Future Generation Computer Systems 155 (2024) 108–120

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Autonomous proactive data management in support of pervasive edge applications

Kostas Kolomvatsos [a,*], Christos Anagnostopoulos [b]

[a] Department of Informatics and Telecommunications, University of Thessaly, 3rd Km of the Old National Road Lamia-Athens, Lamia, 35100, Greece
[b] School of Computing Science, University of Glasgow, Lilybank Gardens 17, Glasgow, G12 8RZ, UK

ARTICLE INFO

Keywords:
Pervasive edge computing
Internet of Things
Distributed tasks management
Data migration
Data selectivity
Proactive decision making

ABSTRACT

Recently, context-aware data management becomes the focus of many research efforts placed at the intersection between the Internet of Things (IoT) and Edge Computing (EC). Huge volumes of data can be collected by IoT devices being 'connected' with EC environments transferring data towards the Cloud. EC nodes undertake the responsibility of managing the collected data, however, they are characterized by limited storage and computational resources compared to Cloud. Evidently, this makes imperative the introduction of *data selectivity* methods to keep locally *only* the data requested by end users or applications for current and future analytics services. In this paper, we study an EC environment where nodes rely on data selectivity and decide the allocation of newly received data to peers, or Cloud when these data are not conformed with local data filters. Data filters are the means for determining local data selectivity by keeping only data that statistically match the needs of nodes (e.g., match the already present data or requests for processing defined by incoming tasks). We contribute with data selectivity and filtering models that support intelligent decisions on *when* and *where* incoming data should be allocated. We intent to 'postpone' the transfer of data to the Cloud by keeping them close to end users. Our approach concludes a data map of an EC environment nominating every node as the owner of specific data (sub)spaces facilitating the placement of future processing tasks. We evaluate and compare our models and algorithms against schemes found in the literature showcasing their applicability and efficiency in pervasive edge computing environments.

1. Introduction

Pervasive Computing (PC) refers to the intelligent placement of data and devices armed with embedded processors around end users to support various types of applications. The current transformation of PC is the combination of two infrastructures: Internet of Things (IoT) and Edge Computing (EC). IoT devices are located very close to end users, thus, their connection with the EC ecosystem results in the Pervasive Edge Computing (PEC) paradigm. Numerous devices are active in these ecosystems facilitating the collection of data and the execution of tasks. Any type of application is supported by a set of services present either on IoT devices or EC nodes. Nodes act as the mediators between the IoT ecosystem and the Cloud having the opportunity to host data and execute e.g., processing analytics and predictive tasks. Nodes are also located close to end users and IoT devices [1] and exhibit computational capabilities that give them the ability to process data and follow a behaviour towards the maximization of the performance. Nodes should be armed with intelligence to adapt their behaviour and be aligned with the dynamics of the environment and dynamic changes in applications' requests.

Apart from running services and executing processing tasks, EC nodes, hereinafter called *nodes*, are adopted for distributed data storage by incorporating all the necessary semantics for data update and access [2,3]. As nodes have less capabilities than Cloud datacenters, they should intelligently decide their line of actions related to: **(a)** what services or tasks should be executed locally; **(b)** what data should be kept locally to support the requested and/or future processing tasks. One of the reasons is that nodes cannot host huge volumes of collected data captured by IoT devices. However, the collected data should remain available as much as possible at the EC infrastructure to reduce the latency that will be faced when processing activities rely on Cloud premises. Recent large scale experimentation activities adopting real settings around the Globe show that **(a)** 58% of end users can reach a close edge server in less than 10 ms while only 29% of end users can enjoy a similar latency from a nearby Cloud location; **(b)** compared to Cloud, edge servers offer lower latency to 92% of end users [4]. These observations become more intense if we focus on areas where Cloud datacenters are 'rare' (e.g., in Africa). Hence, it is more beneficial

* Corresponding author.
E-mail address: kostasks@uth.gr (K. Kolomvatsos).

https://doi.org/10.1016/j.future.2024.02.003
Received 10 October 2023; Received in revised form 3 January 2024; Accepted 6 February 2024
Available online 7 February 2024

to keep data to neighbouring EC nodes (which are located in a close distance compared to the Cloud) maintaining a map of the available datasets instead of sending data and requests/tasks to the Cloud and wait for final outcomes.

In any case, even if a huge amount of resources could be deployed at the edge to evolve into a large-scale distributed computing environment, still, it would not have been an appropriate place for processing huge volumes of data [1]. Therefore, decisions that lead to efficient management of the available resources and incoming processing tasks are related to *which* data should be kept locally at the nodes. Apparently, if incoming data do not 'fit' in the local resources, we can detect the opportunity to have them migrated at peer nodes before it is decided to be transferred to Cloud. Such kind of behaviour lies at the core of the PEC targets, i.e., to deliver on-device/on-node applications and support collaborative models for distributed data management and tasks processing. The described scenario will be useful when we aim to support real-time data processing, which is the main challenge when running applications that interact directly with end users. We consider that the decision making related to the selection of data that will be kept locally consists of a *data selectivity problem*. This requires the introduction of a *data filter* being responsible to be statistically matched against the data. The matching process will result if these data will be locally stored or rejected and allocated to peers otherwise. However, if a new task requests data for accessing and processing, while those data are absent, nodes can adopt two strategies: **(i)** task offloading or **(ii)** migrating the appropriate services/data locally [5]. Either the offloading decision or data migration requires to rely on peers that are in a close distance to enjoy the limited possible latency.

Motivating Example. We focus on a scenario where a set of EC nodes are placed in various locations in a smart city. EC nodes are capable of storing multidimensional data and executing a set of Machine Learning (ML) models or any other processing activity (e.g., missing value imputation, outliers detection). For instance, the discussed data can be the recordings of environmental parameters and urban pollution indicators in a city, e.g., temperature, humidity, ozone, particulate matter (PM), CO2, SO2, and NOx. The collected data are maintained in dedicated data structures and may be replicated/transferred to the Cloud where a central repository of all the available data is available. Processing activities are requested in the form of tasks, e.g., estimation of certain statistics for some or all the data dimensions, execution of regression models for estimating future realizations of the observed features/variables, context based classification of the current situation in a specific part of the city, etc. Nodes rely on a data selectivity method, i.e., a filter that is delivered upon *the incoming requests for processing* to reject newly received data that do not match with the current patterns, thus, they save local resources. For instance, local council authorities may demand from nodes located at the core area of a suburban area to return the pollution levels for a set of environmental monitoring parameters when the observations are over a pre-defined threshold. This way, a dedicated application will be capable of triggering alerts and initiate potential mitigation actions or urban planning (e.g., installations of roadside vegetation barriers). The data (probably those below the pre-defined threshold) can be maintained in close neighbouring nodes located in the same suburb to be part of other processing activities like the provision of environmental data and their estimations to citizens in real-time maps. This way, nodes become 'experts' on data with specific statistical characteristics and efficient in terms of storage being fully adjusted to applications' needs. In this case, where every model comes from a specific dataset, it will be profitable to keep any rejected data assets in the EC ecosystem and migrate them to the most appropriate peer(s) in terms of the statistical patterns and characteristics of the owned dataset. In the example of the environmental urban monitoring, data can be selectively stored to peers in close distance with the node(s) rejecting them in order to reduce the latency when access on them is required. Hence, the rejected data in a node may become useful data in another node being part of the training process under the assumption that nodes are close.

In the above motivating scenario, we identify the need for selectively sharing the locally rejected data when they do not match the local data based on a data filter decision. Our proposal to selectively keep those data at the EC infrastructure as much as possible targets to minimize the latency in the provision of responses and maximize the reusability of the data for current and future analytics tasks. When keeping rejected data in neighbouring node(s), specifically in peers that possess statistically similar data, we evidently enjoy lower latency compared to relying on the Cloud as already noted above. When access to the relocated data is requested by a specific task, then, those data can be accessed either by migrating them back to the original node (if there is available storage capacity) or by offloading the task that demands their processing [6]. In any case, the definition of data filters, which are adopted to decide the discussed rejections, is a very dynamic process that could lead to the necessary updates if continuous requests for processing demand for data that are not participating in the data filters delivered in the previous epochs. Apparently, data filters are concluded at pre-defined intervals in order to keep track of the evolution of the incoming requests.

In this paper, we address the issue of efficient management of locally rejected data and introduce a data selectivity mechanism that combines ML with probabilistic decision making. Distributed data filters are shared across the network edge to guide the decision making upon the optimal relocation of the rejected data to the most suitable peers. Initially, our strategy is to perform processing of data filters locally at every node in light of examining potential matching peers. A clustering process groups the available nodes together based on the statistical similarities of their data filters depicting a matching degree for the most recent past. Additionally, we introduce a probabilistic model being responsible to expose the expected difference between pairs of filters upon historical values trying to assess data filter similarity in a window of observations. Our technical contributions are:

- A machine learning model trained over shared data filters across distributed nodes;
- A probabilistic model to detect the statistical matching between shared data filters within sliding time window;
- An aggregation mechanism merging current view on the filters matching and corresponding historical patterns;
- A decision making mechanism selecting the most suitable peers where locally rejected data are relocated;
- A comprehensive evaluation of our mechanism against approaches found in the literature showcasing the benefits and its applicability in PEC environments.

The paper is organized as follows. Section 2 elaborates on related work. We provide preliminary information in Section 3. The data selectivity problem formulations are discussed in Section 4. Section 5 introduces our models and algorithms, while Section 6 reports on the experimental evaluation. Section 7 concludes the paper with our future agenda.

2. Related work

The IoT and the EC infrastructures are smoothly combined to create a layered architecture where data and services can be allocated in any place. IoT devices can monitor a specific phenomenon collecting the corresponding data [7–10] reporting them in streams [11] towards the Cloud back end. Streams processing is adopted when real time decisions should be made while the processing of data at rest is utilized for long term decisions. The outcome of processing activities, requested in the form of tasks, is reported back to the requestors that may be end users or applications. Some examples of tasks are as follows: queries requesting data retrieval upon the local data [12,13], the training of ML models [6,14], data selection for caching [15], video

pre-processing [16]. When nodes have a high load or do not own data requested by the incoming tasks, those tasks could be offloaded to peers or Cloud [6,17]. Other mitigation actions that nodes can adopt to increase the efficiency of the requested processing are the migration of services [5] or data [18,19]. Both activities try to fill gaps in the capabilities of nodes to avoid any malfunctions or delays in the provision of responses. An intelligent behaviour is concluded when all the aforementioned actions are combined with the ultimate goal to keep data and processing as close to end users as possible to minimize the latency. The aggregation of locally possessed data [20], the collaboration between nodes [21] or the definition of a sink node acting as the single representative of a group [8] where data are stored and processing activities take place [22] are some of examples towards reaching a level of collaboration between autonomous entities. No matter the adopted strategy, the discussed problem is complex if we want to have nodes acting independently trying to meet the performance challenges that dynamic and distributed environments impose.

If we focus on the individual behaviour of nodes, we can easily detect the need for the efficient management of the incoming tasks and the available data. Both, tasks offloading and data migration/replication activities refer in solving the same problem, i.e., to effectively serve the requests for processing. Depending on the status of nodes, e.g., the presence of the appropriate data, the current load, etc, we can decide to offload some tasks or to ask for data migration. Apparently, the first choice is to keep tasks and data at the EC ecosystem instead of relying to the Cloud in order to minimize the latency in the provision of responses. The decision for offloading tasks faces various challenges like the communication overhead [23]. The best possible performance can be only achieved by incorporating into the decision making mechanism the processing of the surrounding contextual information. The fundamental awareness of the local load in combination with the data present at every node consists of a crucial knowledge for the selection of *where* tasks could be executed. Towards this direction, we can rely on a 'map' of the distributed data and their owners to 'affect' any offloading activity [18] while a monitoring mechanism of the local status of nodes [24] can trigger those actions. The research community has already proposed various mechanisms like cooperative schemes [25], models relying on swarm intelligence technologies [26,27], genetic algorithms [28], graph-based schemes and their intelligent management [29] or the optimal stopping theory [30]. In general, any of the proposed models should take into consideration the combination between the local processing load and the demand for every task before it is in a position to deliver the final action [6,31]. Nodes would benefit if they were aware of the distribution, trends, and access patterns of the incoming tasks over their local data [14,32].

Data migration and replication can be adopted to efficiently use the computing resources available at the edge of the network by providing the necessary basis for any processing activity. Latency can be avoided if we place the appropriate data at the appropriate nodes in order to speed up the processing and eliminate the overhead of data transfer when this processing should be executed. Obviously, data migration and/or replication should be performed, if it is possible, in a proactive manner before a processing is requested [33]. Replication is the ideal methodology to handle scenarios where data consumers are located in different geographical areas subscribe to the same data, while only one replica of these data exists in the proper node [34]. Data can be transferred either in a push or in a pull model depending on the node that will initiate the activity (requestor or provider); otherwise, it can be concluded by a central entity that undertakes the responsibility to place the data at the appropriate locations in the ecosystem. Edge to edge learning could provide solutions to the problem of data migration as it targets to optimize the organization of the edge ecosystem [35]. In [36], the authors focus on an optimization algorithm that is responsible to eliminate the delay in data transmission and computation time. Such an approach is beneficial when we try to serve applications targeting to transfer data in the network. The authors of [37] propose a method for data collection in support of privacy upon the smart grid infrastructure. In [38], a secure data channel resisting injection attacks is suggested. Data are continuously transmitted and replicated to the available nodes to support the minimum possible knowledge for decision making. The authors of [39] discuss a high-security model for transferring large scale data to the Cloud back end as well as an inter-cloud knowledge migration scheme. The main observed parameter is communication overhead as it plays a significant role in the transfer of data. In [40], the authors propose an opportunistic mechanism for data offloading based on a random strategy for selecting the nodes from the population where the offloaded data will be allocated. In [41], it is pointed out that data migration could be part of virtualized resources migration. The decision is based on parameters like the time required for the migration activity, the downtime, application quantity and application degradation.

Data selectivity or data filtering is adopted to select the incoming data that will be hoisted locally in a node. Any selectivity/filtering model should be designed to detect the dependencies between nodes, their datasets and incoming tasks [42,43]. Actually, the processing of the constraints that the incoming tasks impose is a useful knowledge to identify and conclude the appropriate local data filter. For instance, DTFilter [44] is proposed to evaluate the incoming tasks based on a searching balanced-tree algorithm avoiding the transmission of duplicate data assets in data streams. Another attempt for data selectivity/filtering is based on data labelling and policy rules expressions to define the conditions that should be satisfied before any data assets is selected to be hosted locally [45]. In [46], the authors present the adoption of Bloom filters to exploit the partitioning of the collected data in light of addressing event streams using sliding windows. The authors of [47] propose a scheme that adopts an ML model for detecting outliers. It is a type of selectivity/filtering method that tries to detect data that may negatively affect the statistics of the dataset and the training of machine learning models. In [48], the authors present a communication standard designed to support generic data acquisition, filtering and buffering protocols between Cloud and EC infrastructures. In [49], the authors discuss a method adopted at the EC ecosystem being responsible to utilize existing ML models to classify a suspicious file in a set of pre-defined categories. The authors of [50] present an adaptive data processing method for EC nodes sensing in ubiquitous NB-IoT. The proposed model process the NB-IoT data to have practical engineering application value. Finally, in [51], a sampling methodology is discussed. The proposed approach relies on the latest data collected to be the subject of processing of a linear fitting and adjusts the next sampling frequency according to the linear median jitter sum and an adaptive sampling strategy.

In this paper, we depart from the state of the art as it is above presented and propose a different mechanism for data selectivity/filtering. Actually, we do not focus on the conclusion of a data filter but we propose a model for allocating the locally rejected data to the appropriate node in the EC ecosystem. Our intention is to support the nodes with a model that tries to keep IoT data at the edge of the network as much as possible. The proposed approach differs from a data migration model in the sense that our mechanism decides the migration of data only when those data do not match to the local dataset. This way, if compared with the relevant literature, the envisioned methodology is motivated by the need of enhancing the efficient management of the incoming data and the 'coordination' of data offloading activities which becomes part of the behaviour of every EC node. Our strategy is built upon the knowledge that local data filters are formulated after the processing of the incoming tasks imposing requests about the data assets that should participate in the demanded activities. We reveal the potentials of the proposed approach focusing on context-aware data allocation processing and defining a novel 'map' over the distributed nodes' datasets capable of supporting decision making and serve any defined task. Our model results the migration of data that will power the distributed datasets by incorporating additional information pieces with great benefits for the potential training of local ML models.

3. Problem fundamentals

3.1. Rationale

We report on the problem fundamentals and preliminaries. Table 1 shows the notations adopted in this paper.

Consider a PEC environment with N distributed nodes from a set $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$. Nodes are directly connected in the EC infrastructure as well as with the IoT devices to receive and host data. In this combined ecosystem (IoT and EC), horizontal (collaborative activities between nodes) or vertical (collaborative activities between IoT devices and nodes) services can access the collected data. Our target is to keep data selection and processing close to end users while avoiding inherent latency when relying to the Cloud. Evidently, nodes' computational capabilities cannot be compared with Cloud's processing power, thus, intelligent schemes are required for efficient management of data and tasks at the edge. One of the most significant decisions that nodes should take is the *selection of data kept locally* due to limited storage resources. The *remaining data* can be either transferred to Cloud or be kept at the edge in peer nodes.

Definition 1. Data selectivity refers to the action of choosing the data assets that could be kept and stored locally to a node for current and future processing.

Data selectivity targets to support fast processing outcomes, low latency and reduced bandwidth consumption [52]. Apparently, data selectivity should be affected by tasks accessing and executed over local datasets. For instance, tasks related to finding the top-k item in data streams [53], training and sharing reusable models [54], and offloading predictive analytics [5,6].

Definition 2. A task (or task request) is a data processing activity that should be performed upon the currently available data satisfying specific data domain constraints (range intervals).

The on-node processing of tasks taking into account their conditions and data domain constraints, requires the definition of a *filtering mechanism* aligned with the demand for data. Such mechanism should timely decide on rejecting data that are not requested locally (currently and in the near future), thus, saving resources. When the decision of transferring the rejected data to the Cloud is 'postponed', thus, having them shared to peers owing *similar* datasets, yields in enhancing nodes' local datasets with new data. Therefore, data sharing should be governed by fundamental rules concerning **(i)** the similarity between the new and local data, **(ii)** the ability of the new hosting nodes to store additional the data, and **(iii)** the expected communication overhead. The anticipated benefits can be obtained considering that nodes are armed with knowledge about where every data asset is located in the edge. In turn, any potential offloading of tasks, which cannot be served locally, will be performed efficiently as well. Our approach is based on the capacity of nodes to incrementally and dynamically build individual data filters upon incoming tasks requests.

Definition 3. A data filter is represented via a set of data domain constraints/range intervals per data dimension such that the currently available data should satisfy.

Any data asset lying *outwith* the filter is allocated either to a subset of peer nodes or Cloud. This decision is made in a sequential manner. When incoming data asset violates the local filter constraints, then, nodes **(i)** check whether the data asset can be migrated to peers having similar datasets; or **(ii)** to Cloud in the absence of any suitable peer.

3.2. Task & filter definitions

Consider the local dataset on node $n_i \in \mathcal{N}$ as a collection D_i of M-dimensional vectors $D_i = \{\mathbf{x}_v\}_{v=1}^{|D_i|}$ with $\mathbf{x} = [x_1, x_2, \ldots, x_M] \in \mathbb{R}^M$. Analytics tasks are represented by a series of data access requests over node's data for e.g., regression and classification. We, therefore, define a task request as

$$\mathcal{T} = \left\{ \left[y_1^l, y_1^h\right], \ldots, \left[y_M^l, y_M^h\right] \right\}, \tag{1}$$

where for $j = \{1, 2, \ldots, M\}$, y_j^l and y_j^h represent the lowest and highest range values for the jth dimension, such that $y_j^l \leq y_j^h$. For instance, the task request $\mathcal{T} = [15, 28]$, with $y_j^l = 15$, $y_j^h = 28$ demands access and processing over data values in the range $[15, 28]$. Tasks are received in streams introducing a set of range values per request. The node's local data filter aggregates all range intervals for each jth dimension. Let us denote the filter for the jth dimension with $\left[\widetilde{y}_j^l, \widetilde{y}_j^h\right]$, where the lowest $\widetilde{y}_j^l$ and the highest $\widetilde{y}_j^h$ interval boundaries correspond to the lowest and the highest values or the average lowest and highest interval boundaries of all the task requests, respectively. Note that nodes consider W incoming requests before the final data filter $\left[\widetilde{y}_j^l, \widetilde{y}_j^h\right]$ is applied. On the other hand, each incoming data $\mathbf{x}$ is matched against $\left[\widetilde{y}_j^l, \widetilde{y}_j^h\right]$, which provides key information to the node on deciding whether to keep locally $\mathbf{x}$ or not. Data filters can be the result of a simple or more complex processing, e.g., they could be the minimum and maximum values of requests or the aggregation of the received values for each of the two lists $\widetilde{y}_j^l \& \widetilde{y}_j^h$. This processing lies beyond the subject of the paper being left for future work. After the processing of all filters for every dimension, nodes devise a data selectivity methodology based on a set of range intervals:

$$\widetilde{\mathcal{T}}_i = \left\{ \left[\widetilde{y}_j^l, \widetilde{y}_j^h\right] \right\}, j = 1, 2, \ldots, M, \tag{2}$$

The $\widetilde{\mathcal{T}}_i$ in Eq. (2) is distributed in the PEC at pre-defined dissemination epochs to keep nodes aware about the filtering activities in their peers. In the time between two reporting epochs, nodes do not have full information on the actual boundaries of their peers' data filters. There is a trade off between the frequent dissemination of filters with the increased possibility of burdening the network. The management of the aforementioned uncertainty and the discussed trade off lies at the first places of our future research directions.

Consider the ability of n_i to discern the suitable peers where locally rejected data could be allocated to remain as much as possible at the EC infrastructure. After every reporting epoch where data filters are distributed in the network, n_i updates its local cache/data structure storing its neighbouring peers' filters. In Eq. (3), the $\mathbf{M}_{N \times M}$ matrix presents an example of such a cache/data structure that stores N data filters (the local one and the $N-1$ filters of peers). Note, each filter has M boundary pairs due to the dimensionality of the collected data. When a data asset $\mathbf{x}$ arrives at n_i, the node matches it against the local aggregated filter $\widetilde{\mathcal{T}}$. This provides information about the suitable peer nodes for potentially hosting $\mathbf{x}$ subject to a decision of the local rejection. An overview of the actions that n_i takes is:

- **Action 1:** If $\mathbf{x}$ matches $\widetilde{\mathcal{T}}_i$, then, $\mathbf{x}$ is stored in the local dataset at node n_i;
- **Action 2:** If $\mathbf{x}$ does not match $\widetilde{\mathcal{T}}_i$, then n_i selects the best possible peer to send over $\mathbf{x}$. This is obtained given the data filters in $\widetilde{\mathcal{T}}$, as will be elaborated later;
- **Action 3:** If $\mathbf{x}$ does not match $\widetilde{\mathcal{T}}_i$ and there is no appropriate peer to host $\mathbf{x}$, then $\mathbf{x}$ is transferred to the Cloud.

Every action is taken subject to the rejection of previous actions in the following order: Action 1, Action 2, and Action 3. This sequential decision making postpones the transfer of $\mathbf{x}$ to the Cloud. The fundamental challenge is to obtain the list of the most appropriate peers capable of hosting $\mathbf{x}$ based on the data filters aggregated in $\widetilde{\mathcal{T}}$. To deal with this

Table 1
Nomenclature.

Notation	Description
N	Number of nodes
$\mathcal{N} = \{n_1, \ldots, n_N\}$	Set of nodes
n_i	The ith node
$\mathbf{x}$	Multivariate data vector
$\mathbf{M}_{N\times M}$	Cache of distributed filters
M	Dimensionality of data vectors
$\mathcal{T}$	The task request for processing
$\left[y_j^l, y_j^h\right]$	Requested interval for processing
$\widetilde{y}_j^l$	The lowest boundary of the final filter
$\widetilde{y}_j^h$	The highest boundary of the final filter
$\widetilde{\mathcal{T}}_i$	The final local data filter for the ith node

challenge and elaborate on a solution, we contribute with a probabilistic quantization-based model over the difference between data filters. The aim is to aggregate the available information in between reporting epochs with historical data filters in light of deciding *where* the locally rejected data should be hosted at the EC infrastructure.

$$\mathbf{M} = \begin{bmatrix} [y_{11}^l, y_{11}^h] & [y_{12}^l, y_{12}^h] & \ldots & [y_{1M}^l, y_{1M}^h] \\ [y_{21}^l, y_{21}^h] & [y_{22}^l, y_{22}^h] & \ldots & [y_{2M}^l, y_{2M}^h] \\ \ldots & \ldots & \ldots & \ldots \\ [y_{N1}^l, y_{N1}^h] & [y_{N2}^l, y_{N2}^h] & \ldots & [y_{NM}^l, y_{NM}^h] \end{bmatrix} \tag{3}$$

4. Data selectivity & filters processing

This section introduces the filter management and the development of an ML model being responsible to facilitate the detection of similar peers in order to support the migration of data when necessary.

4.1. Data filters management

All the received data filters are maintained into a local data structure for processing. The target is to 'group' peers based on their data filters, then, to detect where $\mathbf{x}$ can be allocated. Since $\mathbf{x}$ is a multidimensional vector, any similarity with the processed data filters should be delivered across M 'individual' similarities.

Initially, we have to identify the sub-set of nodes exhibiting similar data filters for every dimension. Those nodes become a first target group for sharing $\mathbf{x}$. Without loss of generality, in the remaining description, we focus on the jth dimension. The same approach stands true for all the adopted dimensions. We cluster the data filters for the jth dimension (jth column of the matrix $\mathbf{M}$). The outcome is $|C|$ clusters (without loss of generality, we consider the same number of clusters for all dimensions). Fig. 1 presents an example of the proposed model. We could rely on any clustering algorithm, e.g., in a distance based approach where all data filters present in a cluster exhibit the lowest possible intra-cluster and the maximum possible inter-cluster distances. Clustering is a widely adopted ML methodology that does not require any training process and delivers the final clustering outcome in a limited amount of time. We have to notice that the number of clusters are pre-defined (see the experimental evaluation of our model) and depict the best possible value for reaching high quality clusters. However, we can also rely on a clustering method that does not require the number of clusters beforehand (it is a matter of implementation and does not affect the proposed model) increasing the autonomous nature of the proposed scheme. The clustering process obtains a set of clusters $C = \left\{C_1^j, C_2^j, \ldots, C_{|C|}^j\right\}_{j=1}^M$ with centroids $\left\{c_1^j, c_2^j, \ldots, c_{|C|}^j\right\}_{j=1}^M$ such that the following equation holds:

$$c_m^j = \frac{1}{|C_m^j|} \sum_{i=1}^{|C_m^j|} \| \left[\widetilde{y}_{ij}^l, \widetilde{y}_{ij}^h\right] - \left[\widetilde{y}_{i'j}^l, \widetilde{y}_{i'j}^h\right] \|, \forall \left(i, i'\right), i \neq i'. \tag{4}$$

Apart from the aforementioned clusters, all nodes maintain historical values related to the reported data filters in a window $\hat{W}$. The aim is to use such information to combine it with the knowledge extracted by the delivered clusters. The proposed model (presented later) aggregates the matching between nodes based on the most recent data filters with the historical patterns identified upon the historical reports that depict the continuous connection between the different data filters. The ultimate goal is to efficiently support the matching between nodes not only on the current but also on past views.

4.2. Nodes matching indicator

After the clustering process, n_i decides to offload the rejected data based on the M clusters (one per dimension). Let us denote the set of clusters where n_i is present with $S = \left\{S_1, S_2, \ldots, S_M\right\}$, $S_j \in C$. Apparently, n_i 'matches' to a sub-set of peers for each dimension. The target is to determine the final sub-set of similar peers for all dimensions. We achieve this focusing on the intersection of different sets $S_j, j = 1, 2, \ldots, M$, i.e., $\widetilde{S} = S_1 \cap S_2 \cap \ldots \cap S_M$. Let n_z be a node being member of $\widetilde{S}$ and exhibits a high similarity with n_i across all the envisioned dimensions. The discussed approach consists of a 'strict' model that tries to identify nodes being in the same clusters for all dimensions. We can extend this list if no node is similar to n_i for all the dimensions and define a threshold over which the delivered similarity is accepted. For instance, we can incorporate in $\widetilde{S}$, nodes that are present at the same cluster with n_i for at least θ dimensions.

We can adopt the L_1 norm for calculating the difference (a.k.a. Manhattan distance) between the filters of two nodes, e.g., n_i and n_z. This means that the difference for the jth dimension is defined by $d_{iz}^j = |\widetilde{y}_{ij}^l - \widetilde{y}_{zj}^l| + |\widetilde{y}_{ij}^h - \widetilde{y}_{zj}^h|$. Apparently, $\max\left(d_{iz}^j\right)$ refers to the diameter of the corresponding cluster. Upon this distance, we define the *Nodes Matching Indicator* (NMI) as the heuristic:

$$\phi_{iz}^j = \alpha \cdot \left(\frac{1}{1 + e^{\gamma d_{iz}^j + \delta}}\right) \tag{5}$$

with

$$\alpha = \begin{cases} \omega \in [1, \infty] & \text{if } n_i, n_z \in S_j, \\ \hat{\omega} \in (0, 1) & \text{otherwise,} \end{cases} \tag{6}$$

and $\gamma \in \mathbb{R}^+$, $\delta \in \mathbb{R}^-$. NMI depicts the distance between n_i and n_z as a portion of a threshold defined by parameters γ and δ. When $n_i \& n_z$ are not in the same cluster, the use of α reduces the final similarity result. If $n_i \& n_z$ are distant, indicating a high difference in their data filters (even if they are placed in the same cluster), ϕ_{iz}^j becomes low (close to zero). When $n_i \& n_z$ have a low difference in their data filters, ϕ_{iz}^j becomes high (close to unity). The holistic NMI between nodes n_i and n_z is obtained as the aggregated indicators of all dimensions:

$$\phi_{iz} = \sum_{j=1}^{M} \phi_{iz}^j. \tag{7}$$

NMI aggregates the current difference of the data filters of the two nodes for all dimensions. This knowledge is integrated with the historical information on the filters' difference, which will be used to identify the list of suitable peers where $\mathbf{x}$ could be migrated/replicated.

5. Suitable peers selection

In this section, we elaborate on the process adopted to select the final subset of suitable peers to decide upon when $\mathbf{x}$ deviates from the local data filter. The final decision is based on the knowledge of the difference between the current data filters as well as historical data filters information. The aim is to select peers that are historically similar with the node rejecting $\mathbf{x}$ avoiding to be affected by any seasonal trends.

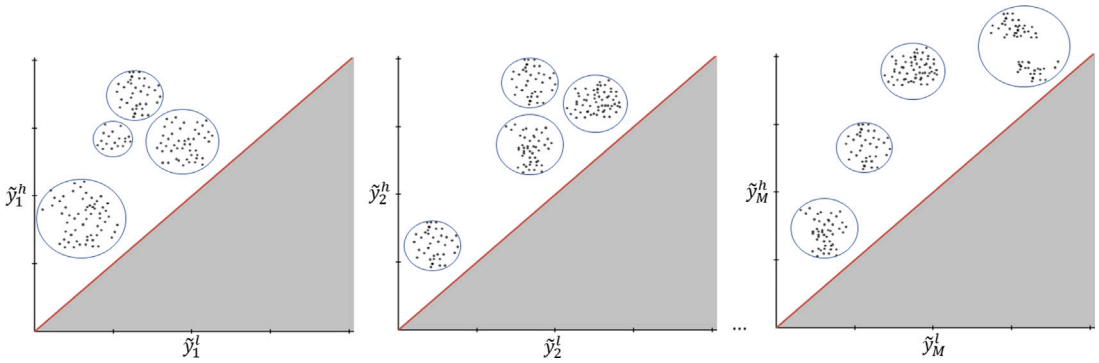


Fig. 1. An example of the envisioned clustering process for every dimension.

5.1. Statistically oriented peers selection

Consider the following random variables: Y_j^l representing the lower boundary of a data filter for the jth dimension with realization y_j^l, and Y_j^h representing the higher boundary of a data filter for the jth dimension with realization y_j^h. Based on this assumption, we consequently define the following random variables ${}_j\Omega_{iz}^l = |Y_{ij}^l - Y_{zj}^l|$ and ${}_j\Omega_j^h = |Y_{ij}^h - Y_{zj}^h|$ depicting the difference of the boundaries for the jth dimension. Apparently, $\Omega_{iz} = \sum_{j=1}^M \left({}_j\Omega_{iz}^l + {}_j\Omega_{iz}^h\right)$ depicts the total difference for a pair of data filters for nodes n_i and n_z. By maintaining the historical differences of the data filters adopted by the two nodes, we calculate the expected difference over time and combine it with the current difference as indicated by ϕ_{iz}.

The linearity of the total difference leads to the linearity of the sum of expected values under the assumption that filters boundaries are independently selected. Hence, we need to estimate the expectation $\overline{\phi}_{iz} = \mathbb{E}\left(\Omega_{iz}\right)$ and involve the summation of expected values for every boundary of the two data filters, i.e., $\overline{\phi}_{iz} = \sum_{j=1}^M \left(\mathbb{E}\left({}_j\Omega_j^l\right) + \mathbb{E}\left({}_j\Omega_j^h\right)\right)$. We can observe that the expected value of the difference between filters is affected by the distribution of the lower and the higher boundaries as it is expected.

Lemma 5.1. *If* $Y_{ij}^l \sim N\left(\mu_{ij}^l, {}_l\sigma_{ij}^2\right)$ *and* $Y_{zj}^l \sim N\left(\mu_{zj}^l, {}_l\sigma_{zj}^2\right)$*, then, we obtain that* $\mathbb{E}\left({}_j\Omega_j^l\right) = {}_l\sigma_{iz}^j\sqrt{\frac{2}{\pi}} \cdot e^{\left(-\frac{({}_l\mu_{iz}^j)^2}{2\cdot({}_l\sigma_{iz}^j)^2}\right)} + {}_l\mu_{iz}^j \cdot \left(1 - 2\Phi\left(-\frac{{}_l\mu_{iz}^j}{{}_l\sigma_{iz}^j}\right)\right)$ *with* ${}_l\mu_{iz}^j = \mu_{ij}^l - \mu_{zj}^l$ *and* ${}_l\sigma_{iz}^j = \sigma_{ij}^l + \sigma_{zj}^l$.

Proof. Recall that ${}_j\Omega_j^l = |Y_{ij}^l - Y_{zj}^l|$ and Y_{ij}^l, Y_{zj}^l are two normally distributed random variables with means μ_{ij}^l, μ_{zj}^l, and standard deviations ${}_l\sigma_{ij}, {}_l\sigma_{zj}$. As it is already proved [55], the difference of the two normally distributed random variables is also a Normal with mean $\mu_{ij}^l - \mu_{zj}^l$ and variance ${}_l\sigma_{ij}^2 + {}_l\sigma_{zj}^2$. Finally, for getting the expectation of ${}_j\Omega_j^l$, we have to observe that ${}_j\Omega_j^l$ follows a folded Normal distribution and we can find the probability distribution function (pdf) by adding together the pdf of both x and $-x$. We then obtain the expected value of the folded Normal distribution, i.e., $\mathbb{E}\left({}_j\Omega_j^l\right) = {}_l\sigma_{iz}^j\sqrt{\frac{2}{\pi}} \cdot e^{\left(-\frac{({}_l\mu_{iz}^j)^2}{2\cdot({}_l\sigma_{iz}^j)^2}\right)} + {}_l\mu_{iz}^j \cdot \left(1 - 2\Phi\left(-\frac{{}_l\mu_{iz}^j}{{}_l\sigma_{iz}^j}\right)\right)$ where ${}_l\mu_{iz}^j = \mu_{ij}^l - \mu_{zj}^l$ and ${}_l\sigma_{iz}^j = \sigma_{ij}^l + \sigma_{zj}^l$. □

Lemma 5.2. *If* $Y_{ij}^h \sim N\left(\mu_{ij}^h, {}_h\sigma_{ij}^2\right)$ *and* $Y_{zj}^h \sim N\left(\mu_{zj}^h, {}_h\sigma_{zj}^2\right)$*,* $\mathbb{E}\left({}_j\Omega_j^h\right) = {}_h\sigma_{iz}^j\sqrt{\frac{2}{\pi}} \cdot e^{\left(-\frac{({}_h\mu_{iz}^j)^2}{2\cdot({}_h\sigma_{iz}^j)^2}\right)} + {}_h\mu_{iz}^j \cdot \left(1 - 2\Phi\left(-\frac{{}_h\mu_{iz}^j}{{}_h\sigma_{iz}^j}\right)\right)$ *where* ${}_h\mu_{iz}^j = \mu_{ij}^h - \mu_{zj}^h$ *and* ${}_h\sigma_{iz}^j = \sigma_{ij}^h + \sigma_{zj}^h$.

Proof. It follows from Lemma 1. □

It should be noticed that Lemmas 1 and 2 can be easily extended to incorporate additional distributions for the random variables depicting the lower and higher boundaries of the data filters.

5.2. Decision making & data allocation

The final decision making is related to the selection of peers where $\mathbf{x}$ can be migrated, if the it is locally rejected. The basis for the proposed decision making is the two NMIs, i.e., ϕ_{iz} and $\overline{\phi}_{iz}$ calculated for all nodes $n_z \in \widetilde{S}$. Recall that ϕ_{iz} depicts the matching of the two filters based only on the most recent reporting epoch while $\overline{\phi}_{iz}$ is calculated over the historical data filters reports trying to identify a long-term similarity between nodes. Before we 'merge' the two NMIs, we normalize $\overline{\phi}_{iz}$ as it represents the expected difference of the two filters. We strategically select a simple and fast heuristic based on a Sigmoid function that eliminates the final outcome (it gets a value close to zero), if it is over a threshold. Hence, we obtain:

$$\widetilde{\phi}_{iz} = \frac{1}{1 + e^{\overline{\gamma}\overline{\phi}_{iz} + \overline{\delta}}}. \tag{8}$$

By selecting the values of $\overline{\gamma} \in \mathbb{R}^+$ and $\overline{\delta} \in \mathbb{R}^-$, we utilize a strategy for the threshold over which we consider that the expected difference is high enough to eliminate the final outcome of the proposed function. The aggregation of two NMIs is, then, adopted by the geometric mean:

$$f\left(\phi_{iz}, \widetilde{\phi}_{iz}\right) = \left(\phi_{iz}\right)^{\xi}\left(\widetilde{\phi}_{iz}\right)^{\xi'}, \tag{9}$$

where ξ and ξ' are the weights for every NMI. The geometric mean is not heavily affected by fluctuations and extreme values of the processed variables and is suitable to develop a mathematical strategy upon the observations. By carefully selecting ξ and ξ', we can pay more attention on the current filters difference or their historical patterns in order to safely decide if $\mathbf{x}$ could be allocated to node n_z. After the estimation of f for every pair of nodes, we weight the final result by incorporating in the final outcome the distance of $\mathbf{x}$ from the centres of the filters, i.e.,

$$\hat{f}\left(\phi_{iz}, \widetilde{\phi}_{iz}\right) = \frac{\hat{\alpha}}{1 + e^{\hat{\gamma}\hat{d} + \hat{\delta}}} \cdot f\left(\phi_{iz}, \widetilde{\phi}_{iz}\right), \tag{10}$$

where $\hat{\alpha}, \hat{\gamma} \in \mathbb{R}^+$, $\hat{\delta} \in \mathbb{R}^-$ and $\hat{d} = \sum_{j=1}^M |x_j - \left(\frac{1}{2}(y_{zj}^h + y_{zj}^l)\right)|$. The final outcomes are sorted in a descending order of $\hat{f}$ and the top-k peer nodes are those that can host $\mathbf{x}$.

6. Experimental evaluation

We experiment with various scenarios where we apply specific values for the adopted parameters and observe the performance of the proposed model. In this section, we present the datasets that feed our experimentation activities, the performance metrics and the realization of the above parameters. We have to notice that we rely

on a custom simulator written in Python where, through the use of dedicated classes, we simulate the available EC nodes, the arrival of new data, the delivery of local data filters for a set of nodes and the envisioned processing through the adoption of the appropriate data structures. We have to notice that, our simulator gives the opportunity to create a number of nodes and their local datasets in order to support a very dynamic experimentation setup. We do not consider a single powerful node that administrates the proposed model as a central point of decision making. In every node, we emulate the reception of new data assets and update/conclude the corresponding data filters at pre-defined epochs. This simulator is running on an Intel i7 processor with 16 GB RAM.

6.1. Performance metrics & setup

It is a strategic decision to rely on real datasets that carry data observed in a specific setting. We adopt: **(a)** the India Air Quality dataset[1] (dataset DS1) which is a combined data repository across years and states and a clean version of the historical daily ambient air quality data provided by the Ministry of Environment and Forests and Central Pollution Control Board of India under the National Data Sharing and Accessibility Policy (NDSAP). In this dataset, there are 435,742 vectors of real values for 13 dimensions. We pre-process that dataset and eliminate missing values and borrow two dimensions ($M = 2$), i.e., the measurements of Sulphur dioxide (SO2) and Nitrogen dioxide (NO2). From the available values, we get the minimum SO2 to be 9 and the maximum value is 1361 and for the NO2 the minimum and maximum are 26 and 2652, respectively; (b) the GNFUV dataset[2] (dataset DS2) that comprises 2 x (4) sets of mobile sensor readings data (temperature, humidity) from four (4) Raspberry Pi's corresponding to a swarm of four (4) Unmanned Surface Vehicles (USVs). The swarm of the USVs is moving according to a GPS pre-defined trajectory floating over the sea surface in a coastal area of Athens (Greece). We adopt the temperature and humidity readings with their values being observed below 100 to formulate a dataset with different statistics than the former one (i.e., the India Air Quality dataset). We consider N nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ and randomly select a number of tuples to be part of the local datasets $D_i, i = 1, 2, \ldots, N$. At pre-defined intervals (i.e., every 10 iterations), we add new tuples in the datasets (to update their statistics) and estimate the new data filters. Recall that we build the local filters upon the last W incoming requests and maintain the $\hat{W}$ recent filters from those produced locally in the nodes. We consider that for every filter, $\tilde{y}_j^l$ and $\tilde{y}_j^h$ are defined as the minimum and maximum values of the incoming tasks. The simulation is performed by producing a random vector in a randomly selected node checking if this vector matches to the local data filter. If not, we apply our model and select the top-k peers that could host the new data. We perform a set of iterations and record the results for a number of performance metrics presented below. The final outcomes are depicted by the average of observations in order to harmonize the detected performance as it is affected by the randomly selected measurements.

We evaluate our model introducing the following performance metrics:

- τ refers in the ability of the proposed model to handle real time decision making and expose if our approach can be adopted in an EC setup. τ is defined as the average time (in seconds) required to delivery the allocation of a random vector when it does not match to the local filter, i.e., $\tau = \frac{\sum \tau_{\mathbf{x}}}{|\mathbf{x}|}$ where $\tau_{\mathbf{x}}$ is the time required to decide for the allocation of $\mathbf{x}$ and $|\mathbf{x}|$ is the number of the processed vectors. Apparently, when $\tau \to 0$, the proposed model spends the minimum possible time for the final allocation and can be considered as an appropriate model for real time decision making;
- ϵ depicts the ability of the proposed model to detect the best possible node to allocate the incoming vectors by checking if the allocated data assets matches to the filter of the receptor. ϵ is defined as: $\epsilon = \frac{J}{E}$ with J being the number of experiments where the allocated vector matches to the filter of the first node in the top-k list for the entire set of the dimensions, and E is the number of the experiments. We have to notice that the matching is detected for all the considered dimensions (a conjunctive scheme). The greater the ϵ is (close to unity) the better the performance of the model becomes. The proposed model reasons about the similarity of the reported filters *before* it decides to allocate the incoming data assets;
- $\hat{\epsilon}$ refers to the matching of the firstly selected node with the incoming vector for at least the 50% of the adopted dimensions (a disjunctive scheme). $\hat{\epsilon}$ represents a more relaxed strategy to support the final decision making. $\hat{\epsilon}$ is similarly defined as the ϵ metric;
- ρ relies on the top-k list of the selected peers and depicts the matching with their local filters. ρ is defined as follows $\rho = \frac{\overline{J}}{k \cdot E}$ where $\overline{J}$ represents the number of experiments where the incoming vector matches to the filters of the top-k nodes. ρ is depicted by the percentage of nodes present in the top-k list that own a filter matching to the allocated vector. Naturally, a high value of ρ (close to unity) represents the best possible performance for the proposed model;
- $\hat{\rho}$ depicts the outcome of a disjunctive strategy, i.e., we detect if the incoming vector matches to at least the 50% of the adopted dimensions in the top-k nodes. $\hat{\rho}$ is similarly defined as ρ.

We compare our model, coined Filters Matching Model (FMM) with the model presented in [40], coined Random Model (RM) and the scheme proposed by [56] coined Distance Based Allocation Model (DBAM). The RM randomly selects the peers from the population where the offloaded $\mathbf{x}$ will be allocated. We consider that RM relies on k peers by selecting them in a sequential order. The DBAM focuses on the distance between nodes and allocates the data to closest peer (the data allocation involves only one peer) targeting to depict the locality of the observations. The popularity of data referred in the corresponding paper [56] does not match to the needs of our model and is omitted in the comparative assessment. In our experimentation, we consider $W \in \{3, 5, 10, 15, 20\}$, $\hat{W} = 50$, $N \in \{5, 10, 20, 50\}$ $k \in \{2, 3, 5, 7, 10\}$ $|C| \in \{2, 3, 4, 7\}$, $E = 50$, $\omega = 2$ $\hat{\omega} = 0.2$, $\hat{\alpha} = 5$. Through the adoption of the aforementioned parameters, we target to **(a)** simulate a dynamic environment where different numbers of tasks (W) are studied to conclude the final data filters; **(b)** simulate a environment where different numbers of nodes N are considered to participate in the ecosystem. This, in combination with different realizations of k, gives us the opportunity to create multiple experimental scenarios where we select different groups of nodes to decide the final allocation of data assets; **(c)** through $|C|$, we experiment with different number of groups in the clustering process to detect if a more 'relaxed' grouping activity (clusters with potentially high intra-cluster distance) negatively affects the final performance. It should be noted that all the parameters can be accessed through a configuration process/file in order to enhance the autonomous nature of the approach.

6.2. Performance assessment

We initially evaluate FMM as far as τ concerns for both datasets DS1, DS2. The delivered outcomes show that FMM requires [0.06, 0.08] seconds (approximately) to allocate a data vector $\mathbf{x}$. This means that 12.5 to 16.5 (approximately) data assets can be processed in a second which is a time interval that can support real time applications. These

[1] https://www.kaggle.com/datasets/shrutibhargava94/india-air-quality-data

[2] https://archive.ics.uci.edu/ml/datasets/GNFUV+Unmanned+Surface+Vehicles+Sensor+Data

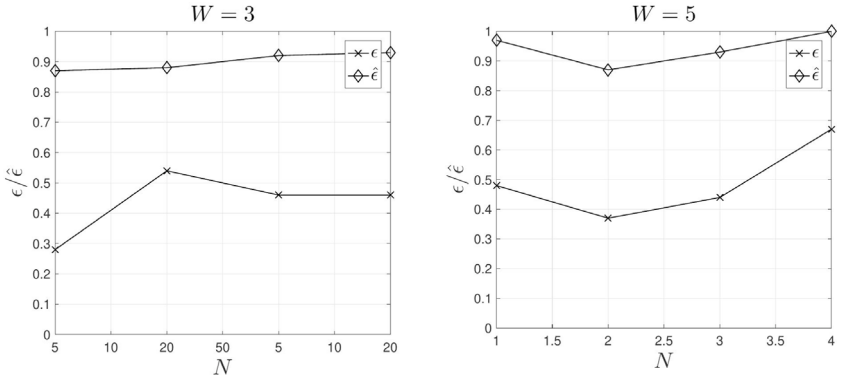


Fig. 2. Performance outcomes related to ϵ and $\hat{\epsilon}$ metrics for different N.

Table 2
Performance outcomes for ϵ and $\hat{\epsilon}$ metrics when DS2 feeds our model (different N).

N	$W=3,\ \|C\|=3,\ k=3$		$W=5,\ \|C\|=3,\ k=3$	
	ϵ	$\hat{\epsilon}$	ϵ	$\hat{\epsilon}$
5	0.28	0.87	0.42	0.96
10	0.28	0.85	0.50	0.93
20	0.30	0.95	0.60	1.00
50	0.38	0.95	0.58	0.96

Table 3
Performance outcomes for ρ and $\hat{\rho}$ metrics when DS2 feeds our model (different N).

N	$W=3,\ \|C\|=3,\ k=3$		$W=5,\ \|C\|=3,\ k=3$	
	ρ	$\hat{\rho}$	ρ	$\hat{\rho}$
5	0.27	0.81	0.33	0.92
10	0.24	0.83	0.54	0.93
20	0.36	0.88	0.60	0.97
50	0.36	0.89	0.58	0.92

results comes natural if we consider the time complexity of the adopted algorithm found to be low as it depends only on the complexity of the clustering algorithm and the number of nodes N. For instance, if we adopt the K-means algorithm, we can have a complexity of $O(M \cdot N^{|C|\cdot 2+1})$ [57] as we perform M clustering activities and we have to group pairs of low and high values for the adopted filters. Moreover, we have to notice that the calculation of the envisioned expectations of the variables that represent the low and high values of the filters are based on the incremental calculation of the corresponding means and standard deviations which does not add a significant burden in the final complexity.

In Figs. 2 & 3, we present our results for ϵ, $\hat{\epsilon}$, ρ, $\hat{\rho}$ for different number of nodes N adopting DS1 to feed our model. In this set of experiments, we keep $|C| = 3$, $k = 3$ and vary W between 3 (left) and 5 (right). We observe that FMM manages to reach high values for the adopted metrics especially when the disjunctive strategy is adopted. Recall that the disjunctive strategy indicates that at least 50% of the dimensions should match to $\mathbf{x}$ before a node is considered as a candidate for the data migration. We also observe that N positively affects the performance as FMM reaches values above 0.5 for ϵ and ρ and close to unity for the remaining parameters. If we focus on the ρ, $\hat{\rho}$ metrics, the vast majority of the selected nodes own filters that match to $\mathbf{x}$. Finally, $\hat{\epsilon}$ is also approaching unity for a high N exhibiting that the dataset of first node selected by FMM matches the allocated data asset for the majority of the envisioned dimensions. As far as the effect of W concerns, a high value positively affects the performance as FMM collects more requests for processing and produces filters that better correspond to the collected data. Tables 2 & 3 present our experimental results for the dataset DS2. We observe that ϵ and ρ values are less than the previous experimentation scenario with the dataset DS1 (the model adopting DS1 outperforms on average with 41% and 23% for ϵ and ρ, respectively). If we focus on the scenario where $W = 5$, the DS2 manages to achieve better results when $N \in \{10, 20\}$ (around -26% and -22% on average for ϵ and ρ, respectively).

In Fig. 4, we present the statistical distribution of the results by providing the boxplot for ϵ and ρ metrics. We clearly observe the 'statistical stability' of the outcomes as the inter-quartile distance is considered as low in the majority of the experimental scenarios. There is not any significant difference from the maximum and minimum values and we do not observe any outliers in our performance outcomes.

Table 4
Performance outcomes for ϵ and $\hat{\epsilon}$ metrics when DS2 feeds our model (different k).

k	$\|C\|=3,\ W=5,\ N=20$		$\|C\|=3,\ W=5,\ N=20$	
	ϵ	$\hat{\epsilon}$	ϵ	$\hat{\epsilon}$
2	0.47	0.87	0.56	0.96
5	0.53	1.00	0.52	0.97
7	0.37	0.93	0.43	1.00
10	0.37	0.96	0.48	1.00

Table 5
Performance outcomes for ρ and $\hat{\rho}$ metrics when DS2 feeds our model (different k).

k	$\|C\|=3,\ W=5,\ N=20$		$\|C\|=3,\ W=5,\ N=20$	
	ρ	$\hat{\rho}$	ρ	$\hat{\rho}$
2	0.45	0.88	0.46	0.98
5	0.49	0.97	0.45	0.94
7	0.39	0.91	0.37	0.87
10	0.37	0.90	0.40	0.88

In general, the boxplots confirm the previous observations and expose the difference in the performance when the two different datasets feed our model. In any case, this difference is considered as low and mainly depends on the interval where the dimensions are realized in the two datasets. This is more intense when only three tasks are utilized to define the data filters as it is exposed by the plots at the left part of the figure. If more tasks (i.e., $W = 5$) are utilized to define the filters, the two datasets lead to similar results.

In Figs. 5 & 6, we present our performance evaluation for different number of selected nodes k adopting DS1 to feed our model. We investigate if the final list of nodes plays a significant role in the outcomes. In this set of experiments, we keep $N = 20$, $W = 5$ and $|C|$ varies between 3 (left) and 5 (right). We observe that, naturally, k affects the outcomes especially if $\rho\&\hat{\rho}$ are our main concern. The increased k causes fluctuations in the outcomes as more nodes are included in the final list that may not completely match to $\mathbf{x}$ for the entire set of dimensions. However, the provided results exhibit that FMM manages to select nodes that are appropriate for hosting the rejected data especially if we adopt the disjunctive strategy. As far as ϵ and $\hat{\epsilon}$ concerns, we obtain similar results as in the previous experimentation scenario. Concerning the adoption of DS2, Tables 4 &

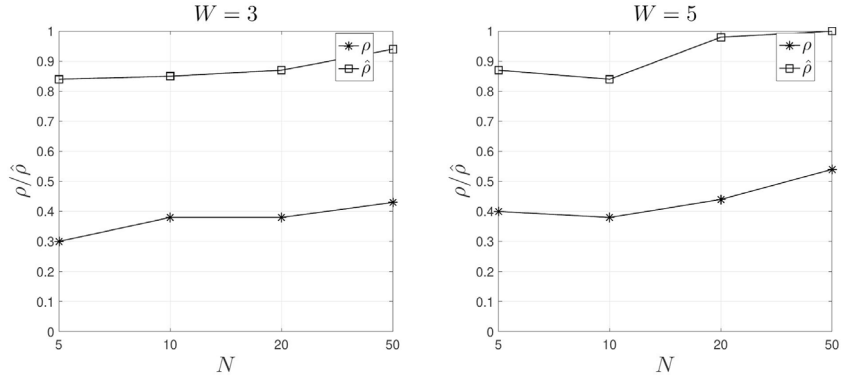


Fig. 3. Performance outcomes related to ρ and $\hat{\rho}$ metrics for different N.

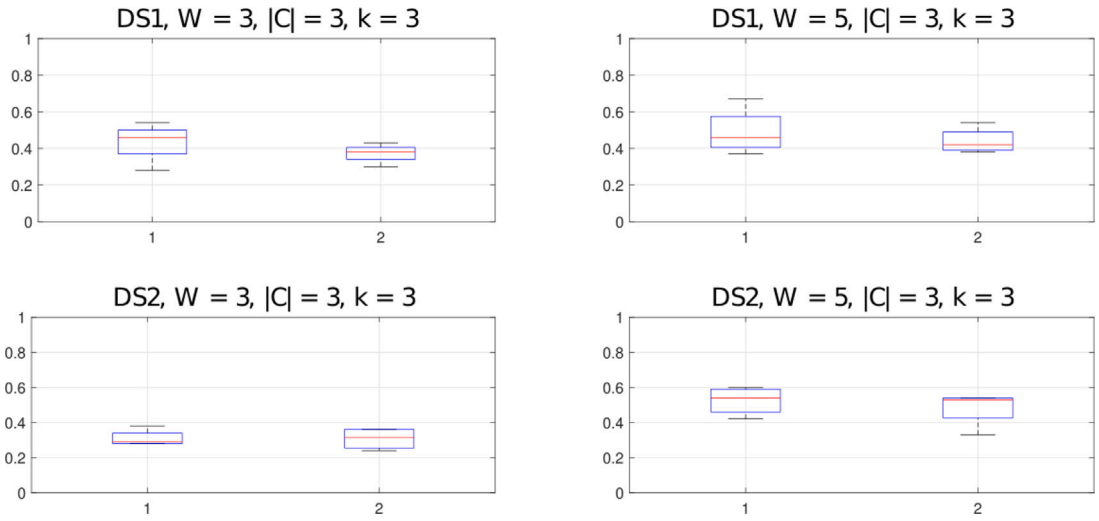


Fig. 4. Performance outcomes distribution for ϵ (1. Left) and ρ (2. Right) metrics and different N.

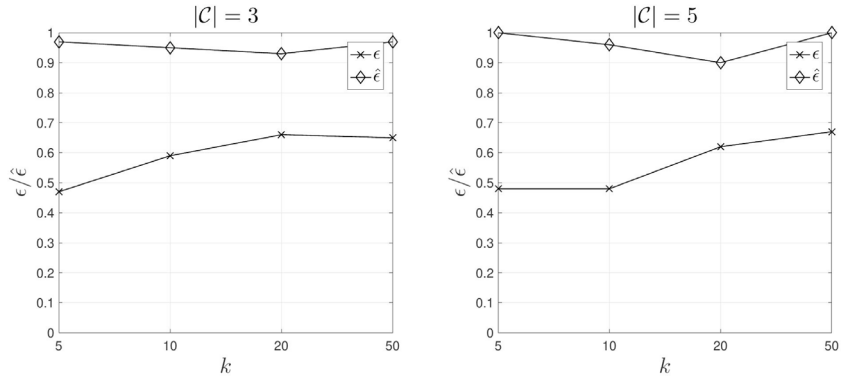


Fig. 5. Performance outcomes for different k (ϵ and $\hat{\epsilon}$ metrics).

5 present our results. Again the model with the DS1 as the core dataset leads to better results for ϵ and ρ metrics except the scenario where $|\mathcal{C}| = 5$ and $k \in \{2, 5\}$. The outcomes related to the $\hat{\epsilon}$ and $\hat{\rho}$ metrics are observed to be similar.

The next set of experiments refers to the adoption of different W and the corresponding results are presented by Fig. 7 when DS1 feeds our model. We keep $|\mathcal{C}| = 4$, $k = 5$ and N varies between 10 (left) and 20 (right). We observe that ϵ is increasing as more tasks are considered to be the basis for delivering the final filters. This means that FMM is capable of detecting filters (minimum and maximum values) that better depict the statistical description of the underlying requests by opening up the min–max interval of the requested data. Recall that data and requests are randomly produced upon the same dataset that governs our decision making. ρ is observed to be governed by fluctuations while the disjunctive strategy ($\hat{\epsilon}$, $\hat{\rho}$), again, manages to lead to results close to unity. The fluctuations in the ρ realization are due to the random management of the tasks and the adopted data as the selected nodes may not match to $\mathbf{x}$ for the envisioned dimensions. When DS2 feeds our model, we observe similar outcomes with $\epsilon \in \{0.43, 0.39, 0.38, 0.54\}$ for $N = 10$ and $\epsilon \in \{0.39, 0.31, 0.45, 0.73\}$ for $N = 20$. As far as ρ concerns we get $\rho \in \{0.30, 0.40, 0.37, 0.58\}$ for $N = 10$ and $\epsilon \in \{0.40, 0.29, 0.44, 0.75\}$ for $N = 20$ confirming our conclusion

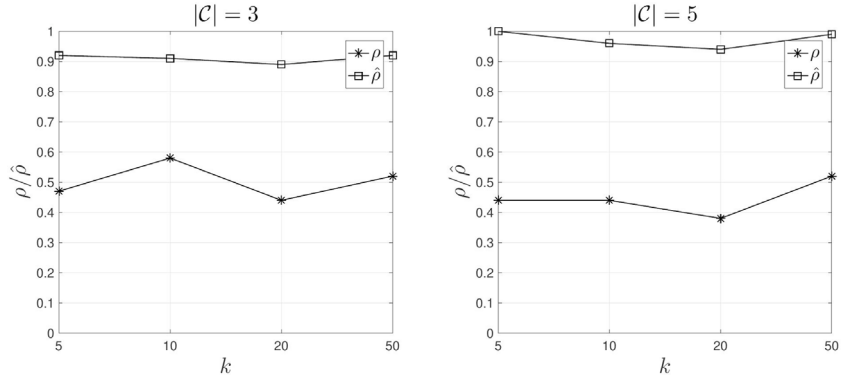


Fig. 6. Performance outcomes for different k (ρ and $\hat{\rho}$ metrics).

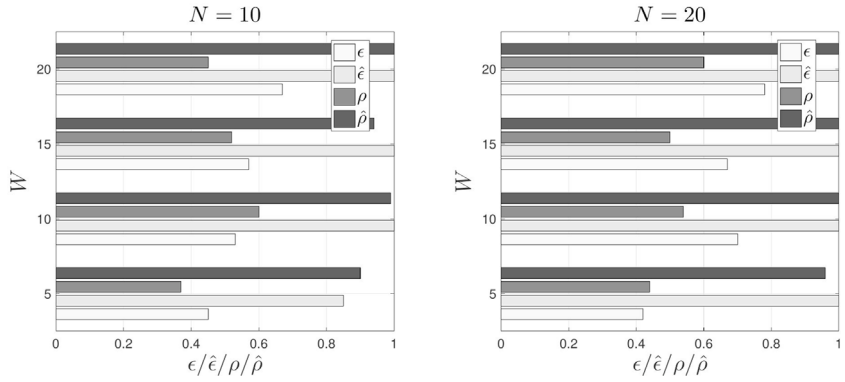


Fig. 7. The performance of the proposed model when it is applied for different W.

that the proposed model behaves well when a high number of tasks are processed to deliver the final filters. This is because the variability of the requests manages to lead to filters that clearly depict and affect the underlying data stored in every node.

In Fig. 8, we provide the performance outcomes for different $|\mathcal{C}|$ adopting DS1 to feed our model and keeping $k = 5$, $W = 10$ and N varies between 10 (left) and 20 (right). We observe that FMM manages to reach high values especially for ϵ, $\hat{\epsilon}$ and $\hat{\rho}$. We can observe that an increased number of nodes positively affect the results as FMM has the opportunity to process more values and detect the appropriate set of peers. The processing of more nodes gives us the ability to efficiently detect the groups of peers (through the clustering) and open up the room for the application of our model that demands for many nodes to be present in the delivered clusters. This can be seen in the results for $|\mathcal{C}| = 7$ and $N = 10$ where every cluster consists of a few nodes, thus, FMM decides, with a penalty (as explained above), to incorporate in the processing and the final list peers from other clusters than the cluster where the node offloading the data asset is present. If we focus on the scenario where DS2 feeds our model, we get: $\epsilon \in \{0.26, 0.43, 065\}$ &$\rho \in \{0.41, 0.37, 0.55\}$ for $N = 10$, $\epsilon \in \{0.57, 054, 0.60\}$ &$\rho \in \{0.44, 0.51, 0.48\}$ for $N = 20$. Again, we observe that the proposed model is in favour of a large ecosystem with many nodes and large groups of peers when the final decision should be made.

In Tables 6 & 7, we present our results for the comparative assessment between the proposed FMM and RM for both datasets DS1 and DS2. We observe that FMM outperforms RM for all the experimental scenarios no matter the adopted dataset. The higher the N, the higher the difference becomes. The random selection of nodes for offloading data it may be the optimal baseline model cornering the time required to deliver the final outcome, however, it does not lead to the best choices for migrating data. Naturally, an increased W, that affects the production of the filters, assists the RM to deliver better results as the statistical representation of the data and filters is better than in other cases. Recall that an increased W gives the opportunity to increase the min–max interval of the requested data.

Table 6
Comparative assessment between FMM and RM for different N (Dataset: DS1).

N	$W = 3,\ \|\mathcal{C}\| = 3,\ k = 3$		$W = 5,\ \|\mathcal{C}\| = 3,\ k = 3$	
	ϵ	ρ	ϵ	ρ
5	64.71%	30.43%	41.18%	5.26%
10	58.82%	22.58%	37.04%	35.71%
20	84.00%	40.74%	69.23%	46.67%
50	228.57%	79.17%	179.17%	80.00%

Table 7
Comparative assessment between FMM and RM for different N (Dataset: DS2).

N	$W = 3,\ \|\mathcal{C}\| = 3,\ k = 3$		$W = 5,\ \|\mathcal{C}\| = 3,\ k = 3$	
	ϵ	ρ	ϵ	ρ
5	0.00%	22.73%	20.00%	6.45%
10	115.38%	84.62%	28.21%	28.57%
20	42.86%	44.00%	30.43%	52.94%
50	137.50%	111.76%	70.59%	31.71%

In Tables 8 & 9, we see our comparative results for different k for both datasets DS1 and DS2. Now, an increased number of nodes present in the final selection list increases the chances of the RM to reduce the difference from the FMM as well. The higher the k is the lower the difference becomes. However, FMM outperforms RM again in all the experimentation scenarios except the scenario where DS2 feeds our model and we target to a limited number of nodes that participate in the final decision making ($k = 2$). We have to notice that in this set of experiments, we adopt $N = 20$ and $|\mathcal{C}| = \{3, 5\}$.

The performance outcomes provided by Tables 10 & 11 confirm our previous observations for the effects of a high W if we focus on the disjunctive strategy. This means that the RM can find room for

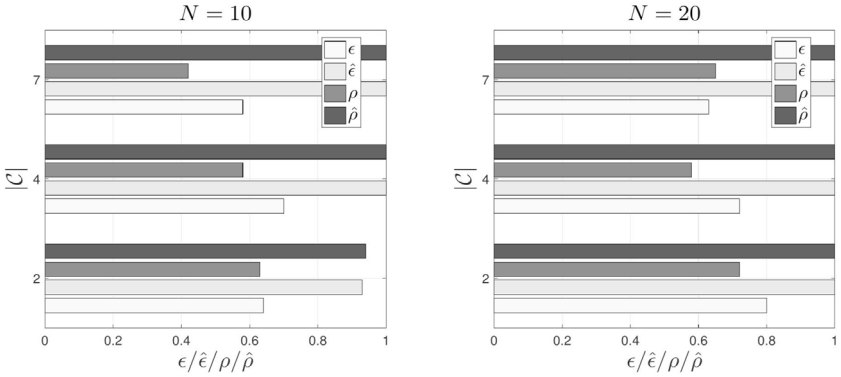


Fig. 8. The performance of the proposed model when it is applied for different number of clusters.

Table 8
Comparative assessment between FMM and RM for different k (Dataset: DS1).

k	$N = 20,\ \|\mathcal{C}\| = 3,\ W = 5$		$N = 20,\ \|\mathcal{C}\| = 5,\ W = 5$	
	ϵ	ρ	ϵ	ρ
2	135.00%	113.64%	60.00%	46.67%
5	84.38%	45.00%	45.45%	22.22%
7	94.12%	57.14%	82.35%	18.75%
10	44.44%	13.04%	63.41%	20.93%

Table 9
Comparative assessment between FMM and RM for different k (Dataset: DS2).

k	$N = 20,\ \|\mathcal{C}\| = 3,\ W = 5$		$N = 20,\ \|\mathcal{C}\| = 5,\ W = 5$	
	ϵ	ρ	ϵ	ρ
2	135.00%	66.67%	7.69%	−4.17%
5	23.26%	8.89%	79.31%	50.00%
7	12.12%	11.43%	0.00%	2.78%
10	42.31%	32.14%	26.32%	8.11%

Table 10
Comparative assessment between FMM and RM for different W (Dataset: DS1).

W	$N = 10,\ \|\mathcal{C}\| = 4,\ k = 5$		$N = 20,\ \|\mathcal{C}\| = 4,\ k = 5$	
	ϵ	ρ	ϵ	ρ
5	50.00%	15.63%	27.27%	22.22%
10	29.27%	33.33%	133.33%	68.75%
15	32.56%	−3.70%	34.00%	51.52%
20	103.03%	−4.26%	16.42%	22.45%

Table 11
Comparative assessment between FMM and RM for different W (Dataset: DS2).

W	$N = 10,\ \|\mathcal{C}\| = 4,\ k = 5$		$N = 20,\ \|\mathcal{C}\| = 4,\ k = 5$	
	ϵ	ρ	ϵ	ρ
5	30.30%	11.11%	21.88%	14.29%
10	−30.36%	5.26%	−32.61%	0.00%
15	0.00%	8.82%	25.00%	29.41%
20	0.00%	−6.45%	−8.75%	10.29%

randomly selecting nodes that may exhibit a high similarity with $\mathbf{x}$. There are two experimentation scenarios, i.e., $W \in \{15, 20\}$, where the RM outperforms the FMM if the ρ metric concerns having DS1 to feed our model. When DS2 is the case, RM outperforms FMM in four experimental scenarios but not a specific trend is observed. We can say that the DS2 lead the proposed model to have similar performance as RM. In any case, the outcomes of the two models are close enough. In all the remaining experimentation scenarios, the FMM outperforms the RM.

In Tables 12 & 13, we provide the comparative performance outcomes between FMM and DBAM taking into consideration ϵ and $\hat{\epsilon}$ metrics for different number of nodes N. Recall that DBAM considers that only the closest peer in the same cluster will be the new host

Table 12
Comparative assessment between FMM and DBAM for different N (Dataset: DS1).

N	$W = 3,\ \|\mathcal{C}\| = 3,\ k = 3$		$W = 5,\ \|\mathcal{C}\| = 3,\ k = 3$	
	ϵ	$\hat{\epsilon}$	ϵ	$\hat{\epsilon}$
5	180.00%	55.36%	60.00%	14.12%
10	80.00%	37.50%	105.56%	6.10%
20	76.92%	15.00%	25.71%	19.23%
50	53.33%	12.05%	45.65%	19.05%

Table 13
Comparative assessment between FMM and DBAM for different N (Dataset: DS2).

N	$W = 3,\ \|\mathcal{C}\| = 3,\ k = 3$		$W = 5,\ \|\mathcal{C}\| = 3,\ k = 3$	
	ϵ	$\hat{\epsilon}$	ϵ	$\hat{\epsilon}$
5	47.37%	10.13%	13.51%	15.66%
10	21.74%	8.97%	61.29%	0.00%
20	11.11%	4.40%	87.50%	29.87%
50	90.00%	31.94%	222.22%	17.07%

of the rejected data. The adoption of DS1 leads to the best possible results compared to the scenario where we adopt DS2. In all the experimentation scenarios, FMM outperforms DBAM and exhibit a very good performance exposing, at the same time, that only the distance between nodes cannot efficiently lead to the best possible allocations. For DS1 we observe that for an increased number of nodes DBAM menages to achieve better performance than in the other scenarios reducing the difference with FMM. This is more 'intense' in the $\hat{\epsilon}$ metric for which the two model exhibits similar behaviour. In the first places of our future research, we consider the combination of the distance based selection with the reasoning of the proposed model to enhance more the efficiency and lead to the best possible performance.

7. Conclusions

The current evolution of the Internet of Things (IoT) and Edge Computing (EC) leads to the combination of two vast infrastructures where various intelligent services can be uploaded to facilitate end users activities. EC nodes can be the mediators between IoT and the Cloud as far as the data transfer concerns and become the executors of tasks over collected data. Due to the computational and storage constraints of EC nodes' resources (they are limited compared to Cloud), we have to select the data that will be hosted locally. This data selectivity approach leads to the creation of distributed datasets with specific statistical characteristics. Our target is to provide a model for allocating data that do not match to the selectivity scheme of a node and keep them at the edge as much as possible by migrating those data assets to peers. We propose a migration model to peers owing similar datasets as they are exposed by the corresponding data filters. Our mechanism introduces the combination of Machine Learning (ML) for grouping

nodes based on their data filters with a probabilistic approach to detect the historical similarity between distributed datasets. We evaluate the proposed model to expose that is capable of delaying the transfer of data to the Cloud by selecting the appropriate peers to host them. The final outcome is the ability to support a comprehensive data map of the EC network, designating each node as the owner of specific data spaces, thus, facilitating the allocation of future processing tasks. In the first place of our future research plans is the definition of a more complex scheme for the management of the filters clustering outcomes in order to detect sub-clusters and the 'direction' of the similarity between two filters. We have to detect this similarity not only based on the distance between two pairs but also this distance should play a crucial role in the final selection of nodes especially when filters lie at the edges of the delivered clusters.

CRediT authorship contribution statement

Kostas Kolomvatsos: Conceptualization, Data curation, Methodology, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Christos Anagnostopoulos:** Formal analysis, Methodology, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Christos Anagnostopoulos reports financial support was provided by European Commission.

Data availability

No data was used for the research described in the article.

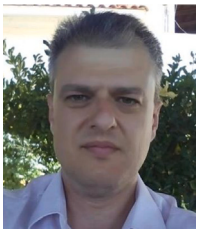
Acknowledgements

This work is partially funded by the Horizon Europe grant 'Integration & Harmonization of Logistic Operations' (TRACE) #101104278.

References

[1] S. Najam, et al., The role of edge computing in Internet of Things, IEEE Commun. Mag. (2018).
[2] S.K. Monga, et al., Elfstore: A resilient data storage service for federated edge and fog resources, in: IEEE International Conference on Web Services, ICWS, 2019.
[3] T. Nagato, et al., Distributed key–value storage for edge computing and its explicit data distribution method, IEICE Trans. Commun. (2019) http://dx.doi.org/10.1587/transcom.2019CPP0007.
[4] B. Charyyev, et al., Latency comparison of cloud datacenters and edge servers, in: 2020 IEEE Global Communications Conference, Taipei, Taiwan, 2020, pp. 1–6.
[5] K. Kolomvatsos, C. Anagnostopoulos, A proactive statistical model supporting services and tasks management in pervasive applications, IEEE Trans. Netw. Serv. Manag. 19 (3) (2022) 3020–3031.
[6] K. Kolomvatsos, A. Anagnostopoulos, Multi-criteria optimal task allocation at the edge, Future Gener. Comput. Syst. 93 (2019) 358–372.
[7] C. Anagnostopoulos, S. Hadjiefthymiades, K. Kolomvatsos, Accurate, dynamic & distributed localization of phenomena for mobile sensor networks, ACM TOSN 12 (2) (2016).
[8] C. Anagnostopoulos, S. Hadjiefthymiades, K. Kolomvatsos, Time-optimized user grouping in location based services, Comput. Netw. 81 (2015) 220–244.
[9] K. Kolomvatsos, et al., Data fusion and type-2 fuzzy inference in contextual data stream monitoring, IEEE TSMC:Syst. 47 (8) (2016) 1839–1853.
[10] K. Kolomvatsos, et al., An efficient environmental monitoring system adopting data fusion, prediction, & fuzzy logic, in: 6th IISA, 2015.
[11] K. Kolomvatsos, et al., Distributed localized contextual event reasoning under uncertainty, IEEE Internet Things J. 4 (1) (2017) 183–191.
[12] O.M. Elzeki, et al., Overview of scheduling tasks in distributed computing systems, Int. J. Soft Comput. Eng. (IJSCE) 2 (3) (2012) 2231–2307.
[13] L.-T.H. Hsieh, et al., Task management for cooperative mobile edge computing, in: 2020 IEEE/ACM Symposium on Edge Computing, SEC, San Jose, CA, USA, 2020, pp. 352–357.
[14] A. Karanika, P. Oikonomou, K. Kolomvatsos, T. Loukopoulos, A. Demand-driven, Proactive tasks management model at the edge, IEEE Fuzz-IEEE (2020).
[15] K. Bhardwaj, et al., AppSachet: Distributed app delivery from the edge cloud, in: 7th Intl. Conf. Mobile Computing, Applications, and Services, 2015, pp. 89–106.
[16] P. Simoens, et al., Scalable crowd-sourcing of video from mobile devices, in: 11th International Conference on Mobile Systems, Applications, and Services, 2013, pp. 139–152.
[17] T. Wang, et al., Fog-based computing and storage offloading for data synchronization in IoT, IEEE Internet Things 6 (2019) 4272–4282.
[18] K. Kolomvatsos, A proactive inference scheme for data-aware decision making in support of pervasive applications, Future Gener. Comput. Syst. 136 (2022) 193–204.
[19] F. Li, D. Wang, 5G network data migration service based on edge computing, Symmetry 13 (11) (2021) 2134.
[20] Y. Yao, et al., EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for wireless sensor networks, in: IEEE International Conference on Mobile Ad-Hoc and Smart Systems, MASS, 2013, pp. 182–190.
[21] A. Zhou, et al., Optimal mobile device selection for mobile cloud service providing, J. Supercomput. 72 (8) (2016) 3222–3235.
[22] S. Sardellitti, et al., Joint optimisation of radio and computational resources for multicell mobile-edge computing, IEEE Trans. Signal Inf. Process. Over Netw. 1 (2) (2015) 89—103.
[23] M. Breitbach, et al., Context-aware data and task placement in edge computing environments, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2019.
[24] G. Boulougaris, K. Kolomvatsos, A. QoS-aware, Proactive tasks offloading model for pervasive applications, in: 9th International Conference on Future Internet of Things and Cloud, FiCloud, 22-24 Aug, Rome, Italy, 2022.
[25] M. Awadalla, Task mapping and scheduling in wireless sensor networks, Int. J. Comput. Sci. 440 (4) (2013).
[26] A. Razavinegad, Task allocation in robot mobile wireless sensor networks, Int. J. Sci. Technol. Res. 3 (6) (2014).
[27] J. Yang, et al., Task allocation for wireless sensor network using modified binary particle swarm optimization, IEEE Sens. J. 14 (3) (2014) 882–892.
[28] X. Hu, B. Xu, Task allocation mechanism based on genetic algorithm in wireless sensor networks, in: ICAIC, 2011.
[29] B. Coltin, N. Veloso, Mobile robot task allocation in hybrid wireless sensors networks, in: 2010 International Conference on Intelligent Robots and Systems, 2010.
[30] T. Ouyang, et al., Cost-aware edge resource probing for infrastructure-free edge computing: From optimal stopping to layered learning, IEEE Real-Time Syst. (2019).
[31] K. Kolomvatsos, C. Anagnostopoulos, A probabilistic model for assigning queries at the edge, Computing 102 (2020) 865–892.
[32] M. Soula, A. Karanika, K. Kolomvatsos, C. Anagnostopoulos, G. Stamoulis, Intelligent Tasks Allocation at the Edge Based on Machine Learning and Bio-Inspired Algorithms, Evolving Systems, Springer, 2021.
[33] K. Kolomvatsos, et al., Proactive & time-optimized data synopsis management at the edge, IEEE TKDE (2020).
[34] T. Huang, et al., A latency-aware multiple data replicas placement strategy for fog computing, J. Signal Process. Syst. 91 (2019) 1191–1204.
[35] V.D. Kumar, et al., Efficient data transfer in edge envisioned environment using artificial intelligence based edge node algorithm, Trans. Emerg. Telecommun. Technol. 32 (2021) e4110.
[36] H.Q. Le, et al., Efficient resource allocation in mobile-edge computation offloading: completion time minimization, in: IEEE International Symposium on Information Theory, ISIT, Aachen, Germany, 2017, pp. 2513–2517.
[37] H. Li, EPPDR: An efficient privacy-preserving demand response scheme with adaptive key evolution in smart grid, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2014) 2053–2064.
[38] Y. Zhang, et al., Privacy-preserving data aggregation against false data injection attacks in fog computing, Sensors 18 (8) (2018) 2659.
[39] M.G. Rao, et al., A secure and efficient data migration over cloud computing, IOP Conf. Ser.: Mater. Sci. Eng. 1099 (2021) 012082.
[40] B. Han, et al., Mobile data offloading through opportunistic communications and social participation, IEEE Trans. Mob. Comput. 11 (5) (2012) 821–834.
[41] K. Kamalpreet, R. Vanita, A systematic way to cloud server optimization using orthogonal recursive bisection technique for cloud migration, IJCSIT 6 (6) (2015).
[42] A. Rath, A. Hristoskova, S. Klein, PFilter: Privacy-aware and secure data filtering at the edge for distributed edge analytics, in: AIAI 2021 IFIP WG 12.5 International Workshops, vol 628, Springer, Cham.
[43] S. Zöller, et al., Distributed data filtering in logistics wireless sensor networks based on transmission relevance, in: 37th Annual IEEE Conference on Local Computer Networks, 2012, pp. 256–259, http://dx.doi.org/10.1109/LCN.2012.6423622.

[44] T. Xia, C. Jin, X. Zhou, A. Zhou, Filtering duplicate items over distributed data streams, in: W. Fan, Z. Wu, J. Yang (Eds.), Advances in Web-Age Information Management, WAIM 2005, in: Lecture Notes in Computer Science, vol. 3739, Springer, Berlin, Heidelberg.
[45] Song Deng, Dong Yue, Aihua Zhou, Xiong Fu, Lechan Yang, Yu Xue, Distributed content filtering algorithm based on data label and policy expression in active distribution networks, Neurocomputing 270 (2017) 159–169.
[46] G. Koloniari, N. Ntarmos, E. Pitoura, D. Souravlias, One is enough: distributed filtering for duplicate elimination, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, 2011, pp. 433–442.
[47] H. Nikpey Somehsaraei, S. Ghosh, S. Maity, P. Pramanik, S. De, M. Assadi, Automated data filtering approach for ANN modeling of distributed energy systems: Exploring the application of machine learning, Energies 13 (2020) 3750.
[48] X. Xu, W. Dai, 'Data. Acquisition, Filtering and buffering protocol design for edge computing nodes, in: 19th IEEE International Conference on Industrial Informatics, INDIN, 2021, pp. 1–6.
[49] Y.J. Kim, C.-H. Park, M. Yoon, FILM: Filtering and machine learning for malware detection in edge computing, Sensors 22 (6) (2022) 2150.
[50] F. Sun, Z. Diao, Edge node aware adaptive data processing method for ubiquitous NB-IoT, J. Sensors 2022 (2022) 9006152.
[51] P. Lou, et al., A data-driven adaptive sampling method based on edge computing, Sensors 20 (2020) 2174.
[52] N.A. Sulieman, L. Ricciardi Celsi, W. Li, A. Zomaya, M. Villari, Edge-oriented computing: A survey on research and use cases, Energies 15 (2022) 452.
[53] K. Kolomvatsos, et al., A time optimized scheme for top-k list maintenance over incomplete data streams, Inform. Sci. 311 (2015) 59–73.
[54] Q. Long, K. Kolomvatsos, C. Anagnostopoulos, 'Knowledge reuse in edge computing environments, J. Netw. Comput. Appl. 206 (2022).
[55] Eric W. Weisstein, Normal difference distribution. From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/NormalDifferenceDistribution.html.
[56] X. Wei, et al., Data placement strategies for data-intensive computing over edge clouds, in: 2021 IEEE International Performance, Computing, and Communications Conference, IPCCC, Austin, USA, 2021, pp. 1–8.
[57] M. Inaba, et al., Applications of weighted voronoi diagrams and randomization to variance-based k-clustering, in: Proceedings of 10th ACM Symposium on Computational Geometry, 1994, pp. 332–339.



Dr. Kostas Kolomvatsos received his B.Sc. in Informatics from the Department of Informatics at the Athens University of Economics and Business, his M.Sc. and his Ph.D. in Computer Science from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens. Currently, he serves as an Assistant Professor in the Department of Informatics and Telecommunications, University of Thessaly. He was a Marie Skłodowska Curie Fellow (Individual Fellowship) at the School of Computing Science, University of Glasgow. He is the leader of the Intelligent Pervasive Systems (iPRISM) research group at the Department of Informatics and Telecommunications, University of Thessaly. His research interests are in the definition of Intelligent Systems adopting Machine Learning, Computational Intelligence and Soft Computing for Pervasive Computing, Pervasive data Science, Distributed Systems, Internet of Things, (Pervasive) Edge Computing and the management of Large Scale Data. He is the author of over 130 publications in the aforementioned domains.

Dr. Christos Anagnostopoulos is an Associate Professor (Senior Lecturer) in Distributed Computing & Data Engineering Systems and Programme Director of the M.Sc. Information Technology and M.Sc. Software Development, School of Computing Science, University of Glasgow. His research expertise is at the intersection of large-scale distributed computing and distributed machine learning at the Edge. He is the author of over 170 scientific journals and conferences and the leader of the Knowledge and Data Engineering Systems Group (IDA Section). Before joining Glasgow, he was an Assistant Professor at Ionian University and Adjunct Assistant Professor at the University of Athens. He has held Postdoctoral position at University of Athens and Research Fellow at University of Glasgow in the area of large-scale distributed computing. He holds a Ph.D. in Computing Science, University of Athens (2008), and he is an associate fellow of the HEA and professional member of ACM, IEEE and IEEE STC.