A Proactive Statistical Model Supporting Services and Tasks Management in Pervasive Applications

Kostas Kolomvatsos and Christos Anagnostopoulos

***Abstract*—The combination of the Internet of Things (IoT) and Edge Computing (EC) can support intelligent pervasive applications that meet the needs of end users. A challenge is to provide efficient inference models for supporting collaborative activities. EC nodes can interact with IoT devices and each other to conclude those activities producing knowledge. In this paper, we propose a proactive scheme to decide upon the efficient management of services and tasks present/reported at EC nodes. Services can be processing modules applied upon local data while being required for the execution of tasks. We monitor the demand for the available services and reason upon their management, i.e., for their local presence/invocation as the demand is updated by the requested processing activities. For each incoming task, an inference process is fired to proactively meet the strategic targets of the envisioned model. We propose a statistical inference process upon the demand for services and the contextual performance data of nodes combining it with a utility aware decision making model. Instead of exclusively focusing on services migration or tasks offloading as other relevant efforts do, we elaborate on the decision making for the selection of one of the aforementioned activities (the most appropriate at a specific time instance). We present our model and evaluate it through a high number of simulations to expose its pros and cons placing it in the respective literature as one of the first attempts to proactively decide the presence of services to an ecosystem of processing nodes.**

***Index Terms*—Edge Computing, Decision Making, Internet of Things, Pervasive Applications, Proactive Inference, Utility Theory.**

I. Introduction

The current evolution of the Internet of Things (IoT) and Edge Computing (EC) provides the opportunity to have numerous devices and processing nodes very close to end users in order to support pervasive services [1]. Such services are provided upon data being collected by IoT devices and transferred, in an upwards mode, to EC processing nodes. EC nodes exhibit limited storage and computational capabilities compared to Cloud, however, they are able to host a number of services facilitating the execution of various processing activities. Such activities are, usually, reported in the form of tasks that 'demand' for retrieving data and knowledge (e.g., predictive analytics, explanatory-driven models). The interesting is that the combination of the IoT with the EC can assist in the realization of collaborative activities where nodes can cooperate each other and conclude the requested tasks. For instance, nodes may exchange services or data or even more offload tasks to be executed by peers under some conditions. Obviously, the speed of reporting tasks and data should be carefully taken into consideration in order to offer efficient pervasive services.

EC nodes, due to the limited storage and computational capabilities, can host only a (sub-)set of the available services and the collected data. Services are necessary to execute the requested tasks dictated by users or applications. Note that users may not be static, thus, the demand for services continuously change due to the variation of tasks' requirements. This means that a task may request a service that could be not locally present. Then, nodes have two solutions, i.e., (i) to request a service migration from peers/Cloud only if the service fits to their computational characteristics; or (ii) offload the task to the peers that already host the service(s). The aforementioned migration activity deals with the decision of *where to migrate a service* for maintaining a high Quality-of-Service (QoS). Evidently, the delivery of the optimal migration strategy is intractable due to the dynamics of the EC ecosystem and the mobility of users. Tasks offloading deals with their uploading to peers/Cloud due to conditions that make impossible the local management. For instance, the node receiving a task may exhibit a high load, thus, it may not secure the efficient conclusion in the time constraints defined by the requestor. Inevitably, the multi-parametric problem (e.g., current load, peers status, network status, time constraints) and its dynamic nature impose various difficulties to find the optimal solution.

Motivated by the above challenges, we propose a novel model for supporting the behaviour of EC nodes. Our intention is to proactively decide the presence/absence of services at every node that serves the incoming tasks. We rely on the use of a statistical model that adopts the theory of order statistics [2] and a decision mechanism built upon the principles of Utility Theory [3]. The reception of a task fires our model and concludes if it is profitable to execute it locally or offload it in a peer node. We reason upon the expected utility of the available actions (i.e., execute the task locally or offload it to peers) to maximize the performance based on the contextual information (e.g., current load, communication overhead) of the node taking the decision. A local execution may initiate the request of a service from peers/Cloud as it could not be present locally. Nodes may decide to host popular services maximizing the re-use of resources when executing tasks. Our model is applied at epochs, i.e., nodes are capable of 'following' the seasonal demand patterns for the available services, thus, they are fully aligned with the need of users/applications. Epochs

Manuscript received for review August, 2021; Corresponding author: K. Kolomvatsos.

K. Kolomvatsos is with the Department of Informatics and Telecommunications, University of Thessaly, Papasiopoulou 2-4, 35131 Lamia Greece.
e-mail: kostasks@uth.gr

C. Anagnostopoulos is with the School of Computing Science, University of Glasgow, 17 Lilybank Gardens, G12 8RZ Glasgow UK.
e-mail: christos.anagnostopoulos@glasgow.ac.uk

can be 'statically' defined or depict the changes in the demand patterns at sequential time instances. The expiration of an epoch fires the re-initialization of the monitoring model to depict the new trends in the demand of services. Our scheme differentiates form other efforts, i.e., we deviate from the legacy service migration schemes and give the initiative to EC nodes for deciding which services will be kept locally. The proposed model proactively decides *the selection between two alternative activities* (we do not focus on the modelling of the activities themselves), i.e., the migration of services to support the local execution of tasks or tasks offloading to peers/Cloud when the status of nodes cannot guarantee the timely and efficient provision of responses. Other relevant efforts exclusively focus on the modelling of services migration or tasks' offloading. In our past efforts, we deal with the selection of tasks that will be executed locally [4] and the selection of peers where tasks could be offloaded [5]. In these efforts, we propose a decision mechanism that is based on the load burden added by a task and the similarity of data at the node offloading the task and the selected peers. Additionally, in [6], we consider the local and the global demand for a task (not services like in the current effort) before we decide the final list of peers that will host it. The difference of the current work is that we consider the potential absence of the necessary services in a node. This makes us to focus on the utility that nodes gain for every decision, i.e., service migration or task offloading. Our analysis focuses on an opportunistic model where every node takes into consideration only the current utilities of the discussed activities. The salient contributions of our work are as follows:

- a proactive statistical inference model built upon the order statistics of the services' demand adopted to retrieve the expected utility of their presence in a node as well as the expected utility of offloading actions. Both utilities are, then, combined with the corresponding cost in terms of the expected response time to deliver the final decision;
- a utility aware decision making model for concluding the place where the incoming tasks will be executed;
- an extensive experimental evaluation and comparative assessment with other models found in the literature showcasing the strengths of our approach.

The paper is organized as follows. In Section II, we provide the discussion upon the related work while in & Section III, we give some introductory information around our problem and the envisioned model. In Section IV, we analytically describe the proposed scheme and expose our theoretical contributions. In Section V, we provide details about the envisioned experimental setup and the simulation outcomes. Finally, in Section VI, we conclude our paper and provide insights on our future research agenda.

II. Related Work

The current trend in the EC is the adoption of (micro)servers close to users instead of centralized devices, thus, it is characterized by a significant reduction of the response time [7]. The interested reader can find a survey on the dedicated hardware and data management processes that can be adopted at the EC in [8]. EC can be 'formulated' in an hierarchical model, i.e., defining functions based on distance and resources. A relevant study is presented in [9] where a methodology for integrating Mobile Edge Computing (MEC) servers and cloudlets is discussed. Obviously, a proper resource management is required as task offloading activities may cause more burden in terms of downtime and energy costs [10]. In general, resource allocation is realized upon a specific strategy that targets to optimize a set of key performance indicators. These strategies are responsible to deliver accessible resources for every request or task [11]. An example architecture is proposed by [12] where the authors focus on the combination of the Cloud and fog layers for resources optimization. Some key performance indicators adopted in the optimization scheme are the requests per hour, response time as well as processing time all managed by a round-robin algorithm, equally spread execution algorithm, and authors' algorithm. The allocation of resources can be modelled as a Gaussian Process Regression scheme to meet the requirements of applications [13]. Such an approach targets to the estimation of the arrival of requests upon historical data, however, the prediction error should be carefully taken into consideration for any decision making. Dynamic resources allocation can be combined with static constraints in scheduling to conclude the best possible solution [14]. The ultimate goal is to achieve load balancing and distribute the computational burden to the available nodes [15]. The aforementioned approaches can assist in reducing the flow latency and increasing the performance while they can adopt complex methodologies due to the incorporation of multiple parameters in the decision making mechanism. Additionally, when focusing on tasks' priorities, measures should be evaluated concerning the 'starvation' event, i.e., task with low priorities should be fairly managed. Apart from the allocation of resources, another challenge is the management of data present at the discussed layers. The collected data that may be reported by a high number of streams, e.g., the evolution of a certain phenomenon (e.g., fire, air contamination) [16], [17], [18], streams generated by autonomous nodes (e.g., unmanned vehicles) [19] and so on and so forth. Upon these streams, we can easily support the extraction of knowledge, the detection of events or any other processing [20]. Obviously, data synchronization or migration could be adopted to setup a common basis for executing the requested processing. A data synchronization model for the envisioned infrastructure is necessary [21]. For instance, a mechanism for offloading computing parts and storage functionalities among the available nodes combined with a coordination model could increase the efficiency of the system. The difficulty is to maintain the required information about the status of numerous processing nodes to conclude the best possible decisions.

Data and services migration can be adopted to fill the 'gaps' in the local knowledge base of a processing node. The interested reader can find a survey of the relevant efforts in [22]. It will be convenient when migrating services to locally cover the processing needs, however, a set of requirements should be taken into consideration. Service migration can be realized in parts to avoid burdening the network, i.e., service pieces should be 'hooked' to the hosting infrastructure rapidly,

smoothly and in an automated manner. Machine Learning (ML) is one of the technologies utilized for such purposes. ML can be the basis for delivering models that estimate the mobility of users and support a proactive services migration mechanism [23]. Reinforcement learning can enhance the behaviour of agents when deciding the appropriate line of actions [24], [25]. The challenge when relying on supervised ML technologies is the discovery of the most representative dataset that will become the basis for its training. It is difficult to collect data that incorporate all the possible versions of nodes' status as there is an increased level of uncertainty in the corresponding decision making. The service migration model could be also formulated as an online queue stability control problem [26]. The problem can be solved by a Lyapunov optimization or a multi-objective optimization scheme [28], [27] and conclude the Pareto optimal solution. Past efforts mainly focus on the minimization of the latency under constraints for energy consumption in EC setups. However, the majority of the research efforts that target to a user-centric service migration model are 'affected' by the increased time to solve the optimization problem unless a set of assumptions are taken into account or approximate solutions can be tolerated. The use of Markov chains is also a potential pathway to solve the problem. In [29], the authors provide the solution to a finite-state Markov decision process through the utilization of a modified policy-iteration algorithm. A Thompson-sampling based scheme is proposed in [30] to support dynamic service migration decisions while in [31], the authors discuss a service allocation module for networked vehicle supported by services located at the Cloud. The target is to enhance the autonomous level of vehicles being capable of selecting neighbours to share the adopted services. In any case, it is proven that Markov-oriented solutions may fail to converge quickly and take extremely long to reach their stationary distribution [32].

Tasks offloading is another subject that has attracted the attention of many researchers. This research domain is important when being combined with the services migration and resources management problems. Scheduling algorithms have been proposed to manage a large set of tasks while their dependencies and requirements for resources should be taken into consideration [33]. The decision making related to tasks offloading may be considered as a multi-criteria problem where multiple parameters could affect the selection of the appropriate processing node where tasks should be allocated [4]. Again, ML and reinforcement learning can optimize the desired actions [34]. Optimization techniques is another solution, e.g., Integer Linear Programming (ILP) combined with a set of heuristics [35]. The drawbacks of the two techniques are already presented above. Particle Swarm Optimization (PSO) can facilitate the detection of the optimal solution towards offloading tasks in the best possible manner [36], [37]. The challenge now is the design and adoption of the appropriate transfer function to deliver a new position management process fully aligned with the requirements of the problem. PSO based solutions may suffer from premature convergence as proven in the respective literature [38]. Genetic algorithms [39] and distributed models [40] could also assist in the provision of the most efficient allocations maintaining the balancing between energy consumption and accuracy. Additionally, tasks may consist of a set of sub-tasks, thus, their dependencies (if present) should be studied. A scheduling algorithm for a set of sub-tasks is proposed in [41] which targets to assign each sub-task into the available nodes for maximizing the performance.

From the above discussion, we can easily detect the use of ML, Computational Intelligence or optimization techniques for deciding the migration of services or the offloading of tasks. An interesting difference of our approach is that it considers both problems at the same time enhancing the ability of nodes to conclude decisions based on the utility they gain selecting the correct action. We do not deal with a method for services migration or tasks offloading but with the selection of the appropriate action no matter the model adopted to perform it. Our statistical model does not demand for a training phase like the efforts adopting ML or the use of multiple conditions/constraints that may lead to sub-optimal solutions like optimization models. The envisioned utility oriented scheme maximizes the benefits for every node in the network and differentiates from other past activities in the domain. In the following table, we provide a comparison of the above presented research efforts and our scheme (SM: Service Migration; TO: Tasks Offloading).

TABLE I: Comparison of research efforts

Research Effort	Target	Technology
[23], [24], [25]	SM	ML
[27], [28]	SM	Optimization
[29], [30], [31]	SM	Statistics based
[4]	TO	Operations research
[34]	TO	ML
[35]	TO	Optimization
[36], [37]	TO	Computational Intelligence
[39]	TO	Genetic Models
Our model	Meta-model (SM/TO)	Utility Theory, Statistics

III. Preliminaries & High Level Description

A. Processing Nodes Management

The nomenclature adopted in this paper is presented by Table II. We consider a scenario where an ecosystem of processing nodes is available for executing various tasks through the use of a set of services. Tasks and services can be of any type ranging from simple processing activities (e.g., statistical processing) to more complex scenarios (e.g., the retrieval of analytics). Without loss of generality, we assume N services, i.e., $\mathscr{S} = \{s_1, s_2, \ldots, s_N\}$. For instance, one can rely on software-defined services for edge processing, thus, we are able to collect data from the IoT environment, process them locally and send the outcome (or a part of the collected data) to the Cloud for further processing. We have to notice that every node has specific computational capabilities for executing a specific set of services and tasks, respectively. The execution of tasks has the target of producing knowledge locally and making the nodes capable of exchanging this knowledge or share it in an upwards mode, towards to the Cloud infrastructure.

Tasks, with the assistance of the N services, are executed upon the locally formulated dataset, i.e., a set of multivariate

vectors $\mathbf{x} = [x_1, x_2, \ldots, x_M]$ with M being the number of dimensions. We consider a stream of tasks for every node $\mathscr{T}^i$ coming from the IoT ecosystem or peers present in the group (when offloading actions are selected). $\mathscr{T}^i$ is the 'hook' of each node with the external environment and facilitates the reporting of tasks formulated in a series of tuples $\langle t, T_t^i, \mathscr{C}_t^i \rangle$; t is the timestamp of the task T_t^i and $\mathscr{C}_t^i$ is a set of constraints. Constraints represent a set of conditions that should be met when executing T_t^i. For instance, T_t^i could demand for the presence of specific services, a deadline for delivering the final outcome and so on and so forth. T_t^i is also reported in the form of the desired processing, e.g., conditions upon the data for which the requested processing will take place. These conditions can be in the form of range queries or any other scheme that the requestor desires. Obviously, every node is capable of detecting the reporting scheme and export the aforementioned characteristics. The study of the discussed scheme lies beyond the scope of the this paper.

Nodes maintain a data structure where they store information related to the demand for each service. Consider the *Services Demand Vector (SDV)* for the ith node, i.e., $\mathbf{d}^i = \left[d_1^i, d_2^i, \ldots, d_N^i\right]$ with $d_j^i \in \mathbb{R}^+$. $\mathbf{d}^i$ is, evidently, affected by the incoming tasks and the processing requests that they demand. SDVs are lists of counters that depict the demand for every available service. After the reception of T_t^i, we can easily detect the required services through the tasks' metadata or the requested processing. For instance, when T_t^i requires a set of regression coefficients, the node receiving it increases the frequency of the corresponding regression service executed upon the local data (or a subset of data if requested). Furthermore, every node is characterized by a load l_t^i at time instance t as the percentage of the maximum load that can support. Tasks, after their reception, are placed in the local queue which exhibits a maximum size as dictated by the capabilities of nodes. l_t^i depicts the occupied percentage of the local queue. When this queue is full, nodes should offload all the upcoming tasks as there is no room for their execution. Additionally, a task can be requested by multiple users/applications, thus, nodes can re-use the spent resources and limit the latency in the provision of responses.

A task may require the implication of multiple services to be efficiently concluded. Suppose that T_t^i requires the implication of $\mathscr{S}' = \left\{s_1', s_2', \ldots, s_{N'}'\right\}$ with $\mathscr{S}' \subset \mathscr{S}$ and $N' \leq N$. Without loss of generality, we consider that the demand for each service at t is realized as the counter of requests for the specific service after receiving the incoming tasks and concluding their execution plans. This means that $\mathbf{d}^i$ is continually updated with the arrival of new tasks becoming the basis for our decision making (described later). We consider that demand indicators are monitored in epochs to avoid the cumulative effect. This means that, at pre-defined intervals, nodes re-initiate the aforementioned data structure and start over the demand monitoring process. Naturally, the adopted data structure contains the observed demand seasonal trends for every service being fully aligned with the recent requests of users/applications as delivered by the incoming tasks. A 'static' definition of the epochs will 'fire' the proposed model after the expiration of the pre-defined time interval. A dynamic approach will incorporate into the decision making a more complex statistical processing for the definition of epochs. For instance, we can easily detect two scenarios concerning the demand for services. The former relates to a 'static' trend which means that the demand for services remains constant during the last monitored time instances while the latter refers in an increasing trend representing a cumulative approach. The dynamic definition of epochs can be related to the detection of the cumulative effect. An example methodology useful for designing monitoring models is the *Minimum Detectable Effect (MDE)* [42]. With the MDE, we identify the change of the smallest magnitude for which a two-sided confidence interval about that change would not include zero. In other words, we rely on the detection of the difference in means between consecutive observations and the event that the estimated difference could be higher than the half of the confidence interval width and deemed statistically significant in approximately 50% of the experiments. In conclusion, an epoch could expire when we observe a significant statistical difference in consecutive means calculated upon the latest time instances. Additionally, nodes may decide to evict some services due to the limited computational capabilities or replace them. The specific mechanism adopted for the eviction of services or their replacement is left for future work.

B. Context aware Services & Tasks Management

Our approach follows a simple rationale, i.e., to *keep the execution of popular tasks locally if the current load is at 'acceptable' levels* (the node is not overloaded). As described earlier, every service corresponds to a specific demand, thus, nodes should decide *if they will request for the migration of the required services or offload the incoming task*. We strategically decide to maintain only popular services locally as edge nodes are characterized by limited storage and computational capabilities offloading tasks requiring non popular services to peers. Concerning the execution of popular services, nodes can benefit from the re-use of the already present resources and, when possible, from intermediate results. Incremental models or caching can be some example solutions towards the re-usability of services/resources.

Naturally, nodes should take into consideration the trade off between the discussed decisions: **Decision 1**. Keep locally the execution of the task and, if needed, request the necessary services; **Decision 2**. Offload the task to the appropriate peer(s). Decision 2 is taken subject to the Decision 1, i.e., the discussed decisions are made in a sequential order. Assume that, in case of Decision 1, T_t^i requires α_1 time units for waiting in the local queue and α_2 time units to be executed. Additional α_3 time units are required to request the necessary services (if not locally present) and install them before the execution of T_t^i begins. Hence, for Decision 1, nodes require $\hat{\alpha} = \alpha_1 + \alpha_2 + \mathbb{1}_{\neg \mathscr{S}'} \alpha_3$ where $\mathbb{1}_{\neg \mathscr{S}'}$ is the binary indicator function that depicts if one or more services required by T_t^i are missing. For Decision 2, nodes require β_1 time units to decide the peer that will host T_t^i, β_2 time units to execute T_t^i in the remote peer (β_2 contains the waiting time in the queue and the execution time) and β_3 time units to send T_t^i (after the

initial decision for offloading it) and receive the outcomes. β_3 represents the total communication time and depends on the status of the network. The total required time for Decision 2 is $\hat{\beta} = \beta_1 + \beta_2 + \beta_3$. Every node, depending on the final decision, has to wait for $\hat{\alpha}$ or $\hat{\beta}$ before it is ready to return the final outcome to the requestor.

Apart from the aforementioned time requirements, nodes should take into consideration the 'long-term' strategy for executing locally the incoming tasks in terms of the presence of multiple services and the re-use of resources. Recall that the popularity of services/tasks may affect the final decision related to the local execution or the offloading action. Nodes may offload tasks that demand for non popular services, thus, paying $\hat{\beta}$, to leave the room open for popular services. However, as nodes act in a very dynamic environment, the demand for services may naturally be updated and a different decision can be made in subsequent steps. In Figure 1, we show an example of the envisioned environment where EC nodes may have interactions between them and with IoT devices.

Nodes, after receiving T_t^i, update, based on their current status and T_t^i's requirements, $\mathbf{d}^i$ and l_t^i checking if the necessary services are present. The update of the demand is easily concluded through the use of simple counters while the update of l_t^i depends on the current status of the queue and the load that T_t^i causes. The latter process can be based on our previous findings [4], [43] where we propose a specific methodology for exposing the load that a task will add to a processing node. l_t^i mainly depends on the complexity of T_t^i and the implication of multiple services. Without loss of generality, we consider that l_t^i is represented by a real value in the unity interval. After updating $\mathbf{d}^i$ and l_t^i, nodes elaborate on Decisions 1 or 2. The proposed decision making mechanism is a function $g: \{\mathbf{d}^i, l_t^i, T_t^i\} \rightarrow \{A\}$ where A is the set of the available actions $A = \{\text{Decision 1}, \text{Decision 2}\}$. Actually, g is adopted to deliver the utility for each decision and concludes the final action for T_t^i upon $\mathbf{d}^i$ & l_t^i. If Decision 1 is selected, nodes confirm the updated l_t^i and place T_t^i at the local queue facing a time cost equal to $\hat{\alpha}$. If Decision 2 is decided, nodes keep l_{t-1}^i as the current load and decides the peer(s) where T_t^i will be offloaded (e.g., adopting a model like the one presented in [4]) while facing a time cost equal to $\hat{\beta}$. In any case, $\mathbf{d}^i$ is kept in the updated form to depict the up-to-date demand indication for every service under consideration. Our expected utility based model is appropriate to depict the expected 'payoffs' when deciding the execution of tasks. The proposed model tries to cover the uncertainty about the correct decision under the dynamics of the EC environment. It can be combined with any scheme adopted for the pre-processing of the incoming tasks as well as any mechanism adopted to deliver the peers for offloading actions when Decision 2 is considered.

IV. Utility based Contextual Decision Making

A. Theoretical Setup

The realization of the proposed model is performed by the adoption of a set of theories/technologies upon the modelling of the contextual information of nodes. Initially, we rely on

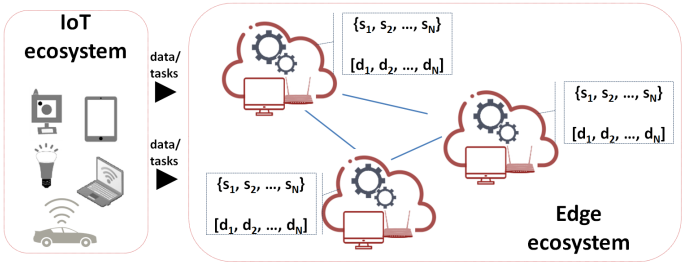

Fig. 1: The envisioned scenario where processing nodes interact with IoT devices and each other.

TABLE II: Nomenclature

Notation	Description
N	Number of services
$\mathscr{S}$	Set of services
s_i	The ith service
$\mathbf{d}^i$	Services demand vector
d_j^i	Demand for the jth service at the ith node
$\mathbf{x}$	Multidimensional data vector
l_t^i	Load of the ith node at t
$\mathscr{T}^i$	The stream of tasks at the ith node
T_t^i	The task reported at t and at the ith node
$\mathscr{C}_t^i$	Characteristics of the task reported at t
$g()$	Utility function adopted to support decision making
$\hat{\alpha}$, $\hat{\beta}$	Required time for local execution and offloading actions
U, $\hat{U}$	Final utilities for local execution and offloading actions

the principles of the Utility Theory and model the expected utility for the available actions when a task is received, i.e., the local execution and the offloading to peers/Cloud. We define the necessary utility functions and deliver the probability of violating the tasks popularity indicator and the threshold of load that secures the tasks' efficient local execution. The study of the probabilistic model requires the adoption of a statistical inference process to conclude the aforementioned probabilities for every observed parameter. Our statistical scheme is built upon a non parametric model to 'speed up' the inference process making us able to estimate the future realizations of the adopted parameters and setup the basis for the appropriate decision making. The demand indicators play a significant role as popular tasks have to be kept locally to maximize the re-use of resources and minimize the response time. For this, we rely on the principles of the order statistics theory and formulate a model that affects the aforementioned utilities based on the popularity of every service. We depict the joint probability of all order statistics before we are in the position to conclude the final decision taken after incorporating the communication cost for the offloading actions. Finally, we perform the decision making focusing on the maximization of the utility as delivered by the previous steps.

B. Utility Functions and Services Demand Management

As noted $\mathbf{d}^i = \left[d_1^i, d_2^i, \ldots, d_N^i\right]$ is adopted to depict the current demand of the available services no matter if they are locally present. $\mathbf{d}^i$ & l_t^i are updated after the arrival of tasks to represent up-to-date information and assist in our envisioned decision making. In the following description, we omit the indexes in the adopted variables to simplify our

notation and focus on the management of a specific service. We define two utility functions for actions (decisions) 1 & 2. We strategically decide to adopt sigmoid functions that are based on the demand for a specific service and depict a proportional management, i.e., a high demand indication will lead to a high value as the outcome of the discussed functions. The sigmoid functions will assist in learning the relationship of the input features and the final output that should be 'smoothly' achieved till a maximum utility value. Our target is to avoid abrupt changes in the utility output no matter the input values. Additionally, the adopted sigmoid functions will keep the utility low if the inputs are below a 'threshold' (depicted by the point where the slope changes) while the opposite stands true when the input is over the aforementioned 'threshold'. The following equations hold true (under the assumption of the corresponding random variables G and $\hat{G}$ defining the envisioned utilities):

$$g = \begin{cases} \frac{\varepsilon}{1+e^{-\gamma d+\delta}} & \text{if } d \text{ is in top-k} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$\hat{g} = \begin{cases} \frac{\hat{\varepsilon}}{1+e^{-\hat{\gamma} d+\hat{\delta}}} & \text{if } l > \hat{\theta} \ \& \ d \text{ is not in top-k} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

g depicts the utility of the local execution of T_t^i and $\hat{g}$ represents the utility of the offloading action. In the above equations, $\varepsilon, \hat{\varepsilon} \in \mathbb{R}^+$ are parameters that depict the maximum utility enjoyed by a node when taking the corresponding decisions while $\gamma, \delta, \hat{\gamma}, \hat{\delta} \in \mathbb{R}^+$ are smoothing parameters. The motivation behind the adoption of the exponential utility functions lies on the freedom we have to incorporate multiple strategies (related to the uncertainty management) in the definition of the utility that a node gains for every realization of the demand indicators. In Figure 2, we plot some example sigmoid utility functions for various realizations of the envisioned smoothing parameters in comparison with a linear utility function. These plots expose the ability of the sigmoid functions to incorporate the uncertainty in the final utility compared to the linear function that can be characterized as 'uncertainty neutral' due to the constant slope. Hence, through the adoption of $\gamma, \delta, \hat{\gamma}, \hat{\delta}$, we can easily define a 'virtual threshold' for services' demand, below which, the final utility is considered as low. In these cases, a very low demand for services does not excuse the local execution of the corresponding tasks, thus, nodes enjoy a limited utility for the specific action. The opposite rationale stands true when the demand is observed to be high. As expected, the utility for the local execution of a task is mainly based on the demand of the required service and if this demand is among the highest in the ranked list, i.e., in the top-k demand realizations observed in the node. $\hat{g}$ is based on the load that the node will have after the reception of T_t^i and the event that the demand is not among the highest observations locally. In that case, the node should decide to offload T_t^i to a peer.

For both decisions, we define the expected utilities U & $\hat{U}$ upon g and $\hat{g}$. We consider the random variable D with its realizations being d values as observed after the updates on $\mathbf{d}^i$. As noted and without loss of generality, we focus on the demand of a specific service. Let $D_{(r)}$ be the random variable depicting that d is the rth smallest among N observations. We assume that $D_{(r)}$ is defined upon the random variable D with a random sample of size N from some distribution, i.e., $D_1, D_2, \ldots, D_N$ with realizations $\mathbf{d} = [d_1, d_2, \ldots, d_N]$. For instance, $D_{(1)} = \min(d_1, d_2, \ldots, d_N)$, $D_{(2)} = 2_{nd}\ \min(d_1, d_2, \ldots, d_N)$ and so on and so forth. Assume that T_j^i 'causes' a demand for a service equal to d. We have to conclude the expected utility for the local execution of T_j^i based on the ranked list to see if d is among the top-k services.

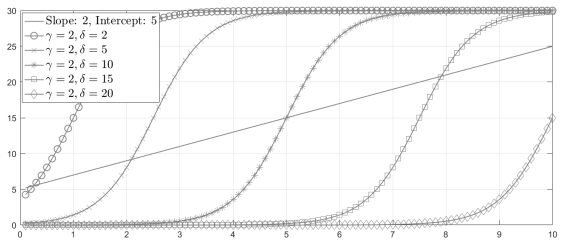

Fig. 2: Example plots of the proposed utility functions in comparison with linear utility realizations.

Proposition IV.1. *The expected utility for the local execution of a task that requires a service with demand d is given by* $\mathbb{E}(G) = \frac{\varepsilon}{1+e^{-\gamma d+\delta}} F_{D_{(N-k)}}(d)$ *where* $F_{D_{(N-k)}}(d)$ *is the cumulative distribution function (cdf) of the variable* $D_{(N-k)}$.

Proof. $D_{(r)}$ depicts the rth smallest observation among N measurements after sorting them in an ascending order. We have to get d among the top-k observations, thus, d should be over the $N-k$ measurements in the ranked list. $D_{(r)}$ has a probability density function (pdf) $f_{D_{(r)}}()$ and a cdf $F_{D_{(r)}}()$. Recall that $F_{D_{(r)}}(d) = P(D_{(r)} \leq d)$, i.e., d is greater than the rth smallest observation in the ranked list. Obviously, for having d at the top-k list (highest values), the probability of the $N-k$th smallest observation should be less than d as depicted by the cdf of the discussed random variable. Based on this probability, we can easily derive the expected utility for Decision 1, i.e., $\mathbb{E}(G) = \frac{\varepsilon}{1+e^{-\gamma d+\delta}} F_{D_{(N-k)}}(d)$. ■

Let L be the random variable depicting the load of the node deciding between Decisions 1 & 2. T_j^i will cause an update on the current load l if the task is going to be kept locally. For deciding the offloading of the task, we have to conclude the expected utility for that action, i.e., to check if l is over a pre-defined threshold $\hat{\theta}$ and see if the required service is not popular (its demand is not in the top-k list).

Proposition IV.2. *The expected utility for the offloading action of a task that requires a service with demand d is given by* $\mathbb{E}(\hat{G}) = \frac{\hat{\varepsilon}}{1+e^{-\hat{\gamma} d+\hat{\delta}}} \left(1 - F_L(\hat{\theta})\right) \left(1 - F_{D_{(N-k)}}(d)\right)$.

Proof. For calculating $\mathbb{E}(\hat{G})$, we have to conclude two probabilities, i.e., the probability that l will be over $\hat{\theta}$ and the probability of d not being in the top-k list. For the former probability, we have: $P(l > \hat{\theta}) = 1 - P(l \leq \hat{\theta}) = 1 - F_L(\hat{\theta})$ with F_L being the cdf of the L distribution. For the latter probability, d is not in the top-k list if it is lower than the $D_{(N-k)}$ realization (no matter the exact position in the ranked list - d

may be first, second, etc but, for sure, below the $N-k$ rank). Hence, we have: $P(d \text{ is not in top-}k \text{ list}) = P\left(D_{(N-k)} > d\right) = 1 - P\left(D_{(N-k)} \leq d\right) = 1 - F_{D_{(N-k)}}(d)$. As both random variables (i.e., L and D) are independent, we, finally, get: $\mathbb{E}\left(\hat{G}\right) = \frac{\hat{\varepsilon}}{1+e^{-\hat{\gamma}d+\hat{\delta}}}\left(1-F_L\left(\hat{\theta}\right)\right)\left(1-F_{D_{(N-k)}}(d)\right)$. ■

C. Statistical Inference over Contextual Data

For inferencing the estimated load and expose its distribution, we rely on the approach discussed in [20]. Load values are continuously monitored and recorded in order to be able to have an estimation about its future realization. Recall that L is the random variable with realizations depicted by l and differs among nodes. Assume that nodes monitor L over the discrete time while storing the most recent L values $l_{t-W+1}, l_{t-W+2}, \ldots, l_t$ in a time window $W > 0$. Based on these values, we can estimate $F_L(\hat{\theta})$ through the estimation of the pdf $f_L(l) = P(L = l)$ of l adopting the widely known non-parametric Kernel Density Estimation (KDE) method [44]. Having W recent samples $\{l_{t-W+j}\}_{k=1}^{j=W}$, the pdf $f_L(l)$ is estimated as:

$$\hat{f}_L(l;W) = \frac{1}{W \cdot h}\sum_{j=1}^{W} K\left(\frac{|l - l_{t-W+j}|}{h}\right) \tag{3}$$

where $h > 0$ is the bandwidth of the symmetric kernel function $K(u)$ (it integrates to unity). One of the most frequent adopted kernel function is the Gaussian, i.e., $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$. We propose the use of an incremental estimation of $\hat{f}_L(l)$, i.e., $\hat{f}_l(l;j)$ for $j = 1, \ldots, W$, is estimated by its previous estimate $\hat{f}_L(l;j-1)$ and the current value l_j, that is, we recursively obtain for $j = 1, \ldots, W$ that: $\hat{f}_L(l;j) = \frac{j-1}{jh}\hat{f}_L(l;j-1) + \frac{1}{jh}K\left(\frac{|l-l_{t-W+j}|}{h}\right)$. Upon capturing l_j at j, $\hat{f}_L(l;j)$ is incrementally estimated by $\hat{f}_L(l;j-1)$, thus, there is no need to store all the previous $j-1$ values for estimating $\hat{f}_L(l;j)$. If we apply the Gaussian function, we obtain an estimation of the cdf $\hat{F}_L(l;W) = \int_{l_{\min}}^{l} \hat{f}_L(u;W)du$ using the aforementioned W values $\{l_{t-W+j}\}_{j=1}^{j=W}$:

$$\hat{F}_L(l;W) = \frac{1}{W}\sum_{j=1}^{W}\frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{l - l_{t-W+j}}{\sqrt{2}}\right)\right) \tag{4}$$

where $\operatorname{erf}(\cdot)$ is the error function of the Gaussian. Hence, at time t, we obtain the estimation of $F_L(\hat{\theta};W)$.

D. Expected Services' Demand Ranking

As mentioned $D_{(r)}$ is defined upon the random variable D with a random sample of size N from some distribution, i.e., $D_1, D_2, \ldots, D_N$ and realizations $\mathbf{d} = [d_1, d_2, \ldots, d_N]$. We consider the scenario where $D_1, D_2, \ldots, D_N$ form a sample that is independent and identically distributed (i.i.d.) which is the easiest case to handle. When the population is not necessarily identically distributed, the joint probability distribution of D_i can be delivered by the Bapat-Beg theorem [45] which depicts the joint probability distribution of order statistics for independent but not necessarily identically distributed random variables w.r.t. their cdf. It should be noticed that the Bapat-Beg algorithm is computationally intractable due to the exponential number of combinations of the adopted distributions. A special case can be met if we assume that the discussed random variables follow only two distributions, thus, the complexity of the method can be reduced as shown by [46].

As exposed by the above analysis, our target is to depict the cdf of the $N-k$ smallest demand indicators compared to d, i.e., the demand of the service required by T_j^i. The $N-k$ element is the 'border' element of the top-k list that significantly affects the decision making of the proposed mechanism. To get the pdf of any order statistic $D_{(r)}$, we have to depict the joint probability of all order statistics and integrate for the specific term. As $D_1, D_2, \ldots, D_N$ are taken to be i.i.d., we consider that $f_D()$ and $F_D()$ are their pdf and cdf, respectively.

Theorem IV.3. *The joint density function of* $D_{(1)}, D_{(2)}, \ldots, D_{(N)}$ *is given by* $f_{1,2,\ldots,N}(d_1, d_2, \ldots, d_N) = N! f(d_1) f(d_2) \ldots f(d_N) \mathbb{I}_{d_1 < d_2 < \ldots < d_N}$.

Proof. See [2]. ■

Theorem IV.4. *The probability of success for* $D_{(r)}$ *when the cdf of* D_i *is* $F_D()$ *is given by* $F_{D_{(r)}}(x) = \sum_{j=r}^{N}\binom{N}{j}(F_D(x))^j (1-F_D(x))^{N-j}$.

Proof. See [47]. ■

Based on Theorem IV.4, we can easily deliver $F_{D_{(N-k)}}$.

Proposition IV.5. *The cdf of the* $N-k$ *order statistic upon services demand values is given by* $F_{D_{(N-k)}}(d) = \sum_{j=N-k}^{N}\binom{N}{j}(F_D(d))^j (1-F_D(d))^{N-j}$.

Proof. It naturally comes from Theorem IV.4 by replacing r with $N-k$. ■

From the above, we can get that the pdf of the $D_{(r)}$ is given by [47]:

$$f_{D_{(r)}}(x) = \frac{N!}{(r-1)!(N-r)!}(F_D(x))^{r-1}(1-F_D(x))^{N-r} f_D(x), x \in \mathbb{R} \tag{5}$$

Hence, the pdf at $N-k$ is given by $f_{D_{(N-k)}}(x) = \frac{N!}{(N-k-1)!k!}(F_D(x))^{N-k-1}(1-F_D(x))^k f_D(x)$. By adopting a specific distribution for D_i, we can conclude the final analytical forms for the above referred formulations and get the final expected utility for Decisions 1 & 2. Trying to define the bottom threshold for the top-k demand list, we can assume that D_i follows a Uniform distribution (we target to a very dynamic environment where demand changes continually) and get the pdf of the $N-k+1$ element (it relies last in the top-k list):

$$f_{D_{N-k+1}}(x) = \frac{N!}{(N-k)!(k-1)!}x^{N-k}(1-x)^{k-1} \tag{6}$$

When the Exponential distribution feeds our demand values, the pdf of the $N-k+1$ element in the ranked list is given by:

$$f_{D_{N-k+1}}(x) = \lambda\frac{N!}{(N-k)!(k-1)!}\left(1-e^{-\lambda x}\right)^{N-k}e^{-\lambda k x} \tag{7}$$

with λ being the rate of the distribution. Figure 3 presents the realization of the pdf for various combinations of k and d

when the Uniform/Exponential distribution is adopted to feed D. At the left plot, we observe a peak when we seek a limited number of top-k demand values ($k \in \{3,5,7\}$) 'transferred' to low d as k is reduced. This means that, naturally, the last element in the top-k list, it, potentially, exhibits a low demand if we incorporate more elements in the list. When we seek at the top-1 element, the distribution of $D_{(N-k+1)}$ relies on high d realizations. The opposite stands for $k = 9$ where the majority of the elements will be included in the top-k list. At the right plot, we adopt $\lambda = 1.5$, i.e., the distribution is in favour of low values.The lower the k is, the lower the pdf becomes leading, in parallel, to a higher boundary for the last $(N-k+1)$ element in the top-k list. For both distributions, an increased ratio $\frac{k}{N}$ will affect the result for the pdf pushing down the corresponding boundary for d.

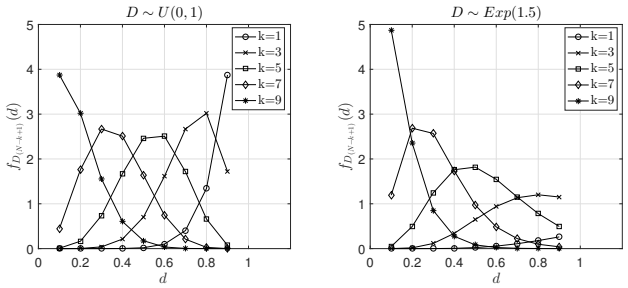

Fig. 3: The effect of k and d on the ranking of the last element in the top-k list ($N = 10$) - left: Uniform distribution; right: Exponential distribution with $\lambda = 1.5$.

E. The Decision Making Process

The proposed inference process is simple, fast and efficient in order to enhance the autonomous behaviour of nodes. We are mainly based on the expected utilities of Decisions 1 & 2 ($\mathbb{E}(G)$, $\mathbb{E}(\hat{G})$) compared to the time required for performing them, i.e., $\hat{\alpha}, \hat{\beta}$. Every utility outcome is updated by the required time to incorporate the communication cost into our inference process. Hence, we feed $\hat{\alpha}, \hat{\beta}$ into an exponential decay function to, subsequently, embed it in the calculation of the final utility. The following equations hold true:

$$U = \mathbb{E}(G) \cdot e^{-\eta \frac{\hat{\alpha}}{\zeta}} \tag{8}$$

$$\hat{U} = \mathbb{E}(\hat{G}) \cdot e^{-\eta \frac{\hat{\beta}}{\zeta}} \tag{9}$$

with η being a smoothing parameter that affects the final utility. We have to notice that ζ denotes the deadline for responding to T_t^i requests and represents the maximum amount of time that can be tolerated by the requestor. The motivation behind the adoption of the exponential decay function is to limit the final utility as the time required for every action approaches or exceeds the deadline set by the corresponding task. Through this model, we 'penalize' the actions that are demanding in terms of time taking into consideration that nodes should return a response before the considered deadline. When $\hat{\alpha}, \hat{\beta}$ exceed ζ, the exponential decay function leads to a very low utility as the required time for delivering the outcome 'violates' the acceptable time interval defined by the task. Naturally, the proposed scheme compares U & $\hat{U}$ and gets the decision that reaches the maximum utility. In case of a tie, our mechanism decides to execute the task locally. Algorithm 1 depicts the proposed decision making process.

Algorithm 1 Local Decision Making

for $t = 1, 2, \ldots$ **do**
  $\langle t, T_t, \mathscr{C}_t \rangle = getTask(\mathscr{T})$;
  Update(**d**);
  Calculate($g, \hat{g}$);
  getExpectedDemandRankings(**d**);
  getExpectedUtilities($\mathbb{E}(G), \mathbb{E}(\hat{G})$);
  Calculate($U, \hat{U}$);
  Decision = max($U, \hat{U}$);
end for

V. Performance Evaluation

A. Experimental Setup & Performance Metrics

We elaborate on the performance of the proposed scheme for revealing its pros and cons. Our Utility aware Model (UM) decides upon the observed demand for services as affected by the execution requirements of the incoming tasks. We, initially, focus on the expected utility for each decision and reveal the parameters that affect it. The intention of our experimentation is to detect the performance of the proposed model at the level of node focusing on the utility that it gains from every decision (i.e., request/migration of a service or the offloading of a task). Two main parameters affect the aforementioned decisions, i.e., the demand for a service and the load of the node. Specifically, the presence/absence of a service locally and the decreased/increased load may heavily affect the final decision. Our experimentation involves both distributions, i.e., Uniform and Exponential, for producing the demand values and various scenarios for realizing k in order to expose how the top-k list affects the final outcomes. The use of the Uniform distribution emulates a heavily dynamic environment where the demand for services could significantly change in consecutive time instances. For example, a low demand (low number of tasks/users request for a service) can be followed by a very high demand (a very high number of tasks/users request for a service). Apparently, such an environment is very difficult to be met in real scenarios (except extreme cases), however, it consists a benchmarking scenario of the proposed model. The Exponential distribution is adopted to emulate a 'smoother' sequential demand realization compared to the previous case. When the Exponential distribution feeds our datasets, we adopt $\lambda \in \{0.5, 1.5\}$ to experiment with a scenario where we observe a low d ($\lambda = 1.5$) and a scenario where the observed measurements exhibit a kind of 'uniformity' in the underlying data ($\lambda = 0.5$). For simplicity in our experimentation, we get $d \in [0,1]$. The demand value can be easily taken in the unity interval, if we adopt a normalization method (e.g., min-max scheme, z-score, etc).

For realizing l, we adopt two datasets found in the respective literature: (i) the Energy Efficiency dataset[1] - D1. From this dataset, we borrow the heating load feature for feeding l. The dataset exhibits a mean equal to 0.517568 while the standard deviation is equal to 0.234111; (ii) the Optical Interconnection Network dataset[2] - D2. From this dataset, we borrow the Processor Utilization feature for feeding l. The dataset exhibits a mean equal to 0.649013 while the standard deviation is equal to 0.194737. Retrieving the aforementioned features, we are able to emulate the load of our nodes and deliver the expected utility for Decision 2. Both, the heating load and the utilization of a processor can easily emulate the load of a node when executing tasks. We also focus on the accuracy of the proposed model in terms of the correct decisions made in every experimentation round. We define the ω metric that depicts the ability of the proposed model to 'follow' the pre-defined strategy, i.e., tasks requiring popular services will be mainly kept for local execution while an increased load and a low popularity will 'fire' an offloading action. The following equation holds true:

$$\omega = \frac{|\{max(U,\hat{U}) \rightarrow \{\text{Decision 1}, \text{Decision 2}\}\}_{correct}|}{|\mathscr{T}|} \cdot 100 \tag{10}$$

where $|\mathscr{T}|$ is the number of tasks processed in our experiments (i.e., 1,000). A correct decision is considered when it is aligned with the strategies adopted to define our utility functions g & $\hat{g}$. If the demand for a service is in the top-k list, Decision 1 should be preferred. Decision 2 should be selected when the load is high (over a pre-defined threshold set at 0.5) and the demand is not at the top-k list. In this set of experiments, we simulate the arrival of a task at every experimentation round (1,000 rounds) and randomly select the service that is affected by the requirements of the incoming task. Then, we expose the utilities for each action and conclude the final decision checking if it is the appropriate one. We also provide a comparative assessment of our UM with: **(i)** a Moving Average (MA) model[3]: The MA processes a time series and accounts for a very short run autocorrelation. It assumes that the next observation is the mean of a number of historical values. In our experiments, we consider the last three measurements of the load as the time series for retrieving the final outcome; **(ii)** a simple Exponential Predictor (EP)[4]. The method is appropriate for estimating data with no clear trend or seasonal patterns; **(iii)** a Polynomial Regression (PR) model adopted by [23]. PR is utilized to estimate the future locations of users, thus, it forecasts services demand. PR is implicitly considering other elements, such as direction/speed of movement, temporal/spatial data locality, and delta between consecutive sensed locations. MA, EP and PR are adopted to estimate the future load of the node compared to our model that takes the decision upon the current measurement. We get the outcomes over 1,000 runs while adopting $N = 10$, $W = 10$, $\hat{\alpha}$, $\hat{\beta}$ are randomly selected in the interval [1,10], $\varepsilon = \hat{\varepsilon} = 10$, $\gamma = \hat{\gamma} = 2$, $\delta = \hat{\delta} = 5$, $\eta = 2$, $\hat{\theta} = 0.5$. Finally, we consider a maximum deadline of 10 time units to deliver the final outcome to the requestor.

[1]https://archive.ics.uci.edu/ml/datasets/Energy+efficiency

[2]https://archive.ics.uci.edu/ml/datasets/Optical+Interconnection+Network+

[3]https://bookdown.org/JakeEsprabens/431-Time-Series/

[4]https://otexts.com/fpp2/

B. Performance & Comparative Assessment

Initially, we report on our evaluation outcomes related to the expected utility for various values of k and d. Our aim is to reveal how these parameters affect the possibility of selecting Decisions 1 or 2 as the appropriate actions to meet the requirements of tasks being aligned with the resources constraints of nodes. Figure 4 presents our performance results when D is fed by the Uniform distribution and we focus on Decision 1. We observe an increasing expected utility no matter k. $\mathbb{E}(G)$ 'follows' the possible increasing trend of d. In other words, a high demand will, naturally, lead to a high utility for the local execution of the corresponding task; a behaviour that is fully aligned with our pre-defined strategy, i.e., tasks that require popular services could be locally executed, thus, to benefit from the potential re-use of already utilized resources. If we focus on the effects of k on the expected utility, we can easily discern that an increased k will lead to an increased $\mathbb{E}(G)$. This is due to the increased possibility of having the requested services in the top-k list and prefer to keep the corresponding task locally re-using the service even if the node has to get it from a peer or Cloud. Naturally, the maximum utility is retrieved when the maximum demand is observed for the specific service(s).

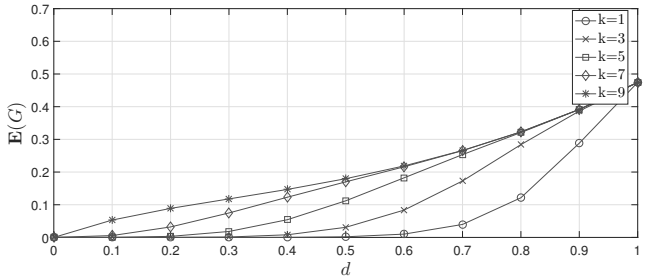

Fig. 4: The expected utility for Decision 1 - Uniform distribution.

A similar trend is observed when we adopt the Exponential distribution to feed D and its realizations d (Figure 5). At the left of the Figure, we adopt near to 'uniform' demand values while at the right, we rely on low demand values. Again, $\mathbb{E}(G)$ is increasing in parallel with d, however, k affects the trend and the maximum utility. We can easily see that the higher the k is the higher the utility becomes in combination with a high demand. In this experimental scenario, especially in the case where we get $\lambda = 0.5$, the utility becomes very low for $k \in \{1,3\}$. Evidently, a limited top-k list increases the possibility of having a service out of it making more preferable the decision of the offloading action. The highest utility is achieved when $k \rightarrow 9$ which means that the vast majority of services will be part of the top-k list (recall that, in our experiments, we adopt $N = 10$).

In Figure 6, we show our results related to the expected utility of the second decision, i.e., the offloading action, when

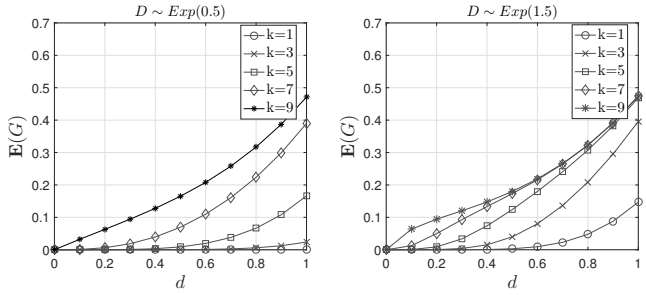

Fig. 5: The expected utility for Decision 1 - Exponential distribution.

the Uniform distribution feeds the random variable D. In this set of experiments, the current load of the node taking the final decision affects the final outcome. Interestingly, we observe that the adopted dataset (D1 or D2) does not mainly affect the final outcome. In addition, the lower the k is the higher the utility becomes. Naturally, a low k (e.g., $k = 1$) will led to many services being out of the top-k list and, in combination with a potential high load, will make Decision 2 more preferable. Another interesting observation is related to the peak of the utility that is affected by d. A low k will lead this peak close to the maximum d, i.e., only a very high demand will make Decision 2 to be not profitable for the node. This peak is reduced as the k increases as more services could be present in the top-k list making profitable Decision 1 instead of Decision 2. We have to notice that the depicted values are combined with costs $\hat{\alpha}$ & $\hat{\beta}$ before we conclude the final utility and support the envisioned decision making.

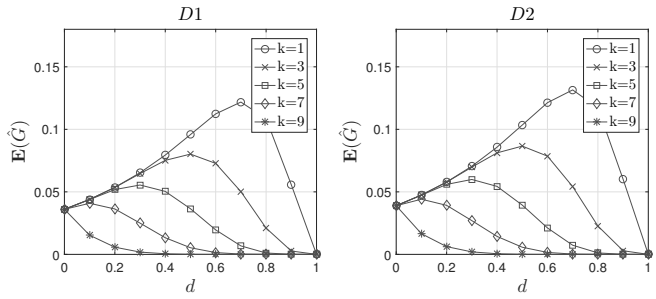

Fig. 6: The expected utility for Decision 2 - Uniform distribution.

In Figures 7 & 8, we see our performance evaluation results related to the expected utility of Decision 2 and the adoption of the Exponential distribution. Again, the adopted dataset for feeding the load of the node does not play a significant role in the final outcome. Similarly to the previous experimental scenario, we observe that a low k will lead to high $\mathbb{E}(\hat{G})$. However, for a low k, $\mathbb{E}(G)$ keeps to be increased no matter d which means that the underlying observed demand measurements when they approach a near 'uniform' scenario ($\lambda = 0.5$) could increase the utility. In that case, as an increased utility could be observed for Decision 1 when $d \to 1$, our model will be in favour of the local execution. In the scenario where $\lambda = 1.5$ (i.e., the distribution delivers low values), the utility seems to be reduced for a high d (except the case where $k = 1$). In general, the utility for the Decision 2 is observed to be high when $\lambda = 0.5$ compared to the scenario when $\lambda = 1.5$.

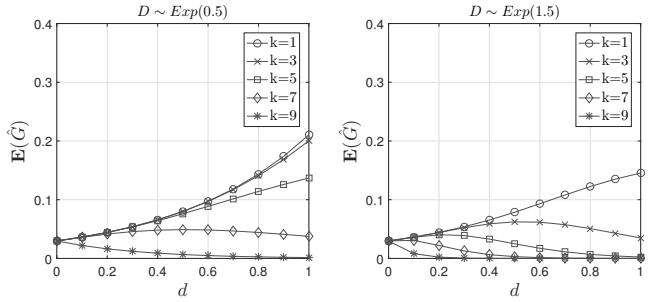

Fig. 7: The expected utility for Decision 2 - Exponential distribution; Outcomes for D1.

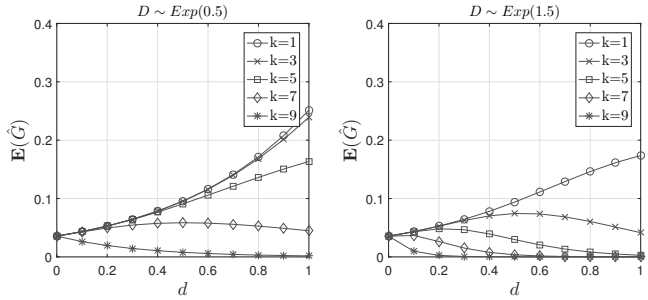

Fig. 8: The expected utility for Decision 2 - Exponential distribution; Outcomes for D2.

In Figures 9 & 10, we present the performance outcomes for the ω metric. The proposed model manages to achieve the optimal results when $k \geq 5$ resulting $\omega = 100$. This stands true no matter the adopted distribution and the dataset utilized to feed the load l. Evidently, we observe a very high number of correct decisions fully aligned with the pre-defined strategy. It should be noticed that, in this set of experiments, we adopt $\lambda = 1.5$, i.e., the Exponential distribution gives us low values for d. The performance of our scheme increases as k increases as well. It is natural to have the best performance when too many services are present at the top-k list, thus, the majority of the experimental scenarios indicate that the Decision 1 should be preferable. However, this does not exclude cases where the load is close to the maximum value and tasks should be offloaded. The worst performance is experienced when $k = 1$ and the local execution is preferable only when the task requests a service that exhibits the highest demand among all services. Concerning the load datasets, it seems that D2 leads to better performance than D1 mainly because D2 exhibits the highest mean, thus, load realizations are relatively higher than D1.

In Tables III & IV, we show the outcomes of the comparative assessment between UM, MA, EP and PR for the ω metric. We observe that UM outperforms the remaining models in the vast majority of the adopted experimental scenarios. MA, EP and PR are affected by the utilized methodology for delivering the estimation of the load that may be concluded in

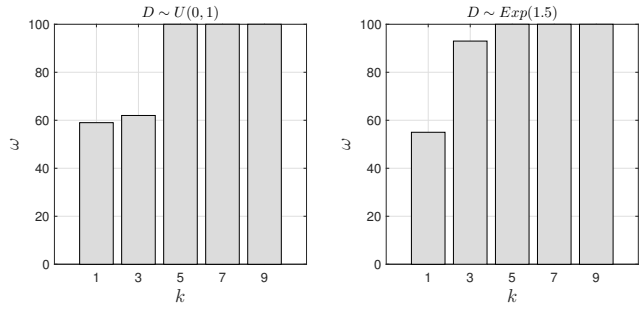

Fig. 9: The accuracy of the proposed model when D1 feeds the dataset of the load of nodes.

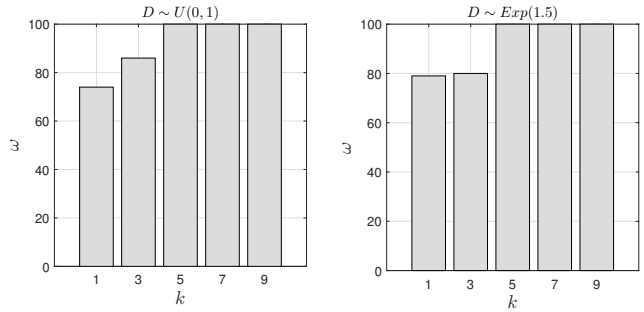

Fig. 10: The accuracy of the proposed model when D2 feeds the dataset of the load of nodes.

low values when multiple historical measurements are taken into consideration. PR achieves the best performance when D2 is combined with the Uniform distribution to feed our parameters and $k \geq 3$. A low l will have a negative impact in the utility delivery for Decision 2. This is more intense in the experimentation with the EP and PR (especially when the Exponential distribution feeds the random variable depicting the demand for services). Recall that in EP the last three load measurements are exponentially combined adopting a smoothing factor equal to 0.5 while in PR the regressions coefficients are retrieved upon past load observations fitting a polynomial. . The UM seems to have better performance when we rely on D1 and MA outperforms when we utilize D2 and $k \leq 3$. As a future extension of our work we consider the incorporation of a mechanism that will manage and estimate the future load of nodes before it feeds it into the decision making model.

TABLE III: Comparative results for D1.

	$D \sim U(0,1)$				$D \sim Exp(1.5)$			
k	UM	MA	EP	PR	UM	MA	EP	PR
1	59	43	31	36	55	74	55	28
3	62	61	44	45	93	54	36	76
5	100	39	26	94	100	64	45	100
7	100	100	100	100	100	100	100	100
9	100	100	100	100	100	100	100	100

VI. Conclusions & Future Work

Services and data management at the edge of the network involve interactions performed either in a vertical or in a horizontal manner. Vertical interactions are related to data collection by the IoT devices and their transfer in an upwards mode to the edge infrastructure while horizontal interactions are based on collaborative activities that can be realized between edge nodes. In this paper, we propose a model that is capable of selecting the appropriate line of actions when tasks arrive at edge nodes. We focus on the demand for services imposed by the requirements of the incoming tasks considering it as the basis for the envisioned decision making. Our scheme decides upon the local execution of tasks, thus, the incorporation of the necessary services (if absent) or their offloading to peers/Cloud. We strategically consider that popular services should be locally incorporated at edge nodes to be able to reuse resources and limit the time required to provide the final responses. The proposed approach combines a statistical inference model and the principles of the utility theory. Our evaluation is performed upon multiple experimental scenarios and real datasets while explaining the pros and cons of our scheme and reveal its characteristics. We show that our mechanism is capable of applying the envisioned strategy while a comparative assessment exposes its ability to get the appropriate decisions for every evaluation scenario. Our future plan is to enhance the proposed model and incorporate more parameters into the decision making mechanism, e.g., study the effect of the network dynamic performance on the decision making.

TABLE IV: Comparative results for D2.

	$D \sim U(0,1)$				$D \sim Exp(1.5)$			
k	UM	MA	EP	PR	UM	MA	EP	PR
1	74	92	74	46	79	92	68	21
3	86	89	72	100	80	95	76	86
5	100	89	73	100	100	89	71	78
7	100	100	100	100	100	92	81	64
9	100	100	100	100	100	91	74	100

References

[1] Najam, S., et al., 'The Role of Edge Computing in Internet of Things', IEEE Communications Magazine, 2018.

[2] Dasgupta, A., 'Asymptotic Theory of Statistics and Probability', Springer, 2008.

[3] Fishburn, P., 'Utility Theory', Management Science, 14(5), 1968.

[4] Kolomvatsos, K., Anagnostopoulos, A., 'Multi-criteria Optimal Task Allocation at the Edge', Future Generation Computer Systems, 93:358–372, 2019.

[5] Soula, M., Karanika, A., Kolomvatsos, K., Anagnostopoulos, C., Stamoulis, G., 'Intelligent Tasks Allocation at the Edge based on Machine Learning and Bio-Inspired Algorithms', Evolving Systems, Springer, 2021.

[6] Karanika, A., Oikonomou, P., Kolomvatsos, K., Loukopoulos, T., 'A Demand-driven, Proactive Tasks Management Model at the Edge', IEEE FUZZ-IEEE, 2020.

[7] Al-Ansi, A., et al., 'Survey on Intelligence Edge Computing in 6G: Characteristics, Challenges, Potential Use Cases, and Market Drivers', Future Internet, 13, 118, 2021.

[8] Oikonomou, P., Karanika, A., Anagnostopoulos, C., Kolomvatsos, K., 'On the Use of Intelligent Models towards Meeting the Challenges of the Edge Mesh', ACM Computing Surveys, 54, 6, 2021, pp. 1–42.

[9] Jararweh, Y., et al., 'The future of mobile cloud computing: Integrating cloudlets and mobile edge computing', 23rd Int. Conf. Telecommun. (ICT), 2016, pp. 1–5.

[10] Mijuskovic, A. et al., 'Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification', Sensors 2021, 21, 1832.

[11] Premsankar, G., et al., 'Edge computing for the Internet of Things: A case study', IEEE Internet Things, 5, 2018, pp. 1275–1284.

[12] Javaid, S., et l., 'Intelligent resource allocation in residential buildings using consumer to fog to cloud based framework', Energies, 12, 2019, 815.

[13] da Silva, R., da Fonseca, N., 'Resource Allocation Mechanism for a Fog-Cloud Infrastructure', IEEE International Conference on Communications, 2018; pp. 1–6.

[14] Xu, X., et al., 'Dynamic resource allocation for load balancing in fog environment', Wireless Communications and Mobile Computing, 2018, art. id 6421607.

[15] Fan, Q., Ansari, N., 'Towards Workload Balancing in Fog Computing Empowered IoT', IEEE Transactions on Network Science and Engineering, 7(1), 2020, pp. 253–262.

[16] Anagnostopoulos, C., Hadjiefthymiades, S., Kolomvatsos, K., 'Accurate, Dynamic & Distributed Localization of Phenomena for Mobile Sensor Networks', ACM TOSN, 12(2), 2016.

[17] Kolomvatsos, K., et al., 'Data fusion and type-2 fuzzy inference in contextual data stream monitoring', IEEE TSMC:Systems, 47(8), 2016, 1839–1853.

[18] Kolomvatsos, K., et al., 'An efficient environmental monitoring system adopting data fusion, prediction, & fuzzy logic', 6th IISA, 2015.

[19] Kolomvatsos, K., et al., 'Distributed Localized Contextual Event Reasoning under Uncertainty', IEEE Internet of Things Journal, 4(1), 2017, 183-191.

[20] Kolomvatsos, K., et al., 'Proactive & Time-Optimized Data Synopsis Management at the Edge', IEEE TKDE, 2020.

[21] Wang, T., et al., 'Fog-Based Computing and Storage Offloading for Data Synchronization in IoT', IEEE Internet Things, 6, 2019, pp. 4272–4282.

[22] Wang, S., et al., 'A Survey on Service Migration in Mobile Edge Computing', IEEE Access, PP(99), 2018.

[23] Bellavista, P., et al., 'A migration-enhanced edge computing support for mobile devices in hostile environments', 13th IWCMC, 2017, 957—962.

[24] Alam, M., et al., 'Multi-agent and reinforcement learning based code offloading in mobile fog', in ICOIN, 2016, 285-–290.

[25] Wang, S., et al., 'Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach', IEEE Transactions on Mobile Computing, 20(3), 2019, pp. 939—951.

[26] Ouyang, T., et al., 'Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing', IEEE Journal on Selected Areas in Communications, 36(10), 2018, pp. 2333–2345, 2018.

[27] Xu, J., et al., 'Path selection for seamless service migration in vehicular edge computing', IEEE Internet of Things, 7(9), 2020, pp. 9040–9049.

[28] Sun, Y., et al., 'Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks', IEEE Journal on Selected Areas in Communications, 35(11), 2017, pp. 2637–2646.

[29] Wang, S., et al., 'Dynamic service migration in mobile edge computing based on markov decision process', IEEE/ACM Transactions on Networking, 27(3), 2019, pp. 1272—1288.

[30] Ouyang, T., et al., 'Adaptive user-managed service placement for mobile edge computing: An online learning approach', IEEE INFOCOM, 2019, pp. 1468–1476.

[31] Sun, Y., et al., 'Learning-based task offloading for vehicular cloud computing systems', IEEE ICC, 2018, pp. 1–7.

[32] Mosel, E., Vigoda, E., 'Limitations of Markov Chain Monte Carlo Algorithms for Bayesian Inference of Phylogeny', The Annals of Applied Probability, 2006, vol. 16, 4, pp. 2215–2234.

[33] Yin, L., et al., 'Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing', IEEE Transactions on Industrial Informatics, 14, 2018, pp. 4712–4721.

[34] Wu, C.-L., et al., 'Mobility-aware deep reinforcement learning with glimpse mobility prediction in edge computing', IEEE ICC, 2020, pp. 1–7.

[35] Yu, Y., Prasanna, V., 'Energy-Balanced Task Allocation for Collaborative Processing in Wireless Sensor Networks', Mobile Networks and Applications, 10(1-2), 2005, pp. 115–131.

[36] Yang, J., et al., 'Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization', IEEE Sensors Journal, 14(3), 2014, pp. 882–892.

[37] Razavinegad, A., 'Task Allocation In Robot Mobile Wireless Sensor Networks', Int. Journal of Scientific & Technology Research, 3(6), 2014.

[38] Gbenga, D., Ramlan, E., 'Understanding the Limitations of Particle Swarm Algorithm for Dynamic Optimization Tasks: A Survey Towards the Singularity of PSO for Swarm Robotic Applications', ACM Computing Surveys, vol. 49(1), 2017, 8, pp. 1–25.

[39] Hu, X., Xu, B., 'Task Allocation Mechanism Based on Genetic Algorithm in Wireless Sensor Networks', in ICAIC, 2011.

[40] Coltin, B., Veloso, N., 'Mobile Robot Task Allocation in Hybrid Wireless Sensors Networks', Int. Conf. on Intelligent Robots and Systems, 2010.

[41] Voinescu, A., Tudose, D. S., Tapus, N., 'Task Scheduling in Wireless Sensor Networks', 6th ICNS, 2010.

[42] Loftis, J., et al., 'Detecting cumulative watershed effects: the statistical power of pairing', Journal of Hydrology, vol. 251(1–2), 2001, pp. 49–64.

[43] Kolomvatsos, K., Anagnostopoulos, C., 'A Probabilistic Model for Assigning Queries at the Edge', Computing, Springer, 102, 2020, pp. 865–892.

[44] Baszczyńska, A., 'Kernel Estimation of Cumulative Distribution Function of a Random Variable with Bounded Support', Stats Trans. 17(3):541–556, 2016.

[45] Bapat, R., Beg, M., 'Order Statistics for Nonidentically Distributed Variables and Permanents', The Indian Journal of Statistics, 51(1), 1989, pp. 79–93.

[46] Glueck, A., et al., 'Fast computation by block permanents of cumulative distribution functions of order statistics from several populations', Communications in Statistics – Theory and Methods. 37 (18), 2008, pp. 2815–2824.

[47] Casella, G., Berger, R. 'Statistical Inference', 2nd ed., Cengage Learning, 2002.

Kostas Kolomvatsos 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 at the 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. His research interests are in the definition of Intelligent Systems adopting Machine Learning, Computational Intelligence and Soft Computing for Pervasive Computing, Distributed Systems, Internet of Things, Edge Computing and Pervasive Data Science. He is the author of over 120 publications in the aforementioned areas.

Christos Anagnostopoulos Dr Christos Anagnostopoulos is an Associate Professor in the School of Computing Science, University of Glasgow. His expertise is in the areas of large-scale distributed data systems and in-network information processing. He has received funding for his research by the EC/H2020, UK EPSRC and the industry. Dr Anagnostopoulos is an author of over 150 refereed scientific journals/conferences. He is leading the Essence: Pervasive & Distributed Intelligence within the Knowledge and Data Engineering Systems Group (IDA Section). He has held postdoctoral positions at University of Glasgow and University of Athens in the area of mobile and context-aware computing. He holds a BSc, MSc, and PhD in Computing Science, University of Athens (2008). He is an associate fellow of the HEA, member of the ACM and the IEEE.