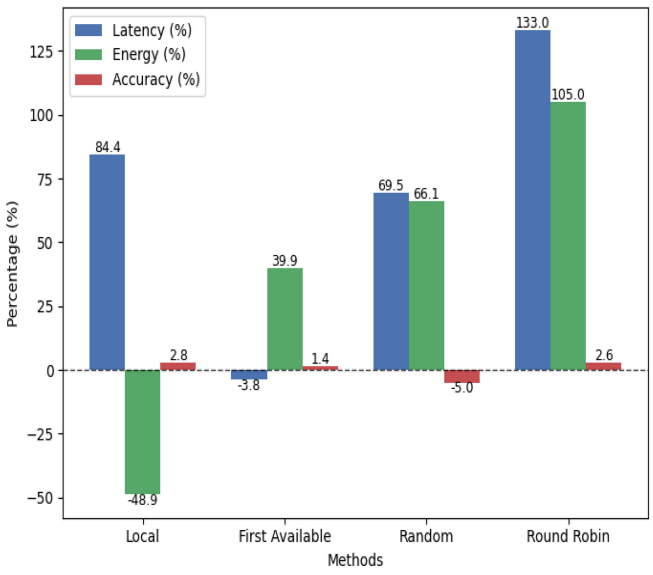

FIGURE 8. Simulation results for each objective comparing the MOEAI-ARO with all baselines, for a virtual testbed (simulation) composed of Nvidia Jetsons devices.

VI. CONCLUSION

We implemented an efficient multi-objective distribution mechanism enhancing the reverse task offloading strategy for a common AI application in IoT topologies. This work

is motivated by the increasing speed at which the computing power and storage capacity of Internet of Things (IoT) devices are growing which opens up new possibilities for IoT applications. MOEAI-ARO (Multi-Objective Edge AI-Adaptive Reverse offloading) is a distribution mechanism that solves related multi-objective optimization problems by methodically determining the best task-splitting strategy that optimizes the three objectives simultaneously. We suggest a novel method that solves the formulated optimization problem by combining two algorithms, namely the NSGA and the Linear Programming. Moreover, we propose a method for the default choice of a solution belonging to the Pareto front which provides a proper equilibrium among all objectives. We thoroughly evaluated our proposal's performance by applying it to an object detection application running on a heterogeneous testbed. The outcomes clearly show that our mechanism outperforms the considered baseline algorithms and enhances the system's overall performance. Our results provide exciting novel opportunities for future investigation. These include predictive offloading methods that could improve the effectiveness of our reverse offloading computing systems by depending on decisions via machine learning models predicting future network conditions.

ACKNOWLEDGMENT

The publication of the article in OA mode is financially supported by HEAL-Link.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, pp. 85714–85728, 2020.

[3] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, 2016, pp. 20–26.

[4] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[5] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 869–904, 2nd Quart., 2020.

[6] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep learning for edge computing applications: A state-of-the-art survey," *IEEE Access*, vol. 8, pp. 58322–58336, 2020.

[7] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000327

[8] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109137.

[9] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing,"'*IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.

[10] C. A. C. Coello, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, NY, USA: Springer, 2007.

[11] M. Ehrgott, "Vilfredo Pareto and multi-objective optimization," *Doc. Math*, vol. 8, pp. 447–453, Aug. 2012.

[12] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, vol. 6. Belmont, MA, USA: Athena Sci., 1997,

[13] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1965.

[14] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[15] G. Jocher, A. Chaurasia, and J. Qiu. "YOLO by ultralytics." 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[16] W. Feng et al., "Latency minimization of reverse offloading in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 5343–5357, May 2022.

[17] J. Feng, W. Feng, and S. Lin, "Reverse computing offloading for enhanced computing capacity in cooperative vehicle infrastructure system," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, 2021, pp. 1011–1016.

[18] M. M. Saeed et al., "Task reverse offloading with deep reinforcement learning in multi-access edge computing," in *Proc. 9th Int. Conf. Comput. Commun. Eng. (ICCCE)*, 2023, pp. 322–327.

[19] G. Pan, H. Zhang, S. Xu, S. Zhang, and X. Chen, "Joint optimization of video-based AI inference tasks in MEC-assisted augmented reality systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 479–493, Apr. 2023.

[20] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 7, pp. 2025–2039, Jul. 2023.

[21] A. Galanopoulos, J. A. Ayala-Romero, D. J. Leith, and G. Iosifidis, "AutoML for video analytics with edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[22] H. Gedawy, K. Habak, K. A. Harras, and M. Hamdi, "RAMOS: A resource-aware multi-objective system for edge computing," *IEEE Trans. Mobile Comput.*, vol. 20, no. 8, pp. 2654–2670, Aug. 2021.

[23] M. Hosseinzadeh, A. Wachal, H. Khamfroush, and D. E. Lucani, "Optimal accuracy-time trade-off for deep learning services in edge computing systems," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.

[24] L. Zhang, F. Wen, Q. Zhang, G. Gui, H. Sari, and F. Adachi, "Constrained multi-objective decomposition evolutionary algorithm for UAVs-assisted mobile edge computing networks," *IEEE Internet Things J.*, vol. 11, no. 22, pp. 36673–36687, Nov. 2024.

[25] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, "Multi-objective computation offloading for Internet of vehicles in cloud-edge computing," *Wireless Netw.*, vol. 26, pp. 1611–1629, Apr. 2020.

[26] Z. Li and Q. Zhu, "Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing," *Information*, vol. 11, no. 2, p. 83, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/83

[27] J. H. Holland, "Genetic algorithms," *Sci. Amer.*, vol. 267, no. 1, pp. 66–73, 1992.

[28] P. Amanatidis, D. Karampatzakis, G. Michailidis, T. Lagkas, and G. Iosifidis, "Adaptive reverse task offloading in edge computing for AI processes," *Comput. Netw.*, vol. 255, Dec. 2024, Art. no. 110844. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624006765

[29] J. Taylor and M. P. Bendsøe, "An interpretation for min-max structural design problems including a method for relaxing constraints," *Int. J. Solids Struct.*, vol. 20, no. 4, pp. 301–314, 1984.

[30] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud With Engineering Applications* (Intelligent Data-Centric Systems), A. K. Sangaiah, M. Sheng, and Z. Zhang, Eds., Academic, 2018, ch. 10, pp. 185–231. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128133149000104

[31] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *J. ACM*, vol. 51, no. 3, pp. 385–463, 2004.

[32] P. E. Black, "Greedy algorithm," in *Dictionary of Algorithms and Data Structures*, P. E. Black, Ed., Gaithersburg, MA, USA: NIST, Feb. 2005. Accessed: Jan. 27, 2025. [Online]. Available: https://www.nist.gov/dads/HTML/greedyalgo.html

[33] V. Cozzolino, L. Tonetto, N. Mohan, A. Y. Ding, and J. Ott, "Nimbus: Towards latency-energy efficient task offloading for ar services," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1530–1545, Apr.–Jun. 2023.
[34] "COCO—Common objects in context." Accessed: Jan. 27, 2025. [Online]. Available: https://cocodataset.org/
[35] S. Sebbio, G. Morabito, A. Catalfamo, L. Carnevale, and M. Fazio, "Federated learning on raspberry pi 4: A comprehensive power consumption analysis," in *Proc. IEEE/ACM 16th Int. Conf. Utility Cloud Comput.*, 2023, pp. 1–6.

PETROS AMANATIDIS was born in Hamburg, Germany, in 1994. He received the B.Sc. degree in computer science and the M.Phil. degree in advanced technologies in informatics and computers from the Department of Informatics, International Hellenic University, Kavala, Greece, in 2019. He is currently pursuing the Ph.D. degree with the Department of Informatics, Democritus University of Thrace. He focusing on Edge Computing, more specifically on the methodologies for optimal data and resource management. In parallel, he works as Research Assistant in several EU-funded research projects.

GEORGE MICHAILIDIS was born in Kavala, in 1984. He received the Civil Engineering degree with the Aristotle University of Thessaloniki, the M.Sc. degree in applied mathematics with the National Technical University of Athens, and the Ph.D. degree from the Ecole Polytechnique of Paris. He is currently serves as an Adjunct Professor with the Department of Informatics, Democritus University of Thrace, Greece. He has authored numerous research papers. He has been a Speaker at a number of scientific conferences worldwide. He has worked at Renault's Research Center (Renault Technocentre) and at two French universities (Ecole Polytechnique, Université Grenoble Alps). In 2017, he was hired as a Software Developer at ANSYS Inc.

DIMITRIS KARAMPATZAKIS received the Diploma degree in electronics and computer engineering from the Technical University of Crete, in 2003, and the Ph.D. degree from the University of Thessaly in 2009. He is an Assistant Professor with the Department of Informatics, Democritus University of Thrace. He has participated in national and european research projects, and publications in international scientific journals and conferences. His current research interests include edge computing, Internet of Things applications, and computer hardware.

VASILEIOS KALENTERIDIS received the B.Sc. degree in physics, the M.Sc. degree in electronic physics, and the Ph.D. degree from the Physics Department, Aristotle University of Thessaloniki in 2001, 2004 and 2012, respectively. He participated in various Greek and European research projects and has publications in book chapters, international scientific journals and conferences. He has worked in high-tech companies in the semiconductor industry as a Designer of analog integrated circuits. He is currently working at the Greek startup company Thess-IC MIKE as an R&D Designer for automotive and IoT applications. Since 2021, he has been serves as an Adjunct Professor with the Department of Informatics, Democritus University of Thrace, Greece.

GEORGE IOSIFIDIS received the Diploma degree in electronics and telecommunications engineering from the Greek Air Force Academy, Athens, in 2000, and the Ph.D. degree from the University of Thessaly in 2012. He was an Assistant Professor with Trinity College Dublin from 2016 to 2020. He is an Associate Professor with the Delft University of Technology. His research interests lie in the broad area of network optimization and economics.

THOMAS LAGKAS (Senior Member, IEEE) received the B.Sc. (Hons.) and the Ph.D. degrees in wireless networks from the Department of Computer Science, Aristotle University of Thessaloniki, in 2002 and 2006, respectively, the M.B.A. degree from Hellenic Open University, in 2012. His postgraduate certificate on teaching and learning from the University of Sheffield in 2017. He is an Assistant Professor with the Department of Informatics, Democritus University of Thrace and the Director of the Laboratory of Industrial and Educational Embedded Systems. He has been a Scholar of the Aristotle University Research Committee, as well as a Postdoctoral Scholar of the National Scholarships Institute of Greece. His research interests are in the areas of IoT communications with more than 150 publications at a number of widely recognized international scientific journals and conferences. He is a Fellow of the Higher Education Academy in U.K. Moreover, he actively participates in the preparation, management, and implementation of several EU-funded research projects.