Edge-cloud continuum solutions for urban mobility prediction and planning

LORIS BELCASTRO[1], FABRIZIO MAROZZO[1], ALESSIO ORSINO[1], DOMENICO TALIA[1], AND PAOLO TRUNFIO[1]

[1]Department of Informatics, Modeling, Electronics and Systems (DIMES), University of Calabria, Rende 87036, Italy (e-mail: {lbelcastro, fmarozzo, aorsino, talia, trunfio}@dimes.unical.it)

Corresponding author: F. Marozzo (e-mail: fmarozzo@dimes.unical.it).

ABSTRACT

In recent years, there has been an increase in the use of edge-cloud continuum solutions to efficiently collect and analyze data generated by IoT devices. In this paper, we investigate to what extent these solutions can manage tasks related to urban mobility, by combining real-time and low latency analysis offered by the edge with large computing and storage resources provided by the cloud. Our proposal is organized into three parts. The first part focuses on defining three application scenarios in which geotagged data generated by IoT objects, such as taxis, cars, and smartphones, are collected and analyzed through machine learning-based algorithms (i.e., next location prediction, location-based advertising, and points of interest recommendation). The second part is dedicated to modeling an edge-cloud continuum architecture capable of managing a large number of IoT devices and executing machine learning algorithms to analyze the data they generate. The third part analyzes the experimental results in which different design choices were evaluated, such as the number of devices and orchestration policies, to improve the performance of machine learning algorithms in terms of processing time, network delay, task failure, and computational resource utilization. The results highlight the potential benefits of edge and cloud cooperation in the three application scenarios, demonstrating that it significantly improves resource utilization and reduces the task failure rate compared to other widely adopted architectures, such as edge- or cloud-only architectures.

INDEX TERMS Edge-cloud architecture, IoT infrastructure, Edge computing, Urban computing, Smart cities, Urban mobility

I. INTRODUCTION

The rapid spread of Internet of Things (IoT) devices is generating huge volumes of data at the network edge [1]. Managing this data flow using highly centralized solutions, such as those based on cloud platforms, is extremely ineffective in terms of response time, network traffic management, power consumption, and scalability [2]. Uploading such huge volumes of data directly to the cloud leads to significant consumption of bandwidth and requires the use of high-power computing solutions to manage the resulting workload. Furthermore, in many application fields such as medicine and security, it is essential to offer low-latency and privacy-preserving services, as data transfer delay or malicious data manipulation can cause significant disservices and even loss of life [3].

In recent years, researchers and IT companies have proposed the adoption of the edge computing paradigm for processing data closer to where they are generated [4]. In this way, the following advantages can be achieved: i) *low latency*, since the computation takes place close to the data source; ii) *energy saving*, as battery-limited devices could offload computing tasks to edge servers for reducing energy consumption; iii) *privacy preserving*, since data are not necessarily uploaded to the cloud, but are processed and analyzed locally; and iv) *scalability*, as a strongly decentralized and distributed approach allows to manage increasing workloads efficiently. The benefits deriving from solutions based on edge computing can be complemented by using those provided by the cloud, as the latter allows to aggregate large amounts of data persistently and perform compute-intensive analyzes using scalable computational resources.

For all these benefits, edge-cloud continuum solutions are increasingly being proposed for new frontier application scenarios such as smart cities, industrial IoT and smart

healthcare [5], [6]. Particularly, in the field of urban mobility, the process of collecting, integrating and analyzing data generated from many sources can greatly benefit from scalable architectures and proximity solutions [7]. For example, tasks like driver assistance, collision avoidance and traffic sign recognition, which require real-time analysis and low response times, can benefit from edge computing [8]. Differently, tasks like diagnostic data collection and analysis, route calculations and targeted advertising, which require a lot of computational resources and access to large datasets, can benefit from the use of cloud computing.

In this paper we analyze how the compute continuum can be exploited to efficiently manage tasks related to urban mobility in large-scale computing environments. In particular, an edge-cloud continuum architecture is exploited to analyze geotagged data generated at the network edge by the movements of IoT objects such as taxis, cars, and smartphones. Once collected, these data can be analyzed through machine learning algorithms in real-time to provide solutions to different problems in our daily life. For example, i) for taxis, discovering the location to which they will have to move to more likely find new passengers; ii) for cars, delivering targeted advertising based on the positions and interests of drivers; and iii) for tourists, recommending new points of interest to visit based on what they like. The main contributions of this work are (i) the description of three application scenarios, in which geotagged data, generated during the movements of IoT objects (e.g., taxis, cars, smartphones), are collected and processed by machine learning algorithms (i.e., next location prediction, location-based advertising, and points of interest recommendation); (ii) a modeling part that defines an edge-cloud continuum architecture able to manage a large number of IoT devices and to efficiently execute machine learning algorithms to analyze the data they generate; (iii) an experimental part in which different design choices are evaluated (e.g., number of devices, type of task, orchestration policies) to improve the performance of machine learning algorithms in terms of processing time, network delay, task failure and computational resource utilization. By evaluating different application scenarios in a real-world environment (the city of Rome) and using settings derived from actual data, we provide a complete and advanced understanding of the benefits of edge-cloud architectures for urban mobility management.

The achieved results showed that the use of an edge-cloud continuum architecture, supported by efficient orchestration policies (e.g., network- or utilization-based), improves resource utilization and ensures a lower task failure rate in comparison to the traditional cloud- or edge-only configurations, where data are entirely processed at the cloud or the edge respectively. Specifically, for all the considered application scenarios, the orchestration policies were able to obtain a significant reduction in processing time (up to 87% compared to the edge-only configuration), a drastic reduction of the number of failed tasks (up to 40% compared to both cloud- and edge-only configurations), and a good lowering of resource utilization (up to 38% compared to the edge- and cloud-only configurations).

The structure of the paper is as follows. Section II discusses related work and introduces the problem statement. Section III describes the proposed edge-cloud continuum architecture. Section IV presents three application scenarios as case studies and a performance evaluation by using two orchestration policies. Finally, Section V concludes the paper.

II. RELATED WORK

Urban computing is a research field that focuses on the study and development of systems and methods for supporting decision-making in urban environments using data generated in cities [9]. In particular, urban mobility is a sub-field of urban computing that refers to the mobility of people and vehicles within cities, including the challenges and opportunities associated with the planning, management and optimization of urban transport systems [10]. The analysis of large amounts of geotagged data generated by IoT devices installed on means of transport and road infrastructures can be used for many purposes, including traffic flow monitoring and transport route planning, decision-making to improve the quality of urban life and the provision of location-based services to citizens [11].

In this scenario, the edge-cloud compute continuum has emerged as a solution to process and analyze the data generated by IoT devices efficiently and in real-time [12], [13]. However, effective resource allocation and orchestration strategies are critical for maximizing the benefits of edge-cloud computing, and for this reason, researchers have focused on optimizing the placement of tasks and data in edge-cloud systems, considering factors such as performance, energy efficiency, cost and reliability [14], [15]. To this end, in the literature different techniques have been proposed that make use of supervised/unsupervised machine learning, deep learning and reinforcement learning [16].

Designing and testing large-scale and multi-layer edge-cloud architectures are still open issues, especially for architectures composed of several components based on different technologies and software stacks [17], [18]. Using a large number of hardware devices for prototyping could be very expensive, as well as setting up real-world experiments could be logistically challenging [19]. For these reasons, simulating edge-cloud continuum solutions is important because it allows testing and evaluating system performance to identify and resolve problems or limitations before the deployment in a real context [20]. In particular, the simulation of edge-cloud architectures allows evaluating many aspects including i) the scalability of the system and its ability to manage large amounts of data generated by IoT devices; ii) the latency of the system, i.e., the time between data collection and processing, ensuring that the system provides results in real-time; iii) the ability to exploit both edge and cloud resources efficiently, to optimize data processing and transmission and to ensure energy sustainability as well. The main issues of modeling IoT systems and how simulation approaches can

assist the design and validation of edge-cloud architectures are discussed in different research papers [21]–[23].

In terms of tools and software solutions, different open-source simulators have been proposed in recent years to simulate IoT environments, such as iFog-Sim [24], IoTSim [25] and EdgeCloudSim [26]. Several research works have made use of simulators to test the behavior of specific IoT applications on edge-cloud architectures [27]–[30]. Unlike these, our work analyzes how a large-scale edge-cloud architecture can be leveraged to efficiently manage urban mobility applications based on machine learning. Through three application scenarios, we show how the data generated by different IoT devices can be efficiently managed and processed using an edge-cloud continuum architecture.

A. APPLICATION SCENARIOS

Urban mobility data can be used in multiple ways to improve people's quality of life and make cities more efficient and sustainable. For example, they can be used for traffic monitoring, route planning and transportation management, among others. We have decided to focus our attention on three different use cases where the geotagged data generated by three different types of IoT objects (i.e., taxis, cars and smartphones) are analyzed through machine learning algorithms. Specifically, the following cases are considered: i) the location to which taxis will have to move to more likely find new passengers; ii) targeted advertising based on the positions and interests of car drivers; and iii) suggestion of the next points to visit based on tourist preferences and behaviors.

Geotagged data generated by taxis can be used to predict their next destination, reducing route costs and traffic congestion. Unlike other forms of public transportation, taxis do not have fixed routes and plan their routes after a passenger is dropped off [31]. GPS trackers in taxis allow for real-time monitoring of the vehicle's location and trajectory analysis can be used to predict where a taxi will move next, known as the *next location prediction* problem [32], which can be modeled as a short-term or long-term prediction task. There are several methods in the literature for this problem, such as frequent patterns and association rules [33], [34], or machine learning-based methods like clustering and Markov chain-based framework [35] or neural network-based models [36].

Geotagged data from vehicles can be leveraged for *location-based advertising*, which can provide car drivers with relevant products, services and offers based on their habits while they are on the road. Popular approaches include using GPS data [37] from the driver's in-car navigation system to deliver location-based ads and offers (e.g., a driver passing by a restaurant might receive a coupon for a discounted meal), as well as using contextual data [38] such as time of day and traffic conditions (e.g., a driver stuck in traffic might receive an ad for a nearby coffee shop).

Similarly, geotagged data generated by people during their movements can be used to provide insights for destination planning, service design and marketing [39]. To achieve this, data mining algorithms are used to discover frequent patterns in user trajectories across interesting locations frequently visited by users, commonly referred to as Points-of-Interest (PoIs) [40]. A common application is related to *PoIs recommendation* to suggest places to visit based on a tourist's route collected from the smartphone during a trip for improving touristic services [41], [42]. Machine learning and data mining models have been used in previous work to solve this problem, such as using a bidirectional LSTM neural network [43] or sequential pattern analysis [44].

B. MACHINE LEARNING SOLUTIONS FOR LARGE URBAN AREAS

With the growth of urban areas and the number of IoT devices, there is an increasing need to develop machine learning algorithms that can scale efficiently on distributed architectures such as those of the edge-cloud continuum. Federated learning, through the hierarchical aggregation of learning models, has emerged as a promising paradigm for overcoming the limitations of traditional centralized approaches, such as those related to bandwidth, latency, and centralized data processing and storage. In federated learning, data are kept on local devices and only model updates are shared, ensuring greater scalability and efficiency but also privacy preservation of sensitive data, making it suitable for large-scale IoT environments such as those of the Internet of Vehicles (IoV) and Intelligent Transportation Systems (ITS) [45].

Recently, several frameworks have been proposed in the IoV that use federated learning. For instance, Balasubramanian et al. [46] proposed a cooperative edge intelligence framework that uses a hybrid stacked autoencoder model called VeNet to perform anomaly detection and classification tasks among multiple edge devices in a decentralized manner. It consists of a local autoencoder that is trained on data collected by each edge device, and a global autoencoder that is trained on a subset of the data from all the edge devices. Similarly, Zhou et al. [47] introduced a novel two-layer federated learning framework for IoV that allows for aggregating models with different architectures and hyperparameters. This approach allows for more flexibility in model selection and greater performance of federated learning frameworks. Overall, the experimental evaluations on real-world and large-scale datasets demonstrate the scalability, efficiency, and potential benefits of using federated learning in urban computing contexts, making it suitable in large-scale IoT environments.

III. SYSTEM ARCHITECTURE

Although cloud computing provides high scalability with dynamic resource allocation, it may raise performance issues as a result of the centralization of data collection and processing [48], [49]. On the other hand, an edge-cloud continuum architecture might address these issues by enabling efficient and fast management of the massive volume of data generated by IoT devices. In particular, these architectures enhance computation capabilities and scalability while reducing net-

work congestion and failed tasks. For these reasons, such architectures can also have a significant impact on urban mobility applications. Figure 1 shows a three-layer edge-cloud continuum architecture for supporting urban mobility.

The edge-cloud continuum leverages all the resources from the edge of the network (e.g., IoT devices) to the core (e.g., cloud data centers) [50]. Specifically:

- The *device layer* includes the components that are leveraged by vehicles and humans to share information during their movements across different urban cells, which define a partitioning of an urban area. These components (e.g., GPS, infotainment devices, on-board cameras) produce a very high volume of data in different formats and in real-time, which is sent to the edge server of the current cell. This data can be combined with the personal data of the users (e.g., preferences and behaviors) and information about the surrounding environment, to deliver advanced, customized and context-aware services.
- The *edge layer* includes heterogeneous hardware components (e.g., gateways, micro data centers), which serve as elements of the infrastructure that collect and partially process raw data generated at the device layer.
- The *cloud layer* provides access to a large set of computing and storage resources, which can be dynamically allocated for executing tasks that cannot be performed by edge servers. From the client's perspective, the cloud is an abstraction for remote and infinitely scalable computing and storage resources. For these reasons, it has emerged as an effective computing paradigm to meet the challenge of processing big data in a limited time and to provide an efficient data analysis environment.

The edge layer includes a key component called *Edge Orchestrator (EO)*, which is responsible for managing and coordinating the execution of tasks, determining whether each task will run on the edge or cloud. It can be programmed to apply different orchestration policies to optimize the overall performance of the architecture. These policies can take into consideration many parameters, such as network congestion, data volume to be processed, and status and load level of both edge nodes and cloud. Two orchestration policies were employed in this work, namely *network-based (edge/cloud-NB)* and *utilization-based (edge/cloud-UB)*, whose pseudocode is shown in Algorithm 1. In particular, for each task to be scheduled, the cell and the associated edge server are identified from the coordinates of the IoT object generating that task (lines 3-4). Then, the desired orchestration policy (i.e. utilization-based or network-based) is applied to decide where the incoming task must be executed. Specifically, the utilization-based policy schedules tasks based on the utilization of edge nodes (lines 6-12). If the average edge utilization is greater than a fixed threshold (i.e., θ_1), the incoming task is offloaded to the cloud (lines 8-9); otherwise, it is assigned to the edge layer (lines 10-11). The network-based orchestration policy (lines 14-21) measures the network delay from the device that generated the task to the cloud (line 14). For deciding where it must be executed, a dummy task that uploads and downloads 1 MB of data is exploited. In detail, the algorithm measures the upload delay which includes both the transmission delay (i.e., the time required to transmit the data over the network) and the processing delay (i.e., the time required for the cloud to process the request). Particularly, the transmission delay includes both the time required to transmit the request and response over the network, which depends on the size of the data being transmitted and the available transmission rate, and propagation delay, which is due to the distance between the server and the cloud. Then, this delay is leveraged to determine the percentage of used bandwidth compared to the maximum bandwidth (line 15). If it is less than a fixed threshold (i.e., θ_2), the incoming task is offloaded to the cloud (lines 16-17); otherwise, it is assigned to the edge layer (lines 18-19). In the end, according to the chosen layer, the task is assigned to the cloud or the edge server of the current cell (line 22).

In the experimental section, the values of thresholds were chosen according to conventions often used on cloud platforms. Indeed, in different technical reports [51]–[53], a threshold value is used to determine when to scale the computing resources (e.g., 80% of the total resources). This is because, if the percentage of resource utilization reaches the threshold value, it can indicate that such resources are under pressure and may not be able to handle any further requests.

Algorithm 1 Edge Orchestrator

```
1:  Initializing EO and orchestration policy p.
2:  procedure GETSERVER(task, coord, θ1, θ2)
3:      cell ← getCell(coord)
4:      edgeS ← getEdgeServer(cell)
5:      layer ← null
6:      if p == Utilization_Based then
7:          edgeUtilization ← getEdgeUtilization()
8:          if edgeUtilization>θ1 then
9:              layer ← CLOUD
10:         else
11:             layer ← EDGE
12:         end if
13:     else
14:         wanDelay ← getUpDelay(task.getDevice(), CLOUD)
15:         wanUBW ← getBandwidthUtilization(wanDelay)
16:         if wanUBW<θ2 then
17:             layer ← CLOUD
18:         else
19:             layer ← EDGE
20:         end if
21:     end if
22:     return (layer == EDGE)?edgeS : cloud
23: end procedure
```

IV. EXPERIMENTAL EVALUATION

To evaluate the performance of the proposed edge-cloud continuum architecture, we used the EdgeCloudSim simulator and considered three different urban mobility scenarios (taxis, cars and tourists with smartphones). Among the open-

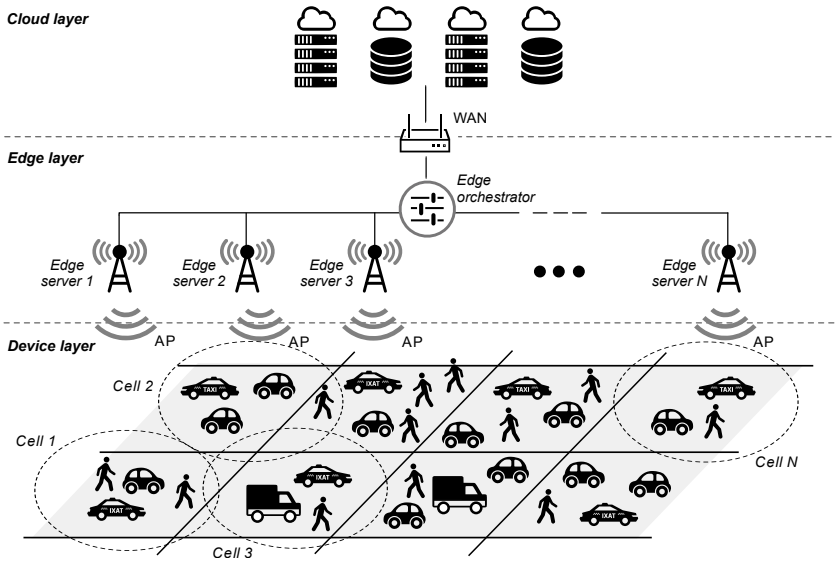


FIGURE 1. The edge-cloud continuum architecture.

TABLE 1. Description of the main EdgeCloudSim simulation parameters

Parameter	Description
Num. of IoT objects	Number of IoT objects used in the simulation scenarios.
Num. of edge servers	Number of edge servers.
MIPS for edge server VM	Computing processor's speed of edge servers in terms of Million Instructions Per Second.
MIPS for cloud VM	Computing processor's speed of cloud in terms of Million Instructions Per Second.
Poisson interarrival	Mean interarrival time between two tasks.
Active period	The active period of the task.
Idle period	The idle period of the task.
Upload data size	Mean input file sizes to upload.
Download data size	Mean output file sizes to download.
Task length	Mean number of instructions to execute the incoming task.

source simulators discussed in Section II, we have chosen EdgeCloudSim, which is particularly well-suited for modeling urban mobility scenarios, since it supports different architectures, devices, and device mobility [26]. Table 1 reports the main parameters required by the simulator along with their description.

The three different applications are concerned with urban mobility in which machine learning algorithms are used to analyze large sets of geotagged data generated during the movements of IoT objects. In particular:

1) *Application scenario 1* is about the taxi destination prediction problem, aimed at establishing the next position where taxis will have to move to have a better chance of finding new customers.
2) *Application scenario 2* models the problem of delivering location-based and targeted advertising to car drivers based on the position of the car and the interests of the driver.
3) *Application scenario 3* concerns the next location recommendation problem applied to tourists, aimed at suggesting new points of interest to visit based on their movements collected by their smartphones.

For each scenario, we considered three common tasks:

- *Data collection task*: it consists in collecting and preprocessing the data generated at the device layer (e.g., data generated by IoT objects).
- *Training task*: it consists in training a machine learning model, which is regularly updated with new mobility patterns. In these experiments, we used a centralized approach for model update. This turns out to be an appropriate choice according to the size of the urban area and the number of devices considered, in contrast to the approaches based on federated learning which are more suitable for larger urban areas scenarios.
- *Prediction task*: it exploits the trained model for suggesting the next location where a taxi should move to find new passengers or to provide location-based advertising and suggestions to car drivers and tourists.

Table 2 reports the main parameters used to configure the simulations, which have been extracted from official reports of public administrations or scientific papers. In particular, we used Rome in Italy as the reference city, and according to the official report [54] we have defined the number of taxis, cars and tourists, i.e. 10K, 100K and 100K respectively. The city has been divided into 100 cells covering about $1km^2$ each. The infrastructure is composed of an edge layer with 100 edge servers configured as a virtual machine (VM) having 4 cores, 4 GB of RAM and 64 GB of storage memory, and a cloud layer configured as a VM equipped with 8 cores, 32 GB of RAM and 1 TB of storage memory. In our simulations, IoT devices follow a Nomadic Mobility Model, in which the time a device remains in a cell before moving to a nearby one is taken from an exponential distribution. The mean

value of the exponential distribution is set to 140 seconds for application scenarios 1 and 2, and at 900 seconds for application scenario 3. In fact, considering that the average speed of a vehicle in the center of Rome is estimated at 26 km/h [55] and that the city area has been divided into cells covering areas of $1km^2$, vehicles at this speed cross a cell on average in about 2.3 minutes (i.e., 140 seconds). Instead, the average speed of a pedestrian walking at a slow pace is 4 km/h [56], therefore the time taken to pass from one cell to another is about 15 minutes (i.e., 900 seconds).

The training and inference times together with the information on the hardware characteristics reported in [35], [40] and [44], have been used to determine the type of tasks and their average length for application scenario 1, 2 and 3 respectively. Moreover, for each task (i.e., data collection, training and prediction) in each application scenario, a set of parameters are required, including the Poisson interarrival, the active/idle period time of tasks, and the amount of data that is downloaded and uploaded. The Poisson interarrival time is used to define the rate at which the devices generate the different types of tasks. Overall, for data collection and prediction tasks modeling, the interarrival times were set to low values to reflect the high level of devices' activity in the urban area under consideration. Conversely, higher values were chosen for the training tasks to model that they are less recurring (e.g., periodic retraining of the machine learning model might occur once a day). The active and idle periods are used to control the amount of time a device spends actively generating or not generating a specific task, while the upload/download data sizes control the amount of data generated and transmitted by devices in the simulation. For example, large upload and download data sizes indicate a task with high data transmission requirements, such as the prediction and data collection tasks that involve significant data exchange.

Among all the parameters described, the number of IoT objects, the Poisson interarrival time and the task length are the ones that mostly drive the results of our simulations and, for this reason, we mainly focused on them to define the different tasks in the application scenarios. Other parameters required to configure and run the simulations (e.g., active/idle period and download/upload data size) were defined differently for each task but uniformly across scenarios.

A. PERFORMANCE EVALUATION

We carried out a large number of experiments to evaluate the edge-cloud continuum architecture. To make the simulation results more significant, we repeated the experiments 10 times for each input configuration and reported the mean values. The experiments are used to assess the behavior of the edge-cloud continuum solution compared to centralized ones that exploit only cloud or edge resources. Specifically, the four configurations we evaluated are the following:

- *Cloud-only*: tasks are performed exclusively on the cloud.
- *Edge-only*: tasks are performed directly on the edge.

TABLE 2. Simulation parameters for the three scenarios.

Parameter	Scenario1 (Taxis)	Scenario2 (Cars)	Scenario3 (Tourists)
Num. of IoT objects	$10k$	$100k$	$100k$
Num. of edge servers	100	100	100
Edge processing speed (MIPS)	$2.5k$	$2.5k$	$2.5k$
Cloud processing speed (MIPS)	$300k$	$300k$	$300k$
WLAN bandwidth (Mbps)	300	300	300
WAN bandwidth (Mbps)	150	150	150
Waiting time in a cell (s)	140	140	900
Data collection task			
Task length (MI)	$25k$	$25k$	$25k$
Poisson interarrival (s)	$1k$	$1.5k$	$2k$
Active period (s)	10	10	10
Idle period (s)	10	10	10
Upload data size (KB)	200	200	200
Download data size (KB)	1	1	1
Training task			
Task length (MI)	$60M$	$30M$	$30M$
Poisson interarrival (s)	$10k$	$35k$	$35k$
Active period (s)	500	500	500
Idle period (s)	10	10	10
Upload data size (KB)	1	1	1
Download data size (KB)	1	1	1
Prediction task			
Task length (MI)	$50k$	$40k$	$35k$
Poisson interarrival (s)	600	900	1000
Active period (s)	5	5	5
Idle period (s)	10	10	10
Upload data size (KB)	200	200	200
Download data size (KB)	200	200	200

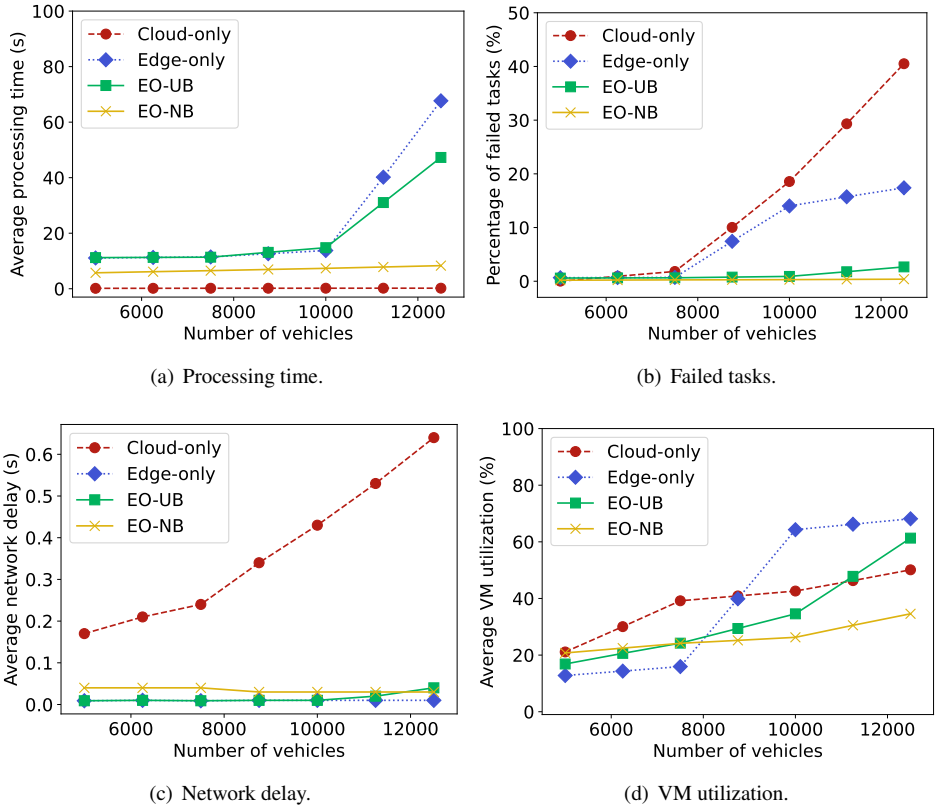
- *Edge/cloud-UB* and *edge/cloud-NB*: tasks are performed locally on edge servers or remotely in the cloud based on the policy of the edge orchestrator (i.e., network-based and utilization-based).

Regarding the edge/cloud configurations, as discussed in Algorithm 1, the decision whether to offload a task to the cloud or perform it on the edge server is driven by two main parameters, i.e. the two thresholds θ_1 and θ_2. In the experimental evaluation of all application scenarios, the threshold for the utilization- and network-based policies was set at 80% [51]–[53], which means that the computing and network resources are preserved from being used no more than 80% of their capacity to avoid their saturation.

The configurations were evaluated and compared on four different metrics, which are the average processing time, percentage of failed tasks, network delay and VM utilization.

1) Application scenario 1: next location prediction for taxis

In this section, we present the main results we obtained for the scenario related to taxi destination prediction. Figure 2 reports the performance metrics for each of the four configurations (i.e., cloud-only, edge-only, edge/cloud-NB, and edge/cloud-UB). As stated before, the application is modeled to simulate the city of Rome, which has around $10k$ taxi licenses according to official data [54]. However, we considered a variable number of taxis, ranging from $5k$ to $12.5k$, to investigate how a different number of taxis can impact the performance metrics in the different configurations.



(a) Processing time.

(b) Failed tasks.

(c) Network delay.

(d) VM utilization.

FIGURE 2. Performance results for the different configurations (cloud-only, edge-only, edge/cloud-UB and edge/cloud-NB) for application scenario 1.

The average processing time obtained by the different configurations is shown in Figure 2(a). The achieved results are stable and consistent over the different runs and exhibit low variance. On average, the relative standard deviation is at most 4% compared to the mean value over the 10 runs. In particular, the edge-only showed the worst results, with a significant drop in performance as the number of vehicles increased (the processing time increases from around 10 seconds with $5k$ vehicles up to around 70 seconds with $12.5k$ vehicles). Instead, the cloud-only configuration achieved a very low average processing time. However, it dramatically increases the number of failed tasks as the number of vehicles increases. In fact, as shown in Figure 2(b), the percentage of failed tasks for the cloud- and edge-only architectures increases rapidly as the number of vehicles increases. In particular, a steep increase can be observed when using more than $7.5k$ vehicles: this means that, as long as there are few vehicles, the cloud-only architecture can handle the incoming workload better than the other configurations, but as the number of devices increases it leads to a higher percentage of failed tasks. On the other hand, the use of the edge orchestrator leads to a lower task failure rate (on average 6.8% for edge/cloud-NB and 1.3% for edge/cloud-UB). This is a crucial aspect to be considered since in many contexts having a high number of failed tasks can compromise the usability of the IoT application. Also for this metric, the results obtained show a low variance, with a relative standard deviation that is at most 2% with respect to the mean value.

Figure 2(c) shows the average network delay. In particular, it emerges how the cloud-only configuration generates a very high network delay because of data transfer from the edge layer to the cloud, resulting in a significant increase in communication delay (up to around 98% higher than the edge-only solution), while processing data locally at the edge does not produce significant effects. For the network delay, simulation results showed a negligible relative standard deviation below the 1%.

Figure 2(d) illustrates the average VM utilization obtained by the different simulated configurations, with a relative deviation from the mean value being at most 7%. The edge/cloud-NB achieved the best result showing a low utilization of resources while keeping, as discussed, a low processing time and a low percentage of failed tasks. Reducing the use of VMs is a crucial aspect in large-scale applications that involve large computational resources because it allows for optimizing costs and energy consumption. Additionally, reducing the risk of saturating computational resources allows for handling any unexpected workload peak that may occur. It should be also noted that the edge-only configuration produces a significant increase in the VM utilization for a high number of vehicles, but it still achieves a lower task failure rate than the cloud-only one (see Figure 2(b)). If

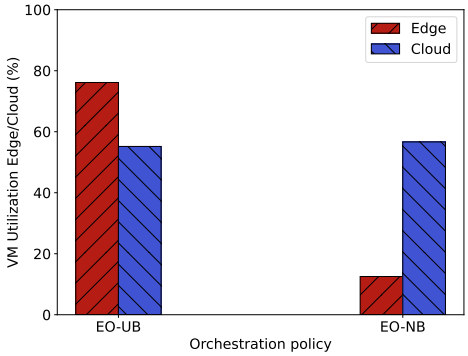


FIGURE 3. Average VM utilization on both cloud and edge with the two orchestration policies for application scenario 1.

we analyze in detail the percentage of VM utilization for the two edge/cloud configurations, we can get more details about the behavior of the edge orchestrator. In particular, Figure 3 shows the percentage of VM utilization on both cloud and edge when considering $12.5k$ vehicles. In this case, the utilization-based policy results in a higher utilization of the edge resources (73% compared to 49% of cloud), while the network-based policy produces a higher utilization of cloud resources (57% compared to 12% of edge).

It is worth noting that a task can fail for one of three reasons: VM capacity, low network bandwidth or due to mobility. In particular, if the utilization of a VM is too high, it may reject incoming tasks. Similarly, if too many vehicles connect to the same edge server, the network may become congested and tasks may fail. Finally, a task may fail due to the vehicle moving from one cell to another, according to the Nomadic Mobility Model. As an example, if we analyze the percentage of failed tasks in the cloud-only configuration, we find out that only the 0.02% fails due to low computation capacity, as the cloud has enough computational resources, while almost all failed tasks are due to network congestion. On the other hand, the edge/cloud-NB is able to balance data traffic between cloud and edge, avoiding sending traffic over the WAN when it is congested.

Overall, the edge/cloud-UB and edge/cloud-NB showed the best results, outperforming the conventional cloud- or edge-only architectures. Compared to the edge-only architecture, the use of the edge orchestrator leads to a drastic reduction in processing time, which ranges from 30% for edge/cloud-UB to 87% for edge/cloud-NB. In addition, compared to both cloud- and edge-only architectures, it permits to reduce the number of failed tasks (up to 38% for edge/cloud-UB and 40% for edge/cloud-NB) and the VM utilization (up to 29% for edge/cloud-UB and 38% for edge/cloud-NB).

2) Application scenarios 2 and 3: location-based advertising for car drivers and PoIs recommendation for tourists

In this section we present the main results obtained by simulating application scenarios 2 and 3, which model the problem of location-based advertising for car drivers and points of interest recommendation for tourists. For the sake of brevity, we have not considered a variable number of IoT objects (i.e., vehicles and people), but only the one closest to the real one. In particular, we considered $100k$ IoT objects for both scenarios, which is about the number of cars and tourists that move around the city of Rome every day [54].

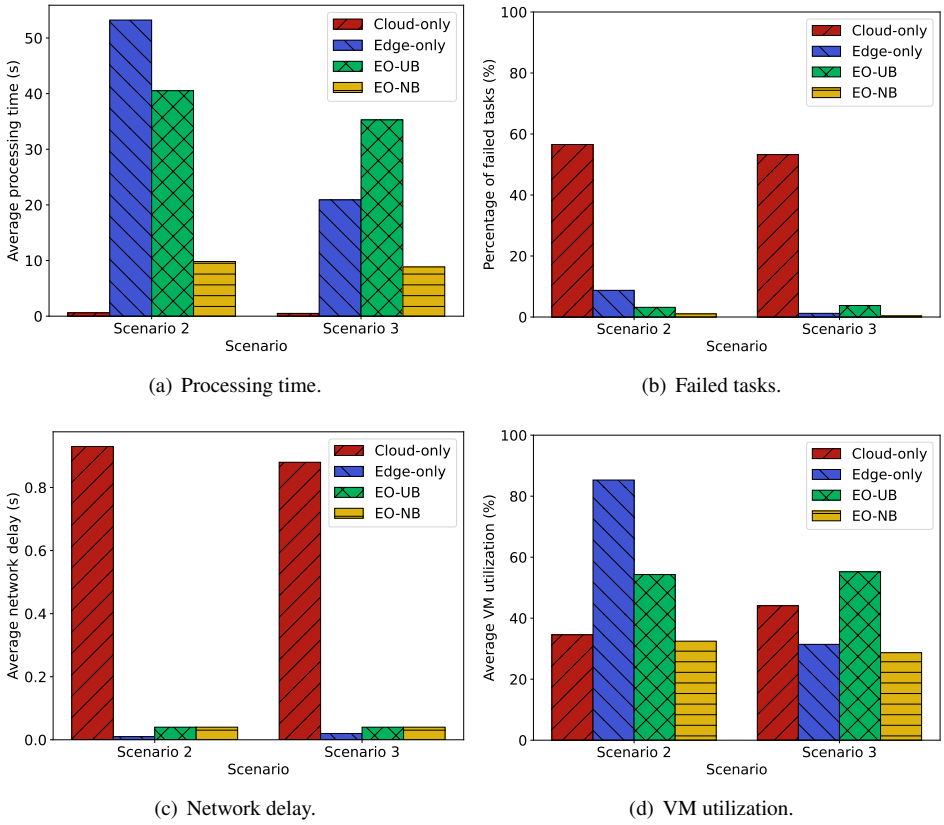
Figure 4 reports the performance metrics for each of the four configurations, i.e., cloud-only, edge-only, edge/cloud-UB and edge/cloud-NB. Specifically, Figures 4(a), 4(b), 4(c) and 4(d) report the average processing time, percentage of failed tasks, network delay and VM utilization, respectively.

In both simulated scenarios, the edge/cloud-NB configuration shows a better processing time than both edge-only and edge/cloud-UB (up to 81% and 76% lower respectively). It should be noted that the computational resources of the cloud allow for a lower processing time. However, as for application scenario 1, the cloud-only configuration is affected by a high percentage of failed tasks and high network delay. Indeed, leveraging both edge and cloud resources, as long as the network is not congested and the WAN delay is negligible, reduces the time required to complete a task.

Concerning failed tasks, the edge/cloud configurations obtain very low failure rates (less than 4%). Particularly, Figure 5 details the percentage of failed tasks due to network congestion, VM capacity and mobility for both configurations. In these two scenarios, mobility from one cell to another is the main cause of task failure. However, especially in application scenario 2, an important part of the tasks fails due to a lack of resources at the edge layer.

Regarding network delay, the simulation results did not reveal significant differences between the two orchestration policies in the edge-cloud continuum, while the cloud-only configuration is heavily affected by data transfer from the edge layer.

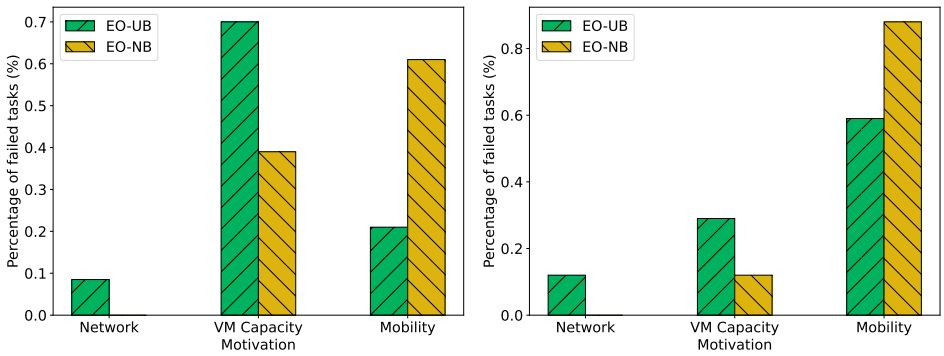
Finally, regarding VM utilization, the edge/cloud-NB and edge/cloud-UB showed a lower percentage than the cloud- and edge-only configurations, reducing the risk of saturating computational resources and allowing for better management of the incoming workload. In particular, the average VM utilization of the edge/cloud-NB is up to 22% and 48% lower than the edge/cloud-UB for the two simulated scenarios. Overall, the edge/cloud-NB configuration performed better



(a) Processing time. (b) Failed tasks.

(c) Network delay. (d) VM utilization.

FIGURE 4. Performance results for the different configurations (cloud-only, edge-only, edge/cloud-UB and edge/cloud-NB) for application scenarios 2 and 3.



(a) Application scenario 2. (b) Application scenario 3.

FIGURE 5. Percentage of failed tasks due to network congestion, VM capacity and mobility for application scenarios 2 and 3.

than the edge/cloud-UB configuration, reducing the processing time by 76%, the number of failed tasks by 3%, and the VM utilization by 27% on average.

V. CONCLUSION AND FUTURE WORK

With the pervasive diffusion of IoT devices, the edge-cloud continuum has been proposed to combine the advantages of edge computing in processing data closer to where they are generated with those of the cloud in supporting compute-intensive tasks. In this paper, we explored the use of edge-cloud architectures for supporting three urban mobility scenarios (i.e., next location prediction, location-based advertising, and of points of interest recommendation), in which machine learning algorithms are used to analyze large sets of geotagged data generated during the movements of IoT objects (e.g., taxis, cars, smartphones).

Several experiments have been carried out for assessing the benefits of the edge-cloud continuum over the traditional cloud- or edge-only architectures. In particular, we exploited a simulation-based approach for designing and testing IoT

applications by using an edge-cloud simulator and two orchestration policies, based on network (edge/cloud-NB) and computational resources (edge/cloud-UB) utilization. The achieved results demonstrated that the edge-cloud continuum architecture, coupled with the defined orchestration policies, outperforms traditional cloud- or edge-only architectures, obtaining a significant reduction in processing time, task failure rate, and resource utilization.

Future research efforts will be devoted to developing advanced orchestration policies that can exploit machine and deep reinforcement learning to improve task scheduling in the edge-cloud continuum. Such policies can be further tested using emulators instead of simulators to evaluate how software interacts with the underlying hardware. Furthermore, in ever-growing urban areas with ever-increasing numbers of IoT devices, it will also be necessary to think about how algorithms can scale efficiently on edge-cloud architectures. Hence, future work should evaluate how machine learning paradigms such as federated learning can overcome the limitations of centralized solutions in large-scale IoT environments.

DATA AND CODE AVAILABILITY STATEMENT

In order to reproduce the experiments reported in the paper, the open-source version of EdgeCloudSim is available on https://github.com/CagataySonmez/EdgeCloudSim, while all the parameters required to run the simulations are reported in the paper.

ACKNOWLEDGMENT

We acknowledge financial support from “National Centre for HPC, Big Data and Quantum Computing”, CN00000013 - CUP H23C22000360005, and from “PNRR MUR project PE0000013-FAIR” - CUP H23C22000860006.

A. REFERENCES

REFERENCES

[1] Loris Belcastro, Riccardo Cantini, Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Programming big data analysis: Principles and solutions. Journal of Big Data, 9(4), 2022.

[2] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. IEEE access, 8:85714–85728, 2020.

[3] Lanfang Sun, Xin Jiang, Huixia Ren, and Yi Guo. Edge-cloud computing and artificial intelligence in internet of medical things: Architecture, technology and application. IEEE Access, 8:101079–101092, 2020.

[4] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. Edge computing: A survey. Future Generation Computer Systems, 97:219–235, 2019.

[5] PJ Escamilla-Ambrosio, A Rodríguez-Mota, E Aguirre-Anaya, R Acosta-Bermejo, and M Salinas-Rosales. Distributing computing in the internet of things: cloud, fog and edge computing overview. In NEO 2016: Results of the Numerical and Evolutionary Optimization Workshop NEO 2016 and the NEO Cities 2016 Workshop held on September 20-24, 2016 in Tlalnepantla, Mexico, pages 87–115. Springer, 2018.

[6] Benazir Neha, Sanjaya Kumar Panda, Pradip Kumar Sahu, Kshira Sagar Sahoo, and Amir H Gandomi. A systematic review on osmotic computing. ACM Transactions on Internet of Things, 3(2):1–30, 2022.

[7] Nabeela Awan, Ahmad Ali, Fazlullah Khan, Muhammad Zakarya, Ryan Alturki, Mahwish Kundi, Mohammad Dahman Alshehri, and Muhammad Haleem. Modeling dynamic spatio-temporal correlations for urban traffic flows prediction. IEEE Access, 9:26502–26511, 2021.

[8] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. IEEE Access, 6:6900–6919, 2018.

[9] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology (TIST), 5(3):1–55, 2014.

[10] Sara Paiva, Mohd Abdul Ahad, Gautami Tripathi, Noushaba Feroz, and Gabriella Casalino. Enabling technologies for urban smart mobility: Recent trends, opportunities and challenges. Sensors, 21(6):2143, 2021.

[11] Kenneth Li-Minn Ang, Jasmine Kah Phooi Seng, Ericmoore Ngharamike, and Gerald K Ijemaru. Emerging technologies for smart cities' transportation: Geo-information, data analytics and machine learning approaches. ISPRS International Journal of Geo-Information, 11(2):85, 2022.

[12] Luiz Bittencourt, Roger Immich, Rizos Sakellariou, Nelson Fonseca, Edmundo Madeira, Marilia Curado, Leandro Villas, Luiz DaSilva, Craig Lee, and Omer Rana. The internet of things, fog and cloud continuum: Integration and challenges. Internet of Things, 3:134–155, 2018.

[13] Xavi Masip-Bruin, Eva Marin-Tordera, Admela Jukan, and Guang-Jie Ren. Managing resources continuity from the edge to the cloud: Architecture and performance. Future Generation Computer Systems, 79:777–785, 2018.

[14] Bo Wang, Changhai Wang, Wanwei Huang, Ying Song, and Xiaoyun Qin. A survey and taxonomy on task offloading for edge-cloud computing. IEEE Access, 8:186080–186101, 2020.

[15] Congfeng Jiang, Xiaolan Cheng, Honghao Gao, Xin Zhou, and Jian Wan. Toward computation offloading in edge computing: A survey. IEEE Access, 7:131543–131558, 2019.

[16] Bin Cao, Long Zhang, Yun Li, Daquan Feng, and Wei Cao. Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework. IEEE Communications Magazine, 57(3):56–62, 2019.

[17] Mohsen Marjani, Fariza Nasaruddin, Abdullah Gani, Ahmad Karim, Ibrahim Abaker Targio Hashem, Aisha Siddiqa, and Ibrar Yaqoob. Big iot data analytics: architecture, opportunities, and open research challenges. ieee access, 5:5247–5261, 2017.

[18] Fabrizio Marozzo, Alessio Orsino, Domenico Talia, and Paolo Trunfio. Edge computing solutions for distributed machine learning. In 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pages 1–8. IEEE, 2022.

[19] Ana Isabel Torre-Bastida, Javier Del Ser, Ibai Laña, Maitena Ilardia, Miren Nekane Bilbao, and Sergio Campos-Cordobés. Big data for transportation and mobility: recent advances, trends and challenges. IET Intelligent Transport Systems, 12(8):742–755, 2018.

[20] Maria Salama, Yehia Elkhatib, and Gordon Blair. Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, pages 251–261, 2019.

[21] Gabriele D'Angelo, Stefano Ferretti, and Vittorio Ghini. Simulation of the internet of things. In 2016 International Conference on High Performance Computing & Simulation (HPCS), pages 1–8. IEEE, 2016.

[22] Gabor Kecskemeti, Giuliano Casale, Devki Nandan Jha, Justin Lyon, and Rajiv Ranjan. Modelling and simulation challenges in internet of things. IEEE Cloud Computing, 4(1):62–69, 2017.

[23] Luis Eduardo Lima, Bruno Yuji Lino Kimura, and Valério Rosset. Experimental environments for the internet of things: A review. IEEE Sensors Journal, 19(9):3203–3211, 2019.

[24] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software: Practice and Experience, 47(9):1275–1296, 2017.

[25] Xuezhi Zeng, Saurabh Kumar Garg, Peter Strazdins, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, and Rajiv Ranjan. Iotsim: A simulator for analysing iot applications. Journal of Systems Architecture, 72:93–107, 2017.

[26] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. Transactions on Emerging Telecommunications Technologies, 29(11):e3493, 2018.

[27] Mluleki Sinqadu and Zelalem Sintayehu Shibeshi. Performance evaluation of a traffic surveillance application using ifogsim. In International Conference on Wireless Intelligent and Distributed Environment for Communication, pages 51–64. Springer, 2020.

[28] Fatin Hamadah Rahman, Thien Wan Au, SH Shah Newaz, and Wida Susanty Haji Suhaili. A performance study of high-end fog and fog cluster in ifogsim. In Computational Intelligence in Information Systems: Proceedings of the Computational Intelligence in Information Systems Conference (CIIS 2018) 3, pages 87–96. Springer, 2019.
[29] Alessandro Barbieri, Fabrizio Marozzo, and Claudio Savaglio. Iot platforms and services configuration through parameter sweep: a simulation-based approach. In 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 1803–1808, 17-20 October 2021.
[30] Sumit Maheshwari, Dipankar Raychaudhuri, Ivan Seskar, and Francesco Bronzino. Scalability and performance evaluation of edge cloud systems for latency constrained applications. In 2018 IEEE/ACM Symposium on Edge Computing (SEC), pages 286–299. IEEE, 2018.
[31] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In Proceedings of the 13th International Conference on Ubiquitous Computing, UbiComp '11, pages 109–118, New York, NY, USA, 2011. ACM.
[32] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. Modelling taxi drivers' behaviour for the next destination prediction. IEEE Transactions on Intelligent Transportation Systems, 21(7):2980–2989, 2019.
[33] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 637–646, 2009.
[34] A. Altomare, E. Cesario, C. Comito, F. Marozzo, and D. Talia. Trajectory pattern mining for urban computing in the cloud. IEEE Transactions on Parallel and Distributed Systems, 28(2):586–599, 2017.
[35] Punit Rathore, Dheeraj Kumar, Sutharshan Rajasegarar, Marimuthu Palaniswami, and James C Bezdek. A scalable framework for trajectory prediction. IEEE Transactions on Intelligent Transportation Systems, 20(10):3860–3874, 2019.
[36] Hao Lin, Guannan Liu, Fengzhi Li, and Yuan Zuo. Where to go? predicting next location in iot environment. Frontiers of Computer Science, 15(1):1–13, 2021.
[37] Christine Bauer and Christine Strauss. Location-based advertising on mobile devices. Management review quarterly, 66(3):159–194, 2016.
[38] Dominik Molitor, Philipp Reichhart, and Martin Spann. Location-based advertising and contextual mobile targeting. 2016.
[39] Weimin Zheng, Mengling Li, Zhibin Lin, and Yangyu Zhang. Leveraging tourist trajectory data for effective destination planning and management: A new heuristic approach. Tourism Management, 89:104437, 2022.
[40] Loris Belcastro, Fabrizio Marozzo, and Emanuele Perrella. Automatic detection of user trajectories from social media posts. Expert Systems with Applications, 186:115733, 2021.
[41] Dmitry Korzun, Ekaterina Balandina, Alexey Kashevnik, Sergey Balandin, and Fabio Viola. Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities: Emerging Research and Opportunities. IGI Global, 2019.
[42] E. Cesario, F. Marozzo, D. Talia, and P. Trunfio. Sma4td: A social media analysis methodology for trajectory discovery in large-scale events. Online Social Networks and Media, 3-4:49–62, 2017.
[43] Sergei Mikhailov and Alexey Kashevnik. Car tourist trajectory prediction based on bidirectional lstm neural network. Electronics, 10(12):1390, 2021.
[44] L. Belcastro, F. Marozzo, D. Talia, and P. Trunfio. Parsoda: high-level parallel programming for social data mining. Social Network Analysis and Mining, 9(1), 2019.
[45] Dimitrios Michael Manias and Abdallah Shami. Making a case for federated learning in the internet of vehicles and intelligent transportation systems. IEEE Network, 35(3):88–94, 2021.
[46] Venkatraman Balasubramanian, Safa Otoum, and Martin Reisslein. Venet: hybrid stacked autoencoder learning for cooperative edge intelligence in iov. IEEE Transactions on Intelligent Transportation Systems, 23(9):16643–16653, 2022.
[47] Xiaokang Zhou, Wei Liang, Jinhua She, Zheng Yan, I Kevin, and Kai Wang. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. IEEE Transactions on Vehicular Technology, 70(6):5308–5317, 2021.
[48] Niloofar Khanghahi and Reza Ravanmehr. Cloud computing performance evaluation: issues and challenges. Comput, 5(1):29–41, 2013.
[49] V Krishna Reddy, B Thirumala Rao, and LSS Reddy. Research issues in cloud computing. Global Journal of Computer Science and Technology, 2011.
[50] Daniel Rosendo, Alexandru Costan, Patrick Valduriez, and Gabriel Antoniu. Distributed intelligence on the edge-to-cloud continuum: A systematic literature review. Journal of Parallel and Distributed Computing, 2022.
[51] M. K. Mohan Murthy, H. A. Sanjay, and Jumnal Anand. Threshold based auto scaling of virtual machines in cloud environment. In Ching-Hsien Hsu, Xuanhua Shi, and Valentina Salapura, editors, Network and Parallel Computing, pages 247–256, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
[52] Best practices for Autoscale - Microsoft Azure. https://learn.microsoft.com/en-us/azure/azure-monitor/autoscale/autoscale-best-practices. Accessed March 2023.
[53] Amazon EC2 Auto Scaling FAQs. https://aws.amazon.com/ec2/autoscaling/faqs/. Accessed March 2023.
[54] Taxis, cars and tourists in Rome. https://www.agenzia.roma.it/ and https://www.comune.roma.it/. Accessed March 2023.
[55] Driving patterns in Rome. https://www.tomtom.com/traffic-index/rome-traffic/. Accessed March 2023.
[56] Mianbo Huang, Guoru Zhao, Lei Wang, and Feng Yang. A pervasive simplified method for human movement pattern assessing. In 2010 IEEE 16th International Conference on Parallel and Distributed Systems, pages 625–628, 2010.

LORIS BELCASTRO is a research fellow of computer engineering at the University of Calabria, Italy. He received a Ph.D. in Information and Communication Engineering at the University of Calabria. In 2012 he held a scholarship at the Institute of High-Performance Computing and Networking of the Italian National Research Council (ICAR-CNR). His research interests include cloud computing, social media and Big Data analysis, distributed knowledge discovery, and data mining.

FABRIZIO MAROZZO is an assistant professor of computer engineering at the University of Calabria. He received a Ph.D. in Systems and Computer Engineering at the University of Calabria. In 2011-2012 he visited the Barcelona SuperComputing Center for a research internship with the Grid Computer Research group in Computer Sciences department. He is a member of the editorial boards of several journals including IEEE Access, IEEE Transactions on Big Data, Journal of Big Data and SN Computer Science. His research focuses on big data analysis, social media analysis, parallel and distributed computing, cloud and edge computing, and machine learning.

ALESSIO ORSINO is a PhD student in Information and Communication Technologies at the University of Calabria, Italy. His research interests include big data analysis, parallel and distributed computing, cloud and edge computing, and machine learning.

DOMENICO TALIA is a professor of computer engineering at the University of Calabria and an honorary professor at Noida University. He is a member of the editorial boards of Computer, Future Generation Computer Systems, IEEE Transactions on Parallel and Distributed Systems, ACM Computing Surveys, the Journal of Cloud Computing—Advances, Systems and Applications, the International Journal of Next-Generation Computing. His research interests include parallel and distributed data mining algorithms, cloud computing, machine learning, Big Data, peer-to-peer systems, and parallel programming models.

PAOLO TRUNFIO is an associate professor of computer engineering at the University of Calabria. In 2007 he was a visiting researcher at the Swedish Institute of Computer Science (SICS) in Stockholm. In 2001-2002 he was a research collaborator at the Institute of Systems and Computer Science of the Italian National Research Council (ISI-CNR). He is in the editorial board of Future Generation Computer Systems, IEEE Transactions on Cloud Computing, and Journal of Big Data. His research interests include cloud computing, social media analysis, service-oriented architectures, distributed knowledge discovery, and peer-to-peer systems.