Proactive, Uncertainty-Driven Queries Management at the Edge

Kostas Kolomvatsos, Christos Anagnostopoulos

Department of Informatics and Telecommunications, University of Thessaly
Lamia, Greece
e-mail: kostasks@uth.gr

School of Computing Science, University of Glasgow
Glasgow, UK
e-mail: Christos.Anagnostopoulos@glasgow.ac.uk

Abstract

Research community has already revealed the challenges of data processing when performed at the Cloud that may affect the performance of any desired application. The main challenge is the increased latency observed when the data should 'travel' to the Cloud from the location they are collected and the waiting time for getting the final response. In an Internet of Things (IoT) scenario, this time could be critical for supporting real time applications. A solution to the discussed problem is the adoption of an Edge Computing (EC) approach where data can be processed close to their collection point. IoT devices could report data to a number of edge nodes that behave as distributed data repositories having the capability of processing them and producing analytics. Analytics should match the requirements of queries defined by end users or applications with the collected data and the characteristics of every edge node. However, when a query is defined, we should identify the appropriate edge node(s) to process it. In this paper, we propose an uncertainty management model to efficiently allocate every incoming query to the available edge nodes. Our scheme adopts the principles of Fuzzy Logic (FL) theory and provides a decision making mechanism for the entity having the responsibility of the envisioned allocations. We combine the proposed uncertainty management scheme with a machine learning model based on a Support Vector Machine (SVM) to enhance the FL reasoning. Our aim is to manage all the hidden aspects of the problem combining two different technologies with different orientations. We also propose a methodology for

the automated generation of the Footprint of Uncertainty (FoU) of membership functions involved in our interval Type-2 FL model. Our experimental evaluation aims at revealing the pros and cons of our mechanism presenting the results of extensive simulations adopting datasets found in the literature and a comparative analysis with other efforts in the domain.

Keywords: Edge computing, Internet of Things, Query allocation, Fuzzy Logic, Uncertainty Management, Interval Type-2 Fuzzy Sets, Automated generation of fuzzy sets

---

1. Introduction

Nowadays, one can observe the advent of the Internet of Things (IoT) that offers a huge infrastructure where autonomous devices can interact with their environment and execute lightweight tasks. In this infrastructure, the network provides the means for the exchange of data taking them from their collection points towards the Cloud for long-term storage and further processing. Processing activities have, usually, the form of queries asking for analytics to create knowledge [52] and support the designed decision making. Frequently, such a mechanism is incorporated into intelligent applications trying to conclude efficient services for end users. However, when relying on Cloud, we face a high average separation between IoT devices and Cloud and increased network latency and jitter [58]. For alleviating the problem of the increased latency, we can perform the processing of queries at the edge infrastructure focusing on an Edge Computing (EC) model [56]. In EC, numerous devices form the network edge like tablets, smart-phones, sensors, gateways or nano data-centers [23]. EC aims at a distributed approach where heterogeneous resources should be interconnected and controlled by various means.

Our scenario involves the execution of queries at the edge of the network where a set of distributed data repositories becomes the basis for the provision of responses. Edge Nodes (ENs) maintain local repositories fed by data collected by the IoT devices connected to them. Data may be replicated for supporting fault tolerant applications. As data are distributed among ENs, it is imperative to define a mechanism that allocates the incoming queries to the appropriate nodes. Such a decision should be made not only based on queries demand for data (query constraints/conditions) but also on various characteristics of ENs themselves. In this paper, we extend our

previous efforts in the same problem [38], [42], [46] focusing on the behaviour of entities that are responsible to allocate queries to the appropriate ENs. We call such entities *Query Controllers* (QCs). We consider that multiple QCs can be present in the Cloud and that a query processor is present to every EN to execute the incoming queries. Every QC has a direct 'connection' with ENs being capable of allocating queries to get responses. Our aim is to provide a decision making mechanism adopted by QCs that will result the appropriate queries allocations in the minimum time. Any decision is made over queries' and ENs' characteristics, thus, we support our decision making mechanisms with methodologies for realizing them. Our modelling process is incorporated into the proposed uncertainty-driven reasoning. We start from the definition of queries constraints and their complexity trying to match them against the status of ENs and their datasets. When the 'annotation' of queries is in place, we support QCs with our uncertainty management scheme to derive the appropriate EN where every incoming query will be allocated. To increase the performance, we incorporate in the uncertainty management scheme, the 'opinion' of a machine learning model for every potential allocation. The goal is to 'aggregate' the output of the machine learning scheme with the remaining parameters to create a powerful mechanism for deciding the desired allocations on the fly.

Motivating example. We assume the Smart Grid (SG) infrastructure where numerous smart meters (IoT devices) can be adopted to record and monitor the energy consumption of consumers. Smart meters can have a two way interaction (through wide-area network protocols) with the energy distribution infrastructure that consists of edge devices (close to smart meters) and the Cloud back end system. Smart meters are capable of collecting energy consumption data transferring them to the Cloud through edge nodes. They report multivariate data (e.g., consumption values, timestamps) being stored at the edge infrastructure to deliver spatio-temporal analytics in limited time. Edge nodes can enable energy utilities or distribution operators with advanced real time monitoring and analytics capabilities over the distributed energy data. Operators, utilities administrators and energy policy makers, at the back end system, may want to instruct queries for generating analytics over the entire network or a part of it. Suppose that the operator wants to get analytics related to energy consumption in a specific interval for a specific area. The discussed interval and the area under consideration will be part of a query that should be allocated only in edge devices that exhibit data in the interval and their location is in the desired area. The

allocation of the query to the entire network is useless taking into consideration that edge nodes with data out of the defined interval and the desired area will spend resources to respond with an empty set. It is better to have a view before hand for the appropriate edge nodes that should respond to the specific query. Operators and administrators may instruct numerous queries upon the collected data, thus, a mechanism that manages them in the most efficient manner and deciding the optimal allocations is necessary. In this scenario, the need of our mechanism is revealed, i.e., our model can facilitate the allocation of the discussed queries to the appropriate node(s) to be aligned with queries requirements and have the final response in the minimum possible time. This motivating example makes us to discern the uncertainty in the decision making incorporated in the mechanism that delivers the final allocations. Such an uncertainty is related to the design of the decision model and the detection of the optimal allocation for each incoming query especially when we consider a high number of conditions. For building the knowledge base and the decision delivery model (i.e., variables to depict inputs and outputs, thresholds for variables, rules, etc), we can rely on Fuzzy Logic (FL) [73] that assists in the definition of the appropriate 'combinations' of multiple parameters (i.e., contextual information) characterizing queries and edge devices while avoiding to adopt crisp thresholds. Additional uncertainty is observed in the design/definition of the discussed model by experts. Some critical questions should be answered during the design phase: What is the appropriate interval where every variable is realized? When a variable is considered that depicts a low/medium/high value? How variables can be efficiently combined to deliver the optimal output (e.g., allocation)? To respond to these questions, we can go a step forward and adopt an interval Type-2 FL System (T2FLS) [49] that is capable of handling the uncertainty present in the design phase. Interval fuzzy sets can be adopted when it is difficult to determine the exact membership values (i.e., the intervals where variables are realized) of the given elements (i.e., inputs and outputs). In Type-2 fuzzy sets, we utilize intervals as membership values, thus, the exact numerical membership degree is a value inside the considered interval [63], [68]. Evidently, through this approach, we are able to cover two levels of uncertainty defining a powerful model for serving numerous queries with the best possible performance.

In our previous efforts, we mainly focus on the adoption of machine learning techniques to learn the appropriate EN for every query. Any decision is made upon queries and ENs characteristics, e.g., we can rely on the 'burden'

that a query adds to ENs, the data required by the query, the load and speed of ENs and so on and so forth. We adopt the contextual information of queries and ENs at the time when the decision should be concluded. Such a decision making, involves uncertainty related to if every decision is the appropriate one based on the current contextual information. The adoption of 'crisp' thresholds and rules based on simple conjunctions are already identified as non efficient [51], making imperative the presence of an uncertainty-driven model. As exposed by the above presented motivating example, for building an efficient uncertainty-driven mechanisms, we can adopt the principles of FL and an interval T2FLS incorporating as inputs the aforementioned characteristics of queries and ENs while resulting the, so called, *Efficiency of Allocation* (EoA). EoA depicts the certainty (or uncertainty) of the optimality of every allocation upon the observed contextual information. The adoption of the T2FLS targets to manage the uncertainty in the design phase, i.e., the definition of memberships functions for input and output parameters. Type-1 fuzzy sets adopt precise two-dimensional membership functions that a user believes that can capture the uncertainty present in the domain of discourse [14] while Type-2 fuzzy sets generalize Type-1 sets adopting three-dimensional fuzzy membership functions [8]. The membership degree of a Type-2 fuzzy set is a fuzzy set itself in the unity interval. The third dimension supports an additional degree of freedom to capture more information about the represented term.

We enhance the proposed T2FLS with an additional input that represents the opinion of an 'expert'. The discussed 'expert' is a machine learning model, i.e., a Support Vector Machine (SVM) model [30]. We combine two different approaches, i.e., the FL with the principles of machine learning (also adopted in our previous efforts) to build and provide a proactive and efficient mechanism for the problem under consideration. FL systems and the SVM models have already combined in the past in other application domains [15]. It is also experimentally proved by the respective literature that the combination of Type-2 based SVM fusion classifiers outperform individual SVM classifiers in most cases [16]. Additionally, experiments show that Type-2 FL controllers perform better than Type-1 based fusion models in general. This motivates us to adopt the output of an SVM model as an input to a Type-2 FL controller and fuse the opinion of the machine learning scheme with the remaining parameters of the adopted scenario. The interesting aspect of our approach is that through the combination of the FL with the SVM model, we can have an FL classification system with good

generalization ability in a high dimensional feature space that is difficult through a 'simple' FL controller. The SVM model is a powerful machine learning approach known to have a good generalization ability. Another significant aspect is that the SVM model can efficiently work on a high (or even infinite) dimensional feature space; a scenario that cannot be covered by a typical FL system.

Apart from that, we advance the state of the art in the FL management and propose a novel methodology for the automated generation of interval Type-2 membership functions. Hence, we are able to provide a *data-aware uncertainty management* model, i.e., the generation of FL membership functions for our fuzzy sets is aligned with the observed data. The following list reports on the contributions of our work while depicting the differences with previous efforts:

- we propose the combination of an uncertainty management technique with a machine learning model for the efficient allocation of analytics queries to a set of ENs;

- we propose a novel technique to conclude the interval Type-2 membership functions for every input and output parameter of our decision making model to provide a data-aware mechanism. We increase the performance of the proposed model having the discussed knowledge bases aligned with the underlying contextual data. We deliver an automated mechanism for realizing the *Footprint of Uncertainty* (FoU) of Type-2 membership functions for each fuzzy set and the definition of membership functions themselves (see Section 5 for more details). This means that our approach does not require the presence of an expert to conclude the FL knowledge base;

- we provide a model that assists QCs in taking allocation decisions aligned with queries' and ENs' contextual information. We aim to deliver the best possible allocation having queries constraints 'matched' against similar data repositories while avoiding the overloading of ENs;

- we evaluate the proposed scheme adopting a large set of simulations and provide numerical results to reveal its performance.

The paper is organized as follows. Section 2 reports on the prior work in the field by presenting important activities related to our problem. Section 3

presents the problem under consideration and some introductory information. Section 4 discusses how to model the incoming queries and the envisioned ENs while Section 5 presents the proposed decision making scheme. In Section 6, we proceed with our experimental evaluation and in Section 7, we conclude our paper by presenting our future research plans.

2. Prior Work

The IoT creates new opportunities for providing novel services to end users. Such services can be offered over numerous devices interacting each other or with their environment. Such interactions aim at collecting data and process them to produce knowledge. In addition, knowledge can be provided through the processing of data that end users may also generate, e.g., tweets, social networking interactions or photos [61]. Knowledge can be exchanged between devices or transferred to upper layers, e.g., Fog or Cloud. Analytics could be the outcome of every processing activity performed locally or at Cloud trying to discover patterns upon data. Various tools for large scale data analytics have been already proposed in the corresponding literature. The majority of them concern batch oriented systems where data are firstly collected, stored and, accordingly, processed [27]. The interested reader can study the performance of batch oriented systems in a set of publications like [2], [20], [33].

Apart from the batch processing, there is the opportunity for streams processing when data are reported by their sources. Stream processing aims to facilitate the online, (near) real time provision of responses in a set of tqueries. As we envision to have the analytics processing in a distributed manner (e.g., in ENs), an important decision is related to tasks/queries allocation and scheduling. It is worth noticing that ENs, usually, have limited computational capabilities, thus, streams processing might be the appropriate approach for delivering analytics in real time. Smart gateways and micro-data centers could be adopted to facilitate the task/query processing [1], [26]. A thorough study on task/query scheduling is performed for Wireless Sensor Networks (WSNs). The main focus of all these efforts is on minimizing the execution time, thus, to deliver the final response with limited latency. To secure the efficient execution of tasks/queries, a number of efforts take into consideration energy constraints [10], [64], a fair energy balance among sensors [21] or a cooperative scheme for exchanging tasks [7]. Other, more advanced, technologies are linear programming [71], swarm intelligence [55],

[70], genetic algorithms [32] or the intelligent management of graph-based schemes [19].

Apart from the collected data, a significant part of the research performed for the delivery of analytics deals with the efficient query management. Actually, we should focus on the management of continuous queries demanding for immediate responses. Obtaining a response in (near) real-time could be very difficult due to limitations defined by the amount of data and the underlying hardware performance. Querying data samples and the provision of progressive analytics is an efficient solution for the described problem [3]. In addition, the parallel execution of queries can increase the performance of applications. Any parallelization activity is realized on top of multiple data partitions that can be the outcome of the distributed reporting of data or a separation process. Separation algorithms can be applied directly on data streams through the adoption of a sliding window approach [9], the definition of sub-streams [72] or taking into consideration multiple types of properties (e.g., balance, structural or adaptation properties) [25]. This way, query operators can be executed in parallel, on the fly, taking into consideration the query semantics or optimization actions (e.g., as presented in [12]).

Uncertainty-driven decision making for queries management has been recently adopted for increasing the performance of the proposed systems. Query optimization is a very difficult task especially in distributed environments. The integration of a query processing subsystem into a distributed database management system enhanced by FL is used for analyzing query response time across fragmentations of global relations [54]. Another research effort, presented in [4], proposes the adoption of FL controllers for resource allocation taking advantage of simple heuristic rules for efficient virtual machines allocation in a distributed setup. FL controllers adopt a knowledge base having the form of a set of fuzzy rules that depict the uncertainty management processing of inputs. An evaluation of the performance of the FL combined with rule-based systems can be found in [62]. The aim of such a combination is to increase the performance, supporting the strength of the FL 'annotation' with a rule-based scheme. The advantages of the FL can be identified in scenarios where information 'inexactness' is present. The authors of [47] try to have an FL model integrated within a database system. The query processing model could be coupled with FL if combined with XML annotations. Apart from the FL, machine learning has been also adopted in queries management. In [69], the authors propose an edge matching technique for comparing the structural resemblances between XML documents.

The presented model builds on top of the edit distance scheme and conventional edge matching based techniques. The final aim is to detect topological edges and repeated substructures before a task/query is assigned to a node. The data present in a node play a significant role in [13]. The authors introduce a novel approach to rewrite queries that are in disjunctive normal form and contain mixture of discrete and continuous attributes. The rewriting process is based on a pre-processing step where data are studied to discover implicit relationships between attributes. The learning model deals with the functional dependencies of data, which are ranked to finally predict their order and rewrite any failing query.

Our past research record involves a set of solutions for the problem of allocating queries to a set of ENs. In [46], we propose a time-optimized scheme for selecting the appropriate ENs through the use of the Odds algorithm. The model mainly focuses on the minimization of the time required for concluding every allocation. In [38], we present a Q-learning scheme to calculate the reward retrieved for every allocation. Our model learns the most efficient allocations mainly based on 'static' information, thus, it should be re-trained at pre-defined intervals to be aligned with the dynamics of the environment. In [43], we extend the work presented in [38] and incorporate into the learning process a load balancing approach comparing it with a clustering model that creates groups of ENs with similar characteristics before it concludes the final allocations. In [39], we propose a probabilistic model for matching queries with datasets and provide a scheme for the conclusion of every allocation upon the expected load of ENs and queries. Such a probabilistic approach is combined with a rewarding mechanism in [41]. The target is to detect the reward that we gain when allocating queries to specific ENs for every pair of characteristics. We extend our previous efforts and focus on the decision related to the offloading of the incoming queries to the appropriate peer nodes. Initially, we adopt a k-Nearest Neighbors (kNN) classifier for deciding the local execution and enhance it with the principles of the Utility Theory for selecting the node to be the hist of the offloaded queries [40]. The demand for each query may affect the decision of the local execution [36]. The strategy is to keep locally popular queries and re-use already delivered outcomes while less popular queries may be offloaded to another peer node. Finally, we propose a scheme for the management of uncertainty in decision making adopting the principles of the FL [37] as an extension of [36]. Going our research a step forward, we try to keep the time required for each allocation minimized, however, taking into consideration that any decision

is uncertain about its optimality. Our research presented in [37] is not exhaustive and does not incorporate a model for covering the uncertainty in the design phase and the coverage of high dimensional feature spaces. The missing contributions in our past research activities that we cover with the current work are: **(i)** Our past efforts do not deal with a combination between FL and a machine learning model for delivering the final allocation; **(ii)** In our previous models, we do not rely on an automated approach for defining the knowledge base of the adopted schemes; **(iii)** Our past models are mostly 'static' meaning that they are applied on top of static values for the envisioned parameters without taking into consideration the continuous update of ENs characteristics.

Comparing our model with other schemes in the respective literature, we can note the following. Our model is based on interval Type-2 FL sets while other efforts adopt Type-1 sets [4], [62], thus, they provide limited uncertainty management. Additional efforts deal with the incorporation of the FL upon databases [47]. Other approaches focus on the similarity of XML documents [69] or a learning process over data dependencies to facilitate the re-formulation of queries when failing [13]. The aforementioned approaches are 'bounded' by the specific characteristics of the underlying system (e.g., database systems, XML management) without paying attention at very dynamic environments like those present in edge computing and IoT. In this paper, we depart from such efforts and go a step forward concerning the management of the uncertainty present in very dynamic environments where 'actors' should update their behavior frequently. We pursue efficiency through the management of multiple parameters to cover all aspects of the dynamics of the environment.

3. Preliminaries

In this section, we report on the envisioned scenario and the problem under consideration providing details about the management of an ecosystem where numerous edge nodes are present. In Table 1, we provide a short description of the parameters adopted throughout the paper.

Table 1: Nomenclature

Parameter	Short Description
$\mathcal{N}$	The set of edge nodes
n_i	The ith edge node
N	The number of edge nodes
DS_i	The dataset available at the ith edge node
$\mathbf{x}$	A multivariate vector reported to edge nodes
$\mathcal{DS}$	The set of the available datasets in the network
$\mathcal{C}$	The set of edge nodes characteristics
l	The load of an edge node
s	The speed of an edge node
Q	The stream of queries
q_t	A query reported at time instance t
CH^q	The characteristics of a query
o	The computational complexity of a query
a	The execution deadline of a query
$\mathbf{w}$	The constraints of a query
$\mathbf{v}$	The context vector defined by edge nodes status and query's characteristics
r_i	The information relevance of a query with the ith dataset
Θ	The set of query complexity classes
θ_i	The ith complexity class
DS_Q	The training dataset for calculating the complexity class of a query
$\mathbf{q}^s$	The vector depicting the membership of a query to the available complexity classes
E	The set of similarity metrics adopted to calculate the complexity class of a query
e_i	The ith similarity metric
ω	Our fuzzy operator adopted to 'fuse' similarity values
SL_{e_i}	The significance of a similarity result
Ω	The operator adopted to 'fuse' ω results
μ	The vector of means of a dataset
σ	The vector of standard deviations of a dataset
$\mathbf{f}_i$	The interval of each dimension as exposed by the corresponding mean and standard deviation
EoA	The efficiency of allocation as delivered by the proposed T2FLS
EoA_{SVM}	The efficiency of allocation as delivered by the adopted SVM model
D_T	The training dataset for the automated generation of fuzzy rules
H	The number of inputs in our fuzzy system
OT	The number of outputs in our fuzzy system
C_i	The ith cluster provided over inputs and output values
c	The centroid of a cluster
ρ	The maximum distance from a cluster centroid
g_L	The lower membership function for a fuzzy set
g_U	The upper membership function for a fuzzy set
ϕ	The Footprint of Uncertainty of a fuzzy set

3.1. The Envisioned Ecosystem

We consider a network setup as presented by Figure 1. We assume a set of ENs, i.e., $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ where every node collects contextual data reported by various IoT devices, or generated through local processing. Local data processing may involve statistical reasoning, inferential analytics, and real-time data management, e.g., estimation of top-k lists over the incoming data streams [45]. The dataset DS_i is available at the ith EN and consists of the basis for any local processing. DS_i is continuously updated as 'fresh'

multivariate vectors arrive through streams, i.e., $\mathbf{x} = [x_1, x_2, \ldots, x_L]^\top \in \mathbb{R}^L$, where L is the number of dimensions (contextual attributes). As multiple end devices may report vectorial data to multiple ENs, replicates among datasets DS_i and DS_j, with $i \neq j$ may be present. The management of potential replicas is beyond the scope of this paper. All datasets form the set $\mathcal{DS} = \{DS_1, \ldots, DS_N\}$. At every EN, we assume that a query processor is responsible to receive a stream of analytics queries (e.g., estimating the regression plane among contextual variables within a time frame), allocated at the EN, execute such queries over DS_i and send the results back to the requestor. A queue is present at each EN where queries are placed and wait for execution. This queue can handle a maximum number of queries; without loss of generality, we consider that queues adopted in ENs have the same length. In addition, we assume the set $\mathcal{C}_i = \{c_{i,1}, c_{i,2}, \ldots, c_{i,m}\}$ depicting ENs' characteristics. For instance, $\mathcal{C}_i = \{l, s\}$ with l representing the current load and s depicting the speed of the corresponding EN. l can be easily estimated through the current number of queries waiting in the corresponding queue, while s indicates the throughput of an EN, i.e., the number of queries responded in a given time unit. The discussed characteristics consist of a finite set of contextual attributes of each node based on the requirements of the application domain. These attributes can de defined in the application design phase and depict information that can be easily retrieved during the functioning of EC nodes. For instance, the load can be calculated as the number of queries present in the corresponding queue compared to the maximum queue size, the speed of processing can be represented by the number of queries executed in a time unit and so on and so forth. The final list is concluded upon the strategy we want to adopt and the information we take into consideration when deciding every allocation. Obviously, all ENs participating in the proposed matching process should expose the same contextual information to avoid problems related with their heterogeneity. At this point, our model assumes homogeneity of EC nodes only in terms of the recorded contextual information and not in terms of their resources, hardware, capabilities, etc. It becomes obvious that the computational capabilities of EC nodes affect the retrieved contextual attributes (e.g., the processing speed).

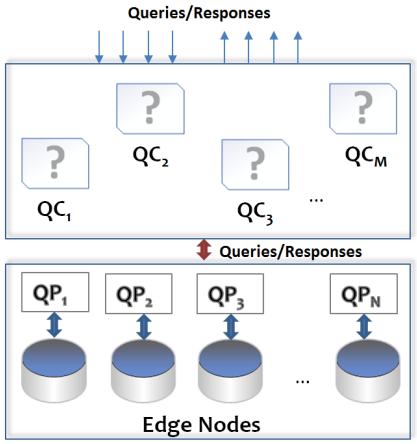


Figure 1: The connection of query controllers and edge nodes.

ENs have direct interactions with entities located at the Fog/Cloud; these entities are QCs which are responsible to serve the incoming query streams. QCs provide the necessary 'interface' for end users or applications where the desired queries can be placed. The interaction between QCs and ENs is concluded by the provision of responses corresponding to the desired query based on the aggregated outcomes retrieved by the invoked ENs. These ENs are decided based on our proposed scheme and could be an individual EN or a sub-set of the available ENs. The selected ENs are those that, at the specific time when the query is issued, exhibit contextual information that will facilitate its *efficient execution* and return the result in the minimum turn around time. QCs, through their continuous interaction with the ENs, can maintain historical performance data as well as statistics of data present in each EN. Based on this context, QCs obtain a holistic overview on the current status of ENs and data that they are stored locally in the ENs. We envision that QCs are software components (possibly part of a platform/module) targeting to realize the proposed reasoning process for the management of the incoming queries. Their capabilities are related to the adoption of the appropriate interfaces to communicate with other components (e.g., a component that receives the incoming queries or a component that sends a query to the appropriate EC node through the use of the available network interfaces) and the appropriate data structures to collect, store and process all the information related to the execution of queries and the performance of the available EC nodes. The vision of having QCs as software components assists in their extendibility by incorporating additional functionalities to

expand their capabilities while being aligned with the requirements of the desired application.

We consider that the incoming queries are reported through a stream $Q = \{q_1, q_2, \dots q_j\}$. At time instance $t \in \mathbb{T} = \{1, 2, \dots\}$ a query q_t arrives to a QC and belongs to a specific *query class*, with specific characteristics, i.e., $CH^q = \left\{c_1^q, c_2^q, \dots, c_{|C^q|}^q\right\}$. For instance, $CH^q = \{o, a\}$ where o stands for the computational complexity and a stands for the deadline for delivering the final result.

3.2. Problem Definition

When q_t arrives in a QC, we should take into consideration q_t's and ENs current contextual information to take the appropriate decision and find the best possible sub-set of ENs to perform the final allocation. Apart from others, we are interested in q_t's data constraints, i.e., we focus on the WHERE clause. Constraints represent the conditions that should be met when we retrieve data corresponding to the final response. q_t can be seen as a $2m$-multidimensional vector

$$\mathbf{w} = \left[\{min_1, max_1\}, \dots, \{min_m, max_m\}\right]^\top \in \mathbb{R}^{2m} \tag{1}$$

such that $\{min_i, max_i\}$ are the minimum and the maximum values defined in the constraints of the i-th dimension. These constraints should be matched against the data vectors present in every EN where the query is directed for execution. The QC creates N *context vectors*; one for each EN. Context vectors refer to the characteristics of q_t and the current status of ENs having the following form (i depicts the index of the corresponding EN):

$$\mathbf{v}_i = \langle o, r_i, l_i \rangle \tag{2}$$

where o is the expected complexity of q_t (elaborated later), r_i is the *information relevance* of q_t with the dataset DS_i and l_i is the load of the ith EN. r_i is a significant parameter as it depicts the 'matching' between q_t and DS_i in terms of the set of potential results. Actually, the information relevance depicts the 'intersection' between 'what a query asks and what a dataset offers'. When q_t's conditions/requirements do not match the statistics of DS_i, the final response will be an empty set. For instance, if q_t asks for stock values in the interval [10,20] and DS_i hosts stocks data in the interval [0,5], the relevance between them is minimized and the final response will be empty. In

general, context vectors represent the minimum sufficient statistics for both an incoming query and each EN based on which the QC should decide the final allocation.

The aforementioned multivariate 'matching' process between q_t and ENs involves uncertainty i.e., we are not sure if any selection will be the optimal as well as we are not sure about the values of the adopted parameters that should define the optimality of the final allocation. In addition, one can also observe uncertainty in the definition of the thresholds adopted to take any decision related to the final allocation. Actually, we 'fuse' the envisioned inputs feeding them into the proposed FL controller that is responsible to derive the EoA. EoA is delivered for each EN and represents the 'belief' of our system that the allocation of q_t to a specific EN will be efficient. Such a fusion is achieved through a finite set of *Fuzzy Inference Rules* (FIRs). Under the principles of fuzzy inference via FIRs, we propose a T2FLS, which defines the fuzzy knowledge base for an EN, e.g., a set of FIRs like: 'when the complexity of q_t is low and the load of an EN is low then the EoA will be high as well'. When the definition of a membership function involves also uncertainty, experts cannot be certain about the discussed membership degree. In such cases, uncertainty is observed not only on the environment of the problem, e.g., how we classify a context value as 'high' or 'low', but also on the description of the term (e.g., 'high') in a FIR. In an interval Type-2 FLS, membership functions are themselves 'fuzzy', which leads to the definition of FIRs incorporating this uncertainty [50]. This approach seems appropriate in our case as FIRs cannot explicitly reflect the knowledge on whether any allocation will be efficient or not as it is affected by the dynamic nature of the problem. For instance, the QC could take a decision based on obsolete information related to the load and data present in an EN. There is a trade off between (i) the increased reporting period (recall that ENs should report their load and data statistics to QCs) leading to decision making on top of 'fresh' information about the status of ENs; and (ii) the minimization of the network overhead for messaging (in case that the reporting period is high). In the following of this paper, we analytically describe how we handle the trade off and the adopted FL controller.

4. Queries & Edge Nodes

As already mentioned, the first step in our process is to model queries and ENs characteristics. We start from the description of the methodology that

delivers the complexity of a query and, then, we model the status of ENs and their datasets. ENs are characterized by their ability to quickly process the incoming queries as depicted by their speed and their current load. Over all these parameters, we apply our T2FLS to derive the envisioned allocations through the EoA that depicts the efficiency of any potential allocation.

4.1. Queries' Characteristics

The delivery of o for any q_t and its implication to l is the key element for our uncertainty-driven decision making. Various research efforts study the complexity of queries [5], [60], [66]. In this paper, for delivering o, we rely on our previous work presented in [42]. It should be noticed that we consider large datasets collected at the EC nodes being the outcome of the data reporting of a high number of IoT devices. We do not focus on a scenario where datasets are small, i.e., a low number of data vectors are available for processing. In that case, the calculation of the complexity for each query could be omitted as even a very complex query will not burden the selected EC node if the dataset is limited. Figure 2 presents the overall process that consists of two parts: (i) the ensemble similarity scheme; (ii) the fusion of multiple aggregated similarity values. Every incoming query is 'matched' against a set of pre-defined queries classified into a number of complexity classes. In this point, we reproduce a small part of our previous research to have a complete view of our model. We consider that $|\Theta|$ complexity classes are available and every $\theta_i \in \Theta$ is aligned with the complexity performed by the operations of q_t required for producing the final response. For instance, Θ could be a set like the following $\Theta \in \{O(nlogn), O(n), O(n^2)\}$. In any case, the complexity classes can be defined beforehand to represent the 'burden' that queries will add to the selected EN. We assign q_t to a complexity class θ_i based on a classification task proposed for such purposes, i.e., a *Fuzzy Classification Process* (FCP). We adopt a fuzzy approach as it is difficult to find an analytical model to deliver the selection of an individual complexity class. The FCP evaluates the membership of q_t in each of the $|\Theta|$ classes. For instance, q_t could 'belong' to the 1st class by 0.2, to the 2nd by 0.5 and to the 3rd class by 0.3. To train the FCP, we adopt a set of historical executed queries along with their corresponding classes. A future extension of our work is to incorporate into our model the historical performance values, i.e., the past execution time for queries belonging to a specific complexity class. This way, we will be able to combine the outcome of the proposed FCP evaluating queries upon the complexity of the required process (we refer in this as the

theoretical complexity) with the real time requirements delivered upon past executions for queries belonging to each complexity class.

We also adopt a set of similarity techniques for concluding the similarity between q_t and θ_i i.e., to detect their 'matching' depicted by a real number (as discussed in the previous examples). Assume that the training dataset is depicted by DS_Q and its tuples are in the form: $\langle p_k, \theta_k \rangle$, $k \in \{1, 2, \ldots, |DS_Q|\}$. p_k represents q_t's statement along with its complexity class $\theta_k \in \Theta$. An example of training tuples is as follows: DS_Q = { ['Select * from tableA', O(logn)], ['Select * from tableB where x=y and z=v', O(n)], ... }. We have to notice that these examples are indicative and adopted to explain the use of our model (complexities may vary based on the type of data structures utilized to store and access the available data). We, then, adopt a function f that delivers a complexity similarity vector encoding the similarity of q_t with every complexity class in Θ:

$$f(q; D_Q) \rightarrow \mathbf{q}^s \in [0,1]^{|\Theta|} \tag{3}$$

$\mathbf{q}^s$'s components assume values in [0,1] demonstrating the degree of membership of q_t to every $\theta_k \in \Theta$. For instance, a vector $\mathbf{q}^s = [0.2, 0.5, 0.3]^\top$ given $\Theta = \{\theta_1 = O(n \log n), \theta_2 = O(n), \theta_3 = O(n^2)\}$ shows that q_t belongs with 20% to θ_1, 50% to θ_2 and 30% to θ_3.

For calculating $\mathbf{q}^s$, we rely on an ensemble similarity scheme. We evaluate the similarity of q_t with every tuple $\langle p_k, \theta_k \rangle \in DS_Q$. The ensemble scheme adopts a set $E = \left\{e_1, e_2, \ldots, e_{|E|}\right\}$ of similarity metrics. Assume that there are three (3) tuples in the training dataset belonging to the θ_i complexity class. Additionally, we rely on two (2) similarity metrics. We get the following similarities for each of the aforementioned tuples: $< 0.10, 0.20 >$ for tuple 1, $< 0.20, 0.25 >$ for tuple 2 and $< 0.30, 0.15 >$ for tuple 3. It is noted that the first value in the example pairs are the outcome of the first similarity metric and the second is the outcome of the second similarity metric. Upon these outcomes, we have to conclude the final similarity between q_t and θ_i. Formally the '2D aggregation' is calculated as follows: $q_k^s = \Omega(\omega \left\{e_i(q_t, \langle s_k, \theta_k \rangle)\right\}, \forall i, \forall \langle p_k, \theta_k \rangle$. ω realizes the envisioned ensemble similarity scheme while the aggregation operator Ω produces the q_k^s through multiple ω values. Actually, ω depicts the similarity outcomes as explained in the above numerical example while Ω is applied upon all ω values to conclude the final similarity with θ_i. For ω, we consider that every single result (i.e., $e_i(q_t, \langle p_k, \theta_k \rangle)$ represents the membership of q_t to a 'virtual' fuzzy set.

ω is a *fuzzy aggregation operator*, an $|\mathcal{E}|$-place function $\omega : [0,1]^{|\mathcal{E}|} \rightarrow [0,1]$) that takes into consideration the membership to every fuzzy set and returns the final value. In the proposed model, we adopt the Hamacher product [31] producing the final ω as follows:

$$\omega = \frac{\dot{e} \cdot \ddot{e}}{a + (1-a)(\dot{e} + \ddot{e} - \dot{e} \cdot \ddot{e})}$$

where $\dot{e}$ and $\ddot{e}$ are two individual similarity values. The use of the Hamacher operator is motivated by the advantages it offers in fuzzy processing and operations as studied in [22].

The second level of aggregation is performed taking into consideration the top-n similarity values based on their *Significance Level* (SL), i.e., a value depicting if a similarity outcome is 'representative' for many other results. We propose the use of the radius γ and calculate the SL as follows: $SL_{e_i} = \frac{1}{1+e^{-(\delta_1 |d(e_i,e_k) \leq \gamma| - \delta_2)}}, \forall i$, where δ_1 and δ_2 are parameters adopted to smooth the sigmoid function. The final results are sorted in a descending order of the SL and the top-n of them are processed with the Hamacher product to deliver the final ω. The Ω operator builds on top of the ω values produced for each tuple in Q_D classified in θ_k. For their aggregation, we rely on a Quasi-Arithmetic mean, i.e., $q_k^s = \left[\frac{1}{m}\sum_{i=1}^{m} \omega_i^{\alpha}\right]^{\frac{1}{\alpha}}$ where α is a parameter that 'tunes' the function. The adoption of the Quasi-Arithmetic mean is motivated by: (i) its simplicity; (ii) it is less affected by fluctuations; (iii) it does not require the arrangement of data like other measures (e.g., median, mode); and (iv) it is completely based on the observations. After calculating the final values for each θ_k, we get $\mathbf{q^s} = \langle \Omega_1, \Omega_2, \ldots, \Omega_{|\Theta|} \rangle$.

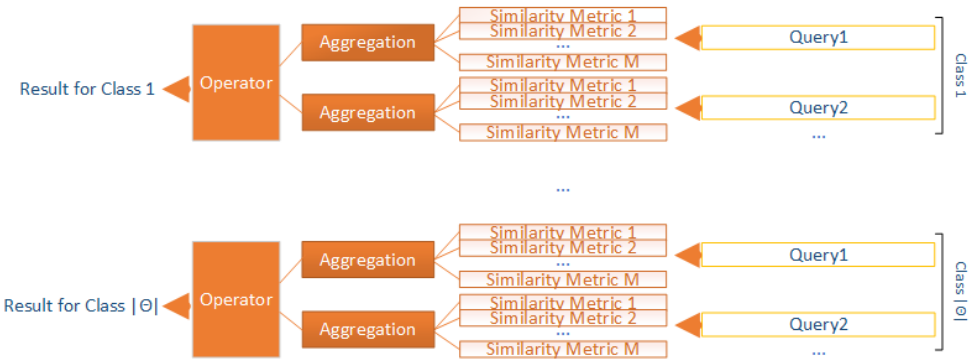


Figure 2: The process for calculating queries complexity.

4.2. Processors Characteristics & Local Datasets

As noted, ENs maintain a queue where the incoming queries are placed and wait for the final processing. The size of the queue and the speed of processing affect the throughput of an EN depicted by the parameter $l_i \in [0, 1]$ (a maximum queue size is adopted for such purposes). When $l_i \to 1$, it means that the EN experiences a high load.

We also propose a distance model to conclude r_i, i.e., the similarity between $\mathbf{w}$ with the data present in ENs. At pre-defined intervals, ENs send to QCs their statistical data represented by two vectors, i.e., the vector of means $\mu = \langle \mu_1, \mu_2, \ldots, \mu_L \rangle$ and the vector containing the standard deviation for each dimension, i.e., $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_L \rangle$. Our problem is to find the overlapping between $\mathbf{w}$ and the intervals represented by the combination of μ and σ. We have $\mu \pm z \cdot \frac{\sigma}{|DS_i|}$ for all the adopted dimensions with $|DS_i|$ depicting the cardinality of the corresponding dataset. z represents the z-value retrieved by the standard normal (Z-) distribution for our desired confidence level. The area in the interval $[-z, z]$ is, approximately, the confidence percentage. For instance, for $z = 1.28$, the area in $[-1.28, 1.28]$ is, approximately, 0.80 (80%). Based on the above, we have to calculate the similarity between $\mathbf{w}$ and N vectors, i.e.,

$$\mathbf{f}_i = \langle \{\mu_{i1} - z \cdot \sigma_{i1}, \mu_{i1} + z \cdot \sigma_{i1}\}, \ldots, \{\mu_{iL} - z \cdot \sigma_{iL}, \mu_{iL} + z \cdot \sigma_{iL}\} \rangle, \forall i$$

For deriving the final r_i, we have to find the final similarity between L intervals, i.e., $\mathbf{w}$ and $\mathbf{f}_i$. Legacy distance/similarity measures (e.g., the Euclidean distance) cannot cope with incorporating cases where $\mathbf{w}$ can be completely contained in $\mathbf{f}_i$. In [29], the interested reader can find a study for calculating the distance over interval data. Based on these metrics, we propose the use of the overlapping metric ψ_k, thus, r_i is defined as follows: $r_i(\mathbf{w}, \mathbf{f}_i) = h(\psi_{ik}), \forall k \in \{1, 2, \ldots, L\}$. In this equation, we propose the use of an aggregation function h responsible to aggregate L distance results, i.e., ψ_{ik}, calculated for each dimension as follows: $\psi_{ik} = 1 - \frac{\|\mathbf{w} \cap \mathbf{f}_i\|}{\min\{\|min_k, \max_k\|, \|\mu_{ik} - \sigma_{ik}, \mu_{ik} + \sigma_{ik}\|\}}$ where

$$w_k \cap f_{ik} = \begin{cases} (\max\{min_k, \mu_{ik} - \sigma_{ik}\}, \min\{\max_k, \mu_{ik} + \sigma_{ik}\}) & for\ \max\{\min_k, \mu_{ik} - \sigma_{ik}\} < \min\{\max_k, \mu_{ik} + \sigma_{ik}\} \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

For h, we adopt the Quasi-arithmetic mean [11], i.e.,

$$r_i = \left(\frac{1}{L} \sum_{k=1}^{L} (\psi_{ik})^{\alpha} \right)^{1-\alpha}$$

.

5. Uncertainty-Driven Queries Allocation

In this section, we describe our T2FLS and the automated process for delivering the adopted fuzzy rules. We also provide details on the automated definition of membership functions for each fuzzy set.

5.1. The FL Scheme

Our FL scheme decides upon the aforementioned set of FIRs that depict our strategy for the allocation of queries. FIRs refer to a non-linear mapping between four inputs: (i) o, (ii) r_i, (iii) l_i, (iv) EoA_{SVM} and one output, i.e., the EoA_i. The EoA_{SVM} represents the result of the proposed SVM model related to the EoA as seen by the adopted machine learning scheme. In our FL controller, we do not rely on a and s as the uncertainty about the matching them is limited. For instance, a simple comparison between them can give us a view if a node can serve a query with a specific deadline depicted by a. The uncertainty is mainly present in the 'combination' of the complexity, the load and the information relevance that are delivered by specific methods as already presented. In addition, there is uncertainty in the combination of the contextual information with the 'opinion' of the 'expert' deciding based on the adopted machine learning model.

The antecedent part of FIRs is a (fuzzy) conjunction of inputs while the consequent part of FIRs is the EoA indicating the belief that the specific allocation will be efficient. The proposed FIRs have the following structure:
IF o is A_{1m} **AND** r_i is A_{2m} **AND** l_i is A_{3m} **AND** EoA_{SVM} is A_{4m}
THEN EoA_i is B_m, where $A_{1m}, A_{2m}, A_{3m}, A_{4m}$ and B_m are membership functions for the m-th FIR mapping o, r_i, l_i, EoA_{SVM} and EoA_i (*values into unity intervals*). The structure of FIRs is the same as in a T1FLS. In a T1FLS, if a linguistic term, e.g., '*high*', was represented by one fuzzy set, we would use one membership function $g(x) \in [0, 1]$ mapping the real value (input) x to a discrete set of pairs $(x_j, g(x_j))$, e.g., $\{(0, 0); (0.25, 0.1); (0.5, 0.75); (1, 1)\}$, where $(0.25, 0.1)$ means that $x = 0.25$ has a membership degree $g(x) = 0.1$. In a T2FLS, $A_{1m}, A_{2m}, A_{3m}, A_{4m}$ and B_m are represented by two membership functions corresponding to *lower* and *upper* bounds [49]. For instance, the term '*high*', unlike in a T1FLS, whose membership for x is a number $g(x)$, is represented by two membership functions. Hence, x is assigned to an interval $[g_L(x), g_U(x)]$ corresponding to a lower and an upper membership function g_L and g_U, respectively. The interval areas $[g_L(x_j), g_U(x_j)]$ for each x_j reflect the uncertainty in defining the term, e.g., '*high*', useful to determine

the exact membership function for each term. This is the above referred FoU. Obviously, if $g_L(x) = g_U(x), \forall x$, we obtain a fuzzy set in a T1FLS. The interested reader could refer in [49] for more information on reasoning under interval Type-2 FIRs. For inputs and the output, we consider an automated membership functions generation model (see next section for more details).

FIRs incorporate the human knowledge on the inference process. The QC, upon receiving q_t, injects the contextual vector into the T2FLS and adopts the following steps: **(Step 1)** calculation of the interval (based on the membership functions) for each input; **(Step 2)** calculation of the active interval of each FIR; **(Step 3)** performance of 'type reduction' to combine the active interval of each FIR and the corresponding consequent. Step 3 produces the interval of the consequent, and accordingly, the defuzzification phase[1] determines a (crisp) value for the EoA_i. The most common method for 'type reduction' is the *center of sets type reducer* [50], which generates a Type-1 fuzzy set as output, which is, then, converted in a scalar value for the EoA_i after defuzzification. The process is repeated for all the available ENs. Finally, the QC performs the final allocation to the appropriate ENs. For this, it sorts the retrieved results in a descending order of EoA and, then, it allocates q_t to the top-k ENs.

5.2. The SVM Model

In this work, we decide to incorporate the result of a machine learning model into the reasoning process of the T2FLS. The aim is to 'support' the FL reasoning with the 'opinion' of an 'expert' built in a completely different scientific orientation. The machine learning model is separated from the FL scheme providing a separated decision making. SVMs are a promising non-parametric technology for performing binary classification. They model the space by creating a feature space which is a finite-dimensional vector space. Every dimension represents a feature of the particular object. In general, the advantages of SVMs are as follows [6]: (i) though the adoption of a kernel, SVMs can build on parameters that show a non-monotone relation to the final score and the probability of default; (ii) the adopted kernel implicitly contains a non-linear transformation of the parameters and no assumptions about the functional form of the transformation; (iii) they provide an efficient

[1]Defuzzification is the process of producing a quantifiable result in FL, given fuzzy sets and the corresponding membership degrees.

out-of-sample generalization; (iv) if the optimality problem is convex, SVMs will return a unique solution; (v) through the selection of the appropriate kernel (e.g., the Gaussian kernel), SVMs can be powerful in the classification of a new sample on top of the training tuples. SVMs are effective in high dimensional spaces as well as memory efficient. In our model, the SVM scheme seems to be the appropriate technique for 'injecting' into the T2FLS the output of a learning process upon a clear margin of separation between classes (i.e., the optimality of the matching process - decision for allocation or not). The ability of the SVM to handle high dimensional spaces gives the opportunity to 'transfer' this aspect into the FL system. It may be difficult to manage high dimensional spaces in an FL model by incorporating too many inputs. Evidently, we are able to keep the proposed model simple, however, efficient covering both, the learning requirements for multivariate decision making and the uncertainty related to any allocation decision. Our model can be easily expanded by incorporating additional parameters/dimensions only in the SVM model keeping as simple as possible the FL part giving the opportunity to cover more advanced/complex application scenarios.

SVMs create linear separating hyperplanes in high dimensional vector spaces. Suppose data are organized in tuples with every tuple containing the pair $(\langle x_1, x_2, \ldots \rangle, y_i)$ where x_j is the value of the jth dimension and y_i is the corresponding class. In our research, we consider that y_i gets two values, i.e., +1 or -1. When $y_i = +1$ means that the specific context vector should result an 'allocate' action while the opposite stands when $y_i = -1$. The problem is to find the optimal separating hyperplane for the specific training data, i.e., the *Maximum Marginal Hyperplane* (MMH) [28]. The associated margin provides the highest possible separation between the envisioned classes. The separating hyperplane can be written as $\mathbf{W} \cdot \mathbf{X} + b = 0$ where $\mathbf{W} = \langle w_1, w_2, \ldots \rangle$ is a weight vector, $\mathbf{X}$ is the data vector and b is a bias. Weights can be adjusted to have the hyperplanes defining the sides of the margin between the points of the first class and line separating the two hyperplanes. When the training process is finished, the SVM is ready to be adopted for classification purposes. Based on the Lagrangian formulations, we get:

$$d(\mathbf{X}^T) = \sum_{i=1}^{p} y_i \alpha_i \mathbf{X}_i \mathbf{X}^T + b_0 \tag{5}$$

where $\mathbf{X}^T$ is a test tuple, α_i and b_0 are numeric parameters delivered by the

optimization process and p is the number of support vectors. $\mathbf{X}^T$ is fed into the SVM and we check where it falls concerning the delivered hyperplanes as identified by the sign of the outcome.

The proposed SVM model is automatically generated upon a training dataset where inputs are o, a, r_i, l_i, s_i and the output is EoA. In the SVM, we adopt more input parameters, as we want to have the 'expert' adopting the learning scheme to take decisions on top of the entire contextual information about queries and nodes. This outcome is incorporated in the uncertainty management mechanism as explained above. Instead of combining the results of the two approaches based on a specific mathematical formulation, we choose to get the SVM result as input to the T2FLS. This means that we try to avoid having the classification error affecting our calculations, i.e., there is uncertainty about the final SVM value especially when noise is applied in the dataset [35].

5.3. Automated Generation of Fuzzy Rules

In this work, we propose an automated generation of membership functions for the adopted fuzzy sets to limit the uncertainty in the their definition. The interested reader can refer in [59] for more details related to the automatic generation of Type-1 or interval Type-2 membership functions. In general, the following techniques can be adopted for generating Type-2 membership functions: (i) heuristics; (ii) histograms; (iii) a clustering algorithm; (iv) genetic approaches. A heuristic is to find the membership function and then to produce the lower bound through a multiplication with a constant in the unity interval. Histograms can be adopted to deliver the distribution of the data and then these are then smoothed and normalized to obtain the minimum degree polynomial function possible. The number of fuzzy sets and their heights are determined by this polynomial function. A clustering algorithm can utilize two equations to define the upper and lower membership functions, rather than the single equation used in the standard method. Genetic approaches try to, firstly, find out the FoU which is the same for all inputs. Afterwards, each input could have a different FoU and, finally, is tested with different FoU for its own sets.

For the automated generation of FIRs, we rely on the training dataset D_T. Suppose every tuple in the training dataset has H inputs and OT outputs, i.e., $\langle x_1, x_2, \ldots, x_H, y_1, y_2, \ldots, y_{OT} \rangle$. For each dimension x_i and y_j, we have to generate the appropriate fuzzy set and its membership functions aligned with the tuples present in D_T. We propose the use of a novel methodology

that is based on machine learning and more specifically on clustering. Let us focus on a specific dimension, e.g., x_i; the same approach stands for the remaining input and output variables. For x_i, we apply a clustering process upon $|D_T|$ tuples and deliver the calculated centroids. For having the same number of fuzzy sets and membership functions for inputs and outputs, we consider that k clusters are delivered. Let the delivered clusters be annotated with $C_1, C_2, \ldots, C_k$ and their centroids be $c_1, c_2, \ldots, c_k$. Every fuzzy set corresponds to a cluster, thus, the centroid ($c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$) will depict the point where the fuzzy set's membership function will approach the unity. We adopt such an approach to have the membership functions 'concentrated' to the points where there is the minimum distance with the numerical data belonging to the specific set/cluster. Afterwards, we have to generate g_L and g_U which represent the lower and the upper membership functions of the interval Type-2 fuzzy set. Values in C_j exhibit a specific standard deviation σ adopted to derive g_L. σ shows the average distance of points around centroids, i.e., it quantifies the amount of dispersion of values present in a cluster.

However, some values could be present close to the maximum radius of the cluster which means that they heavily deviate from the remaining. One can say that these observations lie on an abnormal distance from other values in the population, thus, can be characterized as outliers. In [48], the authors refer in two principal approaches for outliers management: (i) outliers accommodation, i.e., developing of a variety of statistical estimations; (ii) identifying outliers and deciding whether they should be retained or rejected. Various efforts regard outliers as 'bad' data and reject them from further processing [17], [34], [44], [57], [67]. In our model, we consider that values present at the maximum distance from centroids, should be incorporated in our calculations as they affect the uncertainty in the definition of membership functions. This is because outliers negatively affect and increase the deviation of the population, thus, this information shows how the data are spread in the cluster and they should be taken into consideration. We consider the radius of each cluster as the mean distance of the top-k values located in the maximum distance from the centroid ρ. Hence, g_U is defined by ρ while g_L is defined by σ. In Figure 3, we present two examples for two widely adopted membership functions, i.e., the triangular and the Gaussian (c is the centroid of a cluster). The grey area depicts the FoU as defined through σ and ρ. FoU will be eliminated when all values in the cluster are

very close to the centroid. In that case, the uncertainty in the definition of the membership function is minimized leading to a Type-1 fuzzy set ($g_U(\cdot)$ approaches $g_L(\cdot)$). When multiple values are located at distance equal to the radius, i.e., we do not have a 'compact' cluster, the uncertainty is high, thus, the FoU area is expanded. This depicts a high uncertainty in the definition of the corresponding membership function as the data do not 'conclude' to a 'compact' statistical representation. The higher the distance of outliers from the centroid is, the higher the uncertainty becomes. Focusing on Figure 3, we can imagine the 'movement' of the outer functions expanded far way from the g_L representation.

We can easily quantify the FoU ϕ as a function of σ and ρ. Actually, FoU covers an area that is defined by the following equation:

$$\phi = \int_{-\infty}^{+\infty} g_U(x)dx - \int_{-\infty}^{+\infty} g_L(x)dx \tag{6}$$

The equation depicting ϕ depends on the type of the adopted membership function.

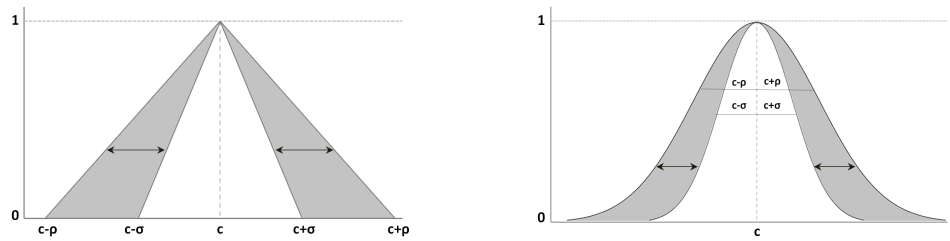
Proposition 1. *When the Triangular membership function is adopted the FoU is quantified by*

$$\phi = \rho - \sigma \tag{7}$$

Proof. Recall that the Triangular membership function is based on three parameters α, β, γ and is defined as follows:

$$f\left(x, \alpha, \beta, \gamma\right) = \begin{cases} 0 & \text{if } x \leq \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \text{if } \alpha \leq x \leq \beta \\ \frac{\gamma-x}{\gamma-\beta} & \text{if } \beta \leq x \leq \gamma \\ 0 & \text{if } x \geq \gamma \end{cases} \tag{8}$$

Both g_U and g_L are defined as the previous equation dictates with: $\alpha = \{c-\rho, c-\sigma\}$, $\beta = c$ and $\gamma = \{c+\rho, c+\sigma\}$ for g_U and g_L, respectively. Applying g_U and g_L in Eq(6) and splitting the integral for being aligned with the aforementioned 'triangular' representations we get: $\phi = \int_{c-\rho}^{c} \frac{x-c+\rho}{\rho}dx - \int_{c-\sigma}^{c} \frac{x-c+\sigma}{\sigma}dx + \int_{c}^{c+\rho} \frac{c+\rho-x}{\rho}dx - \int_{c}^{c+\sigma} \frac{c+\sigma-x}{\sigma}dx$. Actually, ϕ is the area covered by two right (Pythagorean) triangles with sides 1.0 and $\{\rho, \sigma\}$ for g_U and g_L, respectively. We finally get that $\phi = 2\frac{\rho}{2} - 2\frac{\sigma}{2} = \rho - \sigma$. □



(a) The FoU in the triangular membership function. (b) The FoU in the Gaussian membership function.

Figure 3: Modelling the FoU in various membership functions.

Proposition 2. *When the Gaussian membership function is adopted, the FoU is quantified by*

$$\phi = -\frac{\sqrt{\pi}\left(\left(\operatorname{erf}\left(\frac{v_{max}-c}{\sqrt{2}\sigma}\right)+\operatorname{erf}\left(\frac{c}{\sqrt{2}\sigma}\right)\right)\sigma-\left(\operatorname{erf}\left(\frac{v_{max}-c}{\sqrt{2}\rho}\right)+\operatorname{erf}\left(\frac{c}{\sqrt{2}\rho}\right)\right)\rho\right)}{\sqrt{2}} \quad (9)$$

Proof. The Gaussian membership function is based on two parameters μ, s, i.e., the mean and the standard deviation of the sample. It is defined as follows: $f(x,\mu,s) = e^{\frac{-(x-\mu)^2}{2s^2}}$. Hence, for g_U and g_L, we get: $g_U(x,c,\rho) = e^{\frac{-(x-c)^2}{2\rho^2}}$; $g_U(x,c,\sigma) = e^{\frac{-(x-c)^2}{2\sigma^2}}$. We also consider that the variable x has a maximum value equal to v_{max}. Applying both functions into Eq(6) and solving the integrals, we get:

$$\phi = -\frac{\sqrt{\pi}\left(\left(\operatorname{erf}\left(\frac{v_{max}-c}{\sqrt{2}\sigma}\right)+\operatorname{erf}\left(\frac{c}{\sqrt{2}\sigma}\right)\right)\sigma-\left(\operatorname{erf}\left(\frac{v_{max}-c}{\sqrt{2}\rho}\right)+\operatorname{erf}\left(\frac{c}{\sqrt{2}\rho}\right)\right)\rho\right)}{\sqrt{2}}$$

where erf is the error function of the Gaussian. □

It should be noted that similar calculations can be provided for other types of membership functions to get the quantification of FoU and the uncertainty as depicted by the data present in the training dataset. In the first place of our future research plans is the study of the core parameters affecting the uncertainty in membership functions not only for univariate but also for multivariate data.

6. Experimental Evaluation

For revealing the performance of the proposed model, we present our outcomes retrieved by a large set of simulations adopting multiple traces.

Moreover, we provide a comparative assessment with other efforts found in the relevant literature.

6.1. Performance Metrics, Datasets and Simulation Setup

Our simulator is a custom software written in Java where the main 'entities' of our model are represented by a set of classes (the simulator is provided in the Website of the Intelligent Pervasive Systems (iPRISM) research group[2]). In this simulator, we perform the execution of 1,000 queries per experiment and retrieve results for a number of performance metrics. The adopted simulator is very simple and 'produces' queries and nodes characteristics upon four (4) datasets. Two of them are found in an open repository provided and adopted by other researchers in various applications domains. All datasets are governed by a number of different data distributions covering a wide set of experimental scenarios. For instance, the use of the Uniform distribution (see below) simulates a very dynamic environment where data dynamically change and cover the entire interval where our parameters are realized. The reason behind the adoption of many and different datasets is to avoid having the simulator being affected by biases that may negatively impact all the model participating in our comparative analysis. We have to notice that we compare the proposed model not only with our previous efforts to expose the new advances upon our previous outcomes but also with other models found in the respective literature. Our aim is, initially, to identify the time required for concluding the envisioned allocations as well as the 'optimality' of each allocation. Starting from the required time for each allocation, we define R_T, i.e.,

$$R_T = \frac{|Q|}{\sum_{i=1}^{|Q|} T_i^c} \cdot 1,000 \tag{10}$$

In Eq(10), T_i^c is the conclusion time (in ms) for the ith query and $|Q|$ is the cardinality of the set of the incoming queries. We measure R_T as the number of queries allocated in a time unit, i.e., a second. R_T is measured just after the reception of a query and depicts the time required for getting the final node where the query will be allocated. In our experiments, we consider that a single EN is selected for the desired allocation and not a sub-set of nodes. It becomes obvious that R_T depicts the throughput of the QC, thus, we want

[2]https://iprism.eu/presentations and datasets.html

to have $R_T \rightarrow \infty$. A value $R_T \rightarrow 0$ depicts the worst case performance, i.e., the time devoted to get a result from the model is high. For presenting the 'optimality' of the allocations, we rely on the load, speed and information relevance of the selected node. Initially, we record l^* (number of queries waiting in the corresponding queue compared to the maximum queue size) , s^* (number of processed queries per time unit, i.e., per second) and r^* (similarity level between queries and datasets) , where the $*$ indication represents the specific characteristics of the selected node. Obviously, we want $l^* \rightarrow 0$, $s^* \rightarrow s_{max}$ (where s_{max} is a maximum speed for the discussed nodes) and $r^* \rightarrow 1$. In addition, we define the 'optimal' node n^{opt} which represents the 'ideal' node that it exhibits the lowest load, the highest speed and the highest relevance among all nodes present in the group. n^{opt} is adopted to show the 'optimality' of the selected node $n^{selected}$ as delivered by our model. We define the distance of $n^{selected}$ with n^{opt} as follows: $V = \|n^{selected}, n^{opt}\| = \sqrt{(l^{selected} - l^{opt})^2 + (s^{selected} - s^{opt})^2 + (r^{selected} - r^{opt})^2}$ For having the best possible performance, we have to get $V \rightarrow 0$. As V is calculated through the use of the Euclidean distance, it represents if our model reaches the characteristics of n^{opt}. To achieve that, all the adopted characteristics should be close to the optimal value (i.e., n^{opt}). It is worth noticing that in case of ties in the outcomes of the T2FLS, we pay attention on the load of each node getting first the node with the minimum load. For comparison purposes, we provide results for a scenario where in case of ties in the fuzzy outcome, we adopt a random order of the available nodes.

We compare the proposed *Fuzzy Logic Model FLM* with our previous work [43], [46], [42] (Model 1 - M1, Model 2 - M2, Model 3 - M3, respectively). Moreover, we compare our model with the model proposed in [74] where the authors describe a task allocation model (TAM) in heterogeneous parallel and distributed computing systems (the grid computing environment). In their work, they propose an algorithm for supporting the behaviour of the Local Grid Manager (LGM) that allocates tasks to a number of Site Managers (SMs). The LGM calculates the cost for each allocation based on the following formula: $Cost = OR + \frac{TS}{LS}$ where OR is the occupation ratio defined by the total tasks placed at a specific SM compared to the total capacity of the SM, TS is the size of the task and LS is the communication speed between the LGM and an SM. Tasks are allocated in the SM with the lowest cost. The discussed formula is adapted to our case taking l as OR, o as TS and the communication speed equal to a constant small number (in our

case, we consider that the communication speed between QCs and nodes is negligible). We have to notice that in this model as in ours, in case of ties, we rely on the minimum load among the nodes involved in each tie.

We evaluate our model for different realizations of N getting $N \in \{10, 100, 1000\}$. For the type of the incoming queries and the delivery of their complexity class, we rely on the dataset and the methodology presented in [42]. For the remaining parameters, i.e., query vectors, l, s and the data reported in nodes, we rely on a set of traces. Our data are retrieved by the following traces: **(i) Dataset 1**. A synthetic trace based on the Uniform distribution delivering values for query vectors, data present in each node, l and s. In this dataset, all parameters are randomly produced in the interval [0,1] based on the Uniform distribution. We produce a data vector at each simulation step (1,000 in total) ; **(ii) Dataset 2**. A synthetic trace based on the Gaussian distribution delivering values for query vectors, data present in every node, l and s. In this dataset, all parameters are randomly produced in the interval [0,1] based on the Gaussian distribution. We produce a data vector at each simulation step (1,000 in total) ; **(iii) Dataset 3**. The trace reported by [18] retrieved by the UCI Machine Learning Repository[3]. The dataset is created by adopting an extensive set of simulations performed upon the OPNET Modeler. The authors consider message passing between processors. Packets arrive randomly based on a Poisson process and processors retrieve packets from the First-Come-First-Served buffer at a specified rate. Processors extract a packet from the input queue, process it for a period of time and, then, generate the output message. The dataset contains 641 data vectors and ten (10) features. From this dataset, we adopt the processor utilization dimension. The average processor utilization measures the percentage of time that threads are running in a processor, thus, it can be 'aligned' with the load of our edge nodes. We adopt these data to feed the load parameter l. The remaining parameters take values as described for Datasets 1 & 2; **(iv) Dataset 4**. The trace reported by [65] also retrieved by the UCI Machine Learning Repository[4] The dataset is created upon the energy analysis of twelve (12) different building shapes. Buildings differ w.r.t. the glazing area, its distribution and the orientation. The authors rely on 768 building shapes. The final dataset consists of 768 samples and eight (8) features. From this

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

[4]http://archive.ics.uci.edu/ml/datasets/Energy+efficiency?ref=datanews.io

dataset, we borrow values for l and more specifically, we focus on the heating and cooling loads as reported by the dataset. The remaining parameters take values as described for Datasets 1 & 2. For each query, we pay attention on the constraints part and produce the minimum and maximum values for each dimension as described above. In addition, we randomly select the deadline for every query based on the Uniform distribution. The complexity class is delivered as described in [42] and [66] based on TPC-DS and TPC-H benchmarking datasets (http://www.tpc.org/). TPC-DS and TPC-H are decision support benchmarks modelling several generally applicable aspects of a decision support system including queries and data maintenance. TPC-DS involves a set of queries adopted to measure the response time in a single user mode, the throughput in a multi user mode and the data maintenance performance given the characteristics of the underlying infrastructure (e.g., hardware, operating system, configuration). TPC-H consists of a number of business oriented ad-hoc queries and concurrent data modifications selected to depict a broad industry-wide relevance. The discussed queries are defined to be executed upon huge volumes of data exhibiting a high degree of complexity. In our experiments, we classify our evaluation queries in five (5) classes ($|\Theta| = 5$) with an increasing complexity from 1 to 5. Then, we combine the randomly generated values for each dimension and the deadline with a complexity class to proceed with the application of our model.

6.2. Performance Assessment

We report on the performance of our model concerning l^*. In Figure 4, we present the scatter plot of l^* for the adopted datasets. It should be mentioned that, in this plot, we present the sorted list of l^* for each dataset. We observe that for datasets 1 & 4, l^* is very low especially when $N \rightarrow 1,000$. In datasets 2 & 3, l^* is low when $N \rightarrow \{10, 100\}$. In general, the observed load of the selected node is below 0.5 except dataset 2 and $N = 1,000$. The proposed model exhibits a good performance in the synthetic dataset where data are produced randomly without following any pattern (recall that this is represented through the use of the Uniform distribution). In this experimental scenario, we cannot have a 'view' on the trend of the parameters, i.e., at consecutive experimental rounds, values can cover the entire interval where parameters are realized. This means that our scheme can be aligned with the needs of a very dynamic environment where values for each parameter and data present at ENs are continuously updated. Hence, our model can react in unexpected scenarios related to the realized values.

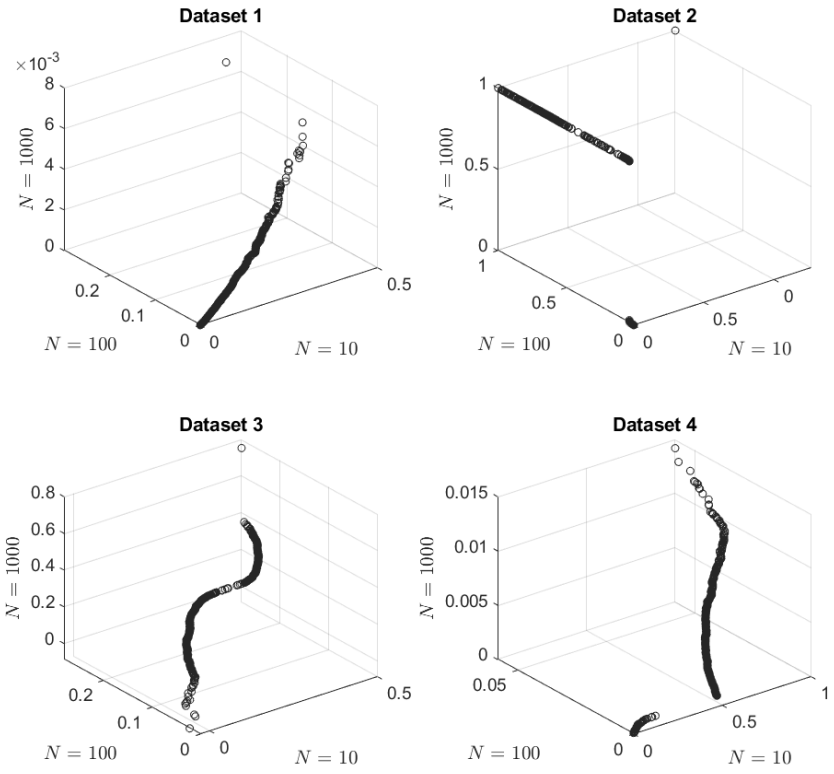


Figure 4: The relationship of the outcomes related to l^* for all datasets.

In Figure 5, we provide the histograms for each experimental scenario and for every dataset. We observe that our mechanism, in the majority of the experimental scenarios, is able to identify and select ENs with a low load targeting to speed up the queries execution process. In any case, recall that our fuzzy scheme tries to 'match' the complexity of a query with the load of ENs. Hence, complex queries demanding increased resources will not burden the selected ENs. In addition, the selection of ENs with a low load means that we provide a 'load balancing' mechanism combined with other parameters like the relevance of the query constraints with the datasets present in ENs. Depending on the dataset, a high number of nodes, i.e., N, leads to better performance especially when the Uniform distribution is adopted. In this case, our scheme has multiple 'choices' to select the node that will host the query, however, it manages to conclude the best possible outcome.

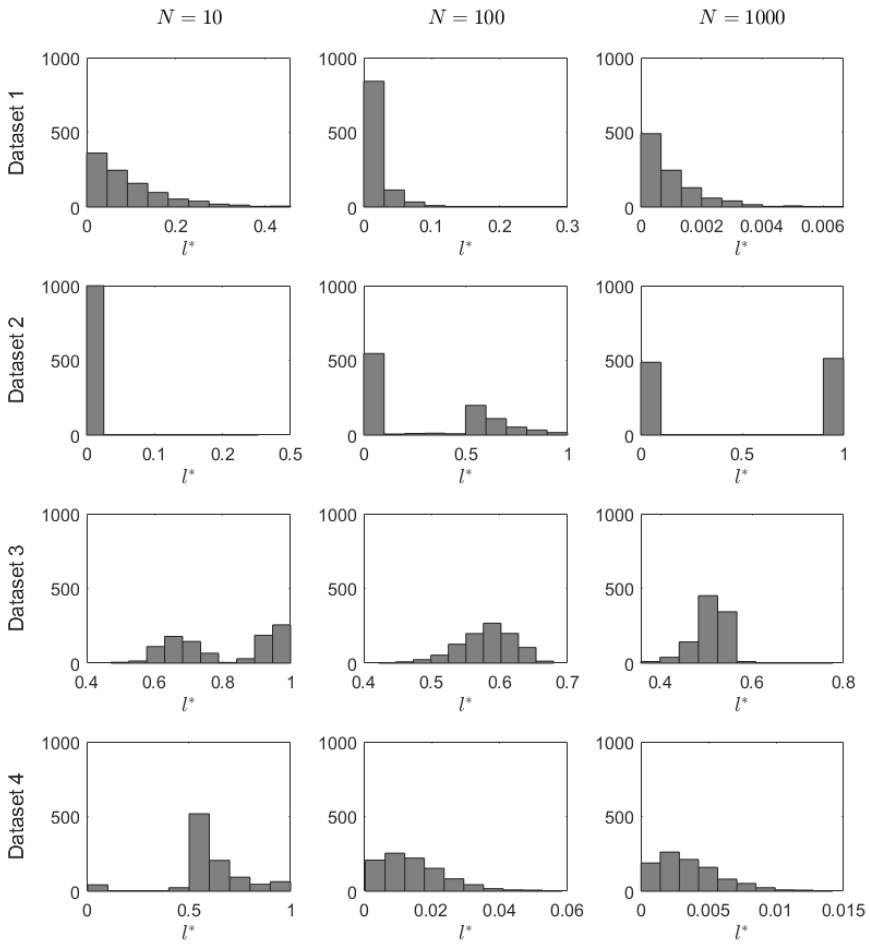


Figure 5: Histograms provided by l^* for all datasets.

In Figures 6 & 7, we present our results related to s^*. We observe that our results cover the entire interval (i.e., [0,10]) where the speed is realized. This means that even if the speed is part of the decision making process of the machine learning model (i.e., the SVM) for delivering the EoA, the limited uncertainty in the speed evaluation makes the FL model to pay no attention on s. However, the average s^* is around 5.0 for all the adopted datasets. Recall that s is a subject of the internal processes depending mainly on the available hardware and the adopted query execution plans.

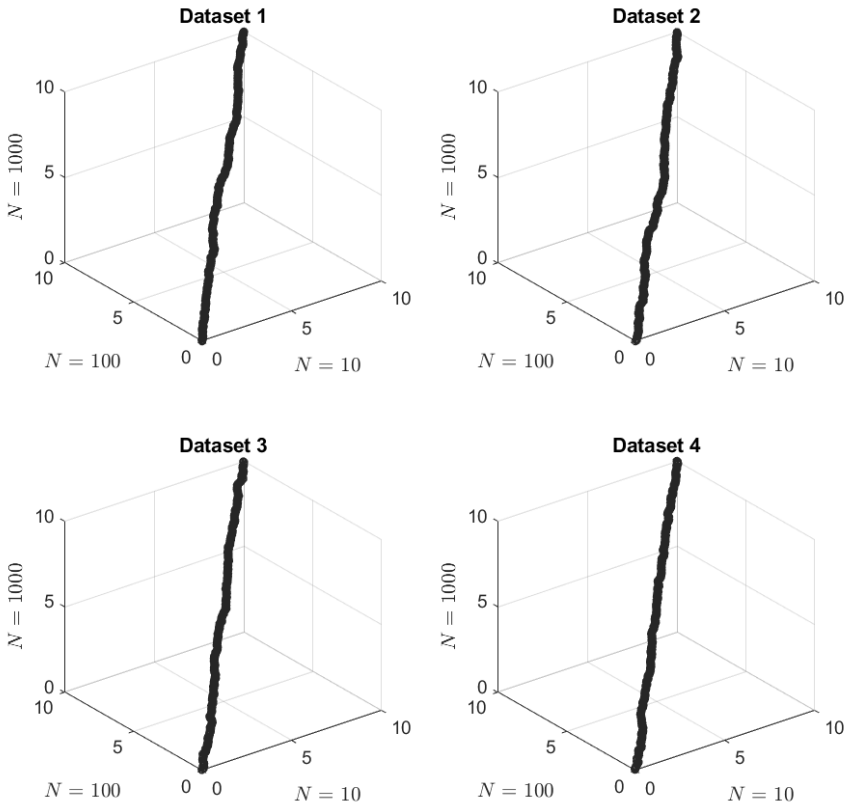


Figure 6: The relationship of the outcomes related to s^* for all datasets.

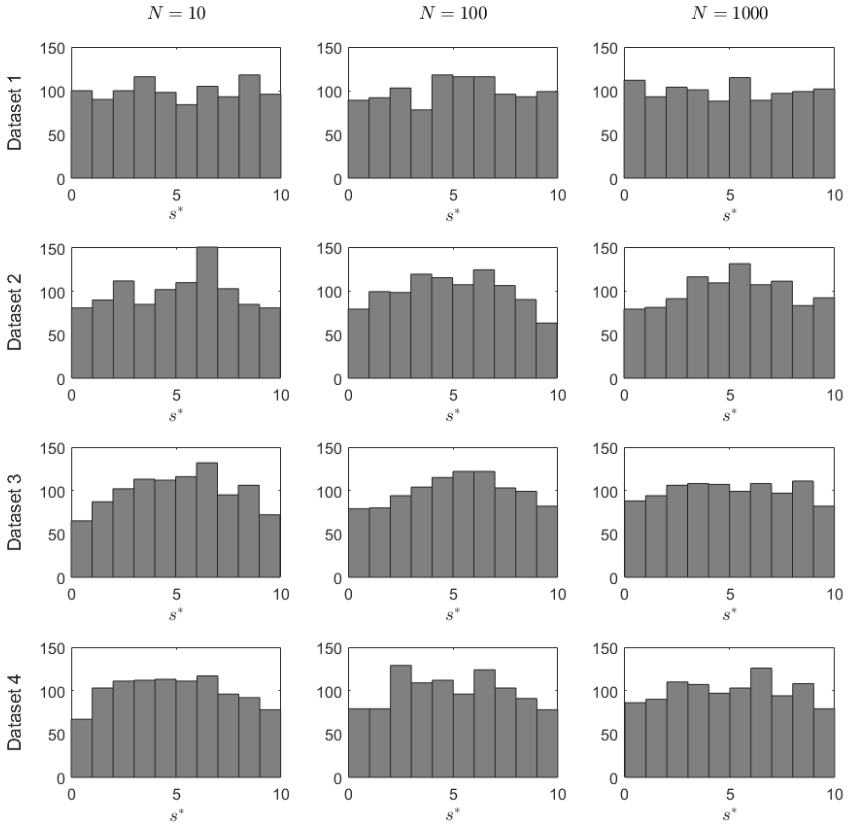


Figure 7: Histograms provided by s^* for all datasets.

The proposed model achieves the best possible performance when considering the information relevance between the queries constraints and the statistics of the available datasets. This is depicted by Figures 8 & 9. Our model is capable of resulting the r^* very close to the maximum possible value (i.e., unity). This means that query constraints are fully 'matched' against the available data and the appropriate EN is selected. Based on this observation, we can conclude that the execution of the query to the specific nodes will return the appropriate data without spending time and resources. Histograms show that r^* is concentrated close to unity for all the experimental scenarios and the adopted datasets. Hence, no matter the data distribution, the proposed statistical similarity process manages to exhibit the best pos-

sible performance being not affected by the statistics of each specific trace.

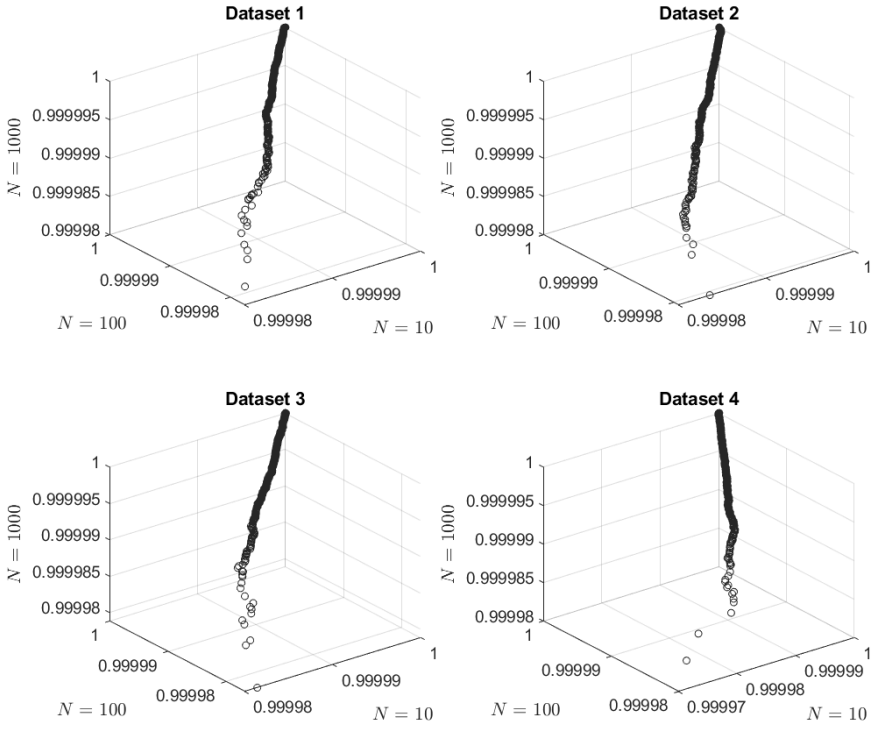


Figure 8: The relationship of the outcomes related to r^* for all datasets.

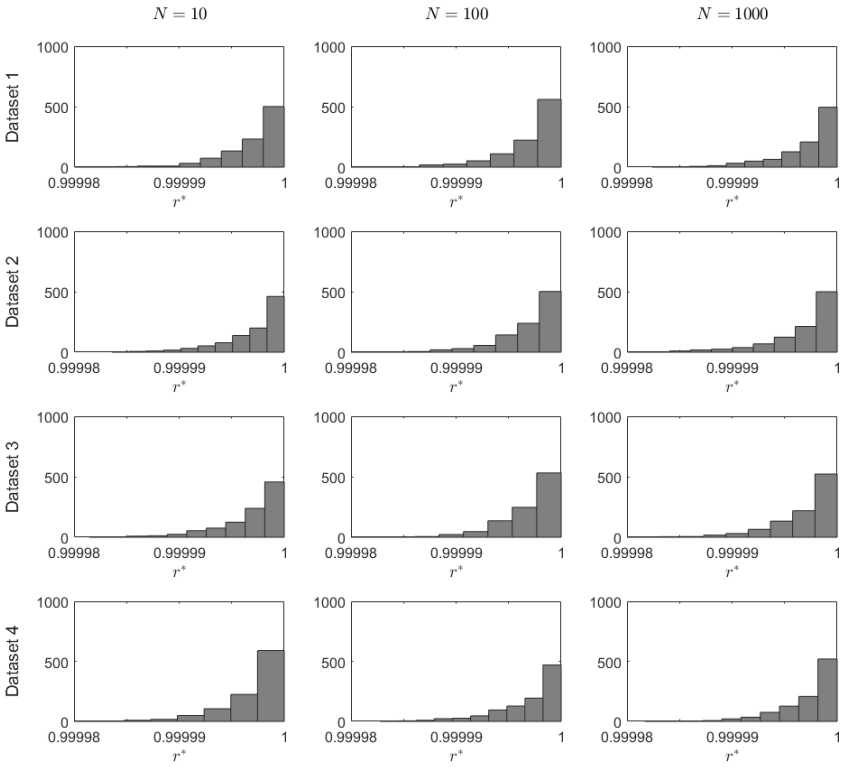


Figure 9: Histograms provided by r^* for all datasets.

When we focus on the throughput of QCs, we get the outcomes presented by Figure 10. Naturally evolved, R_T is high when $N \to 10$ and becomes low when the number of ENs increases. In the latter case, every QC should spend more time to conclude the final allocation, thus, the throughput is minimized. The lowest number of queries served per second is around 17 while the highest number is around 660. These numbers indicate that the proposed mechanism is not time consuming and the provision of the final response for each query is mainly affected by the response time of each EN. If our model is combined with a fast query execution scheme hosted in ENs could limit the observed latency in the provision of responses to end users or applications.

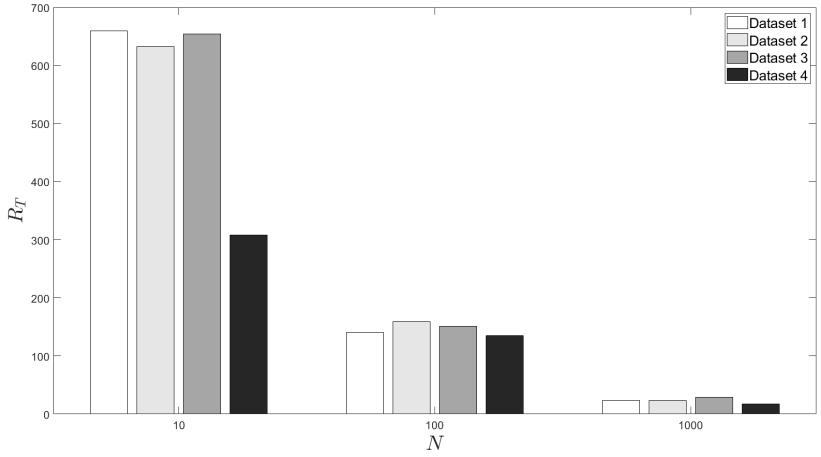


Figure 10: Our results concerning the throughput of the proposed model.

The distance of $n^{selected}$ from n^{opt} is depicted by Figure 11. In these results, we do not pay attention on s. We observe that the proposed model manages to deliver outcomes close to the ideal node. The interesting is that as $N \rightarrow 1,000$, our model exhibits the best possible performance no matter the adopted dataset. Our results are affected by l^* and r^* which are close to their optimal values as already discussed above, especially when we have to do with an increased number of ENs. When s is also taken into consideration, we get the outcomes provided by Figure 12. Now, our results are completely affected by s^* while observing an increment in the difference from n^{opt} as $N \rightarrow 1,000$. The 'uniformity' of the s^* as delivered in our experimental evaluation is that affecting most the optimality of the proposed solution.

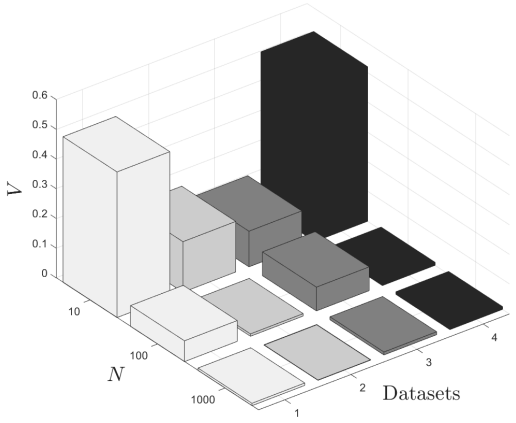


Figure 11: The distance from n^{opt} (results do not take into consideration the speed of ENs).

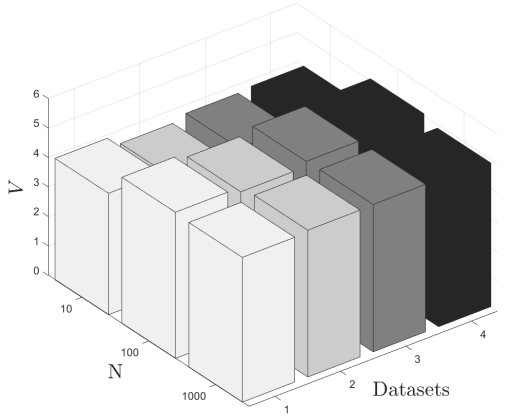


Figure 12: The distance from n^{opt} (results take into consideration the speed of ENs).

We perform an additional set of experiments to reveal the 'behaviour' of the T2FLS for different complexity classes. We consider that all the incoming queries belong to the same complexity class and report on the performance of our model concerning l^*. Figure 13 presents our results for Datasets 1 & 2. We observe that the complexity class of a query does not 'heavily' affect

the performance in contract to the number of nodes. This is natural as our model focuses on the load that a query will cause searching in the available nodes to find the best possible solution. In the majority of the outcomes, l^* is limited exhibiting the advantages of our T2FLS that is capable of finding a node with limited load to host the incoming queries.

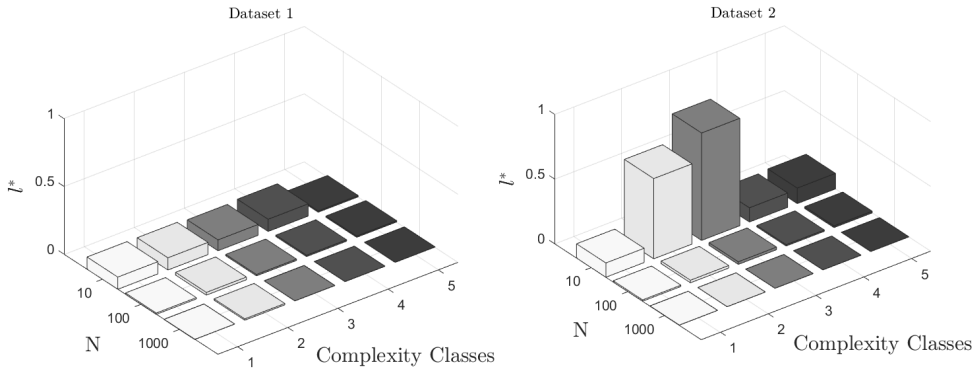


Figure 13: Performance outcomes for different complexity classes.

We proceed with a comparison of our model with our previous efforts and other models found in the relevant literature. The comparison refers in the throughput of QCs as well as the load of the selected node. Our model results an R_T varying from 17 to 660 queries per second for $N \in \{10, 100, 1000\}$ and an average $l^* \in [0.0009, 0.60]$ with the majority of values concentrated close to the lowest limit as already discussed. The Model M1 presented in [43] adopts a learning mechanism accompanied by a load balancing module. The discussed scheme manages to allocate 58.48 queries for $N = 100$ (our model serves 148.41 in average) to 454.55 queries for $N = 20$ (our model serves 300.70 queries per second, in average, for all datasets and for the same number of nodes). The average load of $n^{selected}$ (the learning scheme) is around 0.10 for $N \in \{2, 5, 20, 50, 100\}$. The average load for the clustering scheme is between 0.160 and 0.670. The model M2 discussed in [46] adopts a 'simple' learning scheme for the decision making. The performance of M3 related to the throughput is 50-100 queries per time unit when $N \to 500$ (our model serves 49.85 queries per second, in average, for all datasets and for $N = 500$ - the worst performance is retrieved when adopting the fourth dataset).

Finally, the model M3, discussed in [42], presents an allocation scheme where queries assignments are concluded by the assistance of an ensemble similarity model responsible to deliver the complexity class. Afterwards, the complexity class is be matched against the current load of every EN. M3 exhibits an R_T in the interval (approx.) [18, 125] for $N \in \{10, 100, 1000\}$ while l^* is in [0.28, 0.46] for all the adopted distributions.

We also compare our FLM with the scheme proposed in [74]. In Figures 14, 15 & 16, we present our comparative results for l^*, s^* and r^*, respectively. We observe that, concerning l^*, the FLM outperforms the TAM for all the experimental scenarios and the adopted datasets. The difference is high when we deal with an increased number of ENs. Related to s^*, the two models exhibit a similar performance as presented by Figure 15. An increased number of nodes will also increase the s^*. Finally, concerning r^*, we observe that the FLM also outperforms the TAM exceptional to two cases (i.e., dataset 1, $N = 10$ & dataset 4, $N = 10$).

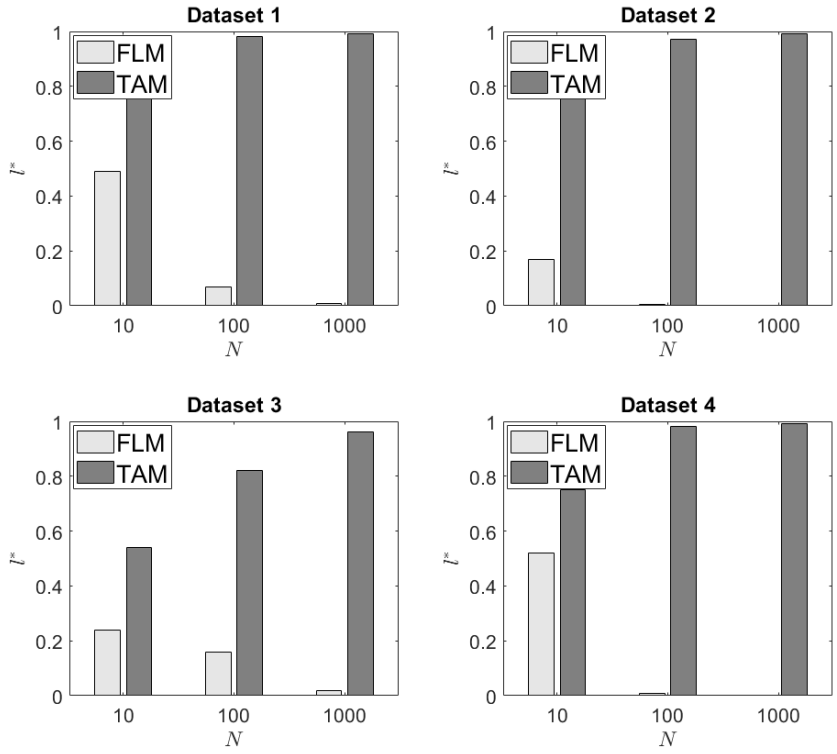


Figure 14: Comparative assessment between FLM and TAM for l^*.

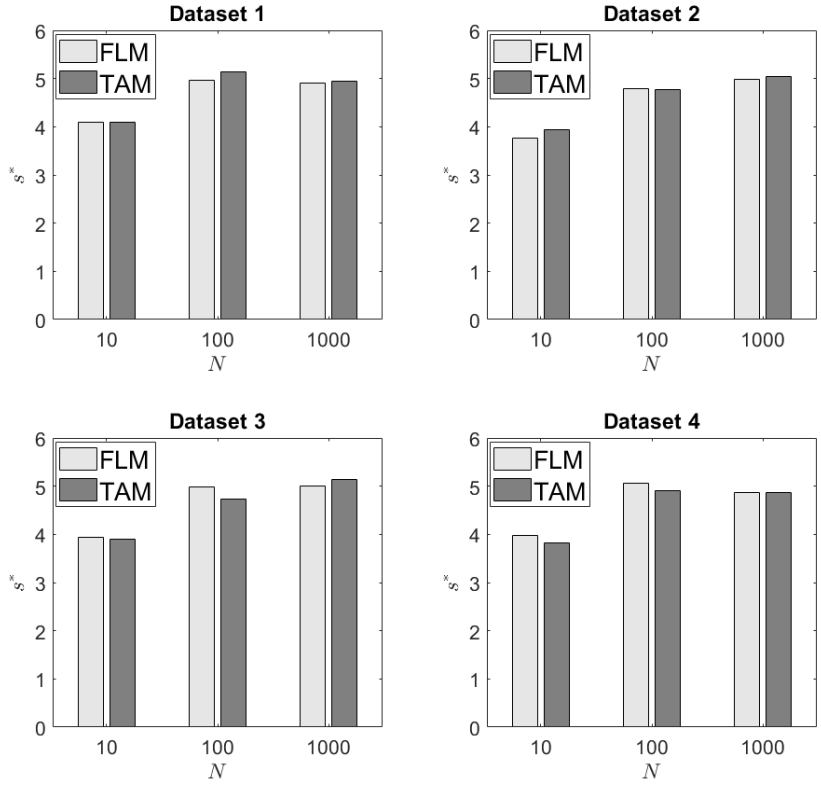


Figure 15: Comparative assessment between FLM and TAM for s^*.

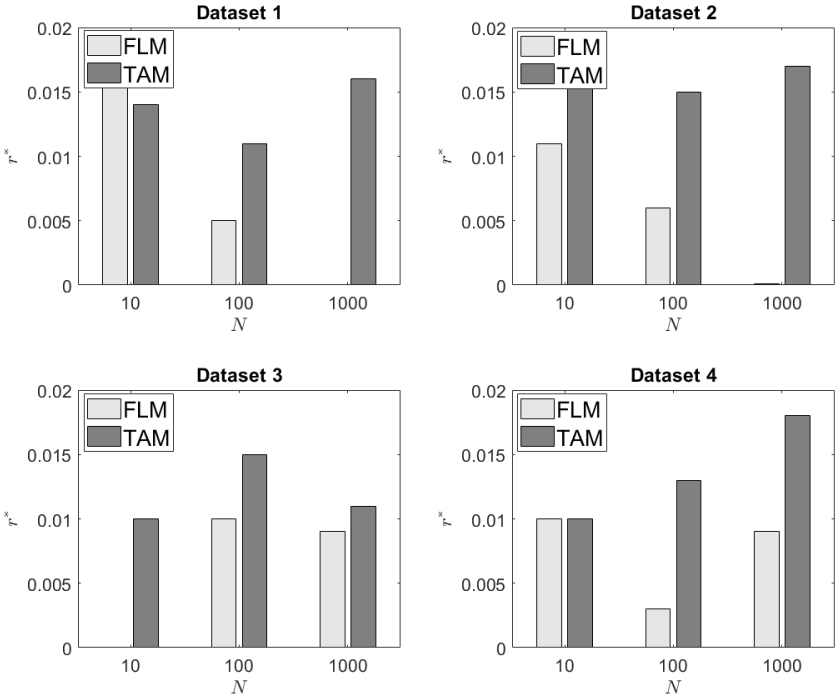


Figure 16: Comparative assessment between FLM and TAM for r^*.

Our last comparative scenario involves the case where no action is adopted when ties are identified in the FL outcomes as well as in evaluations delivered by TAM. Table 2 presents our outcomes. Again, we observe that the FLM outperforms the TAM and a high difference is realized in all the experimental scenarios.

Table 2: Comparison outcomes for FLM vs TAM for l^* (random selection in ties)

	Dataset 2		Dataset 3		Dataset 4	
N	FLM	TAM	FLM	TAM	FLM	TAM
10	0.078	0.780	0.140	0.270	0.090	0.740
100	0.200	0.970	0.220	0.400	0.110	0.980
1000	0.220	0.990	0.370	0.490	0.130	0.990

7. Conclusions & Future Work

The IoT infrastructure involves numerous autonomous devices that can interact with their environment, collect data and transfer them to the Cloud for further processing. Current advances propose the use of an additional processing layer between the IoT devices and the Cloud, i.e., the edge infrastructure. Edge nodes can host a (sub-)set of data and perform simple analytics queries/tasks as requested by end users or applications. This way, the latency in the provision of responses will be minimized as the processing activities are kept close to data sources. In this distributed infrastructure, multiple nodes are available to host the incoming queries/tasks on top of the local datasets. We propose a methodology for deciding the (sub-)set of edge nodes that will host and execute every query/task. We provide a decision making model that takes into consideration the contextual information related to queries and edge nodes (i.e., their characteristics). Any decision is aligned with the discussed information trying to efficiently match queries with edge nodes. We adopt the principles of Fuzzy Logic to support the management of the uncertainty related to the efficiency of each allocation. In addition, we also adopt a machine learning model to play the role of an expert that provides its view on the efficiency of each allocation. The opinion of the machine learning expert is, then, fed into the fuzzy model to deliver the final outcome. Being aligned with the available data, we propose an automated knowledge bases extraction scheme to support a data-aware decision making. Our experimental results indicate the pros and cons of the proposed approach and setups the basis for a comparative assessment. We show that our model is capable of efficiently 'matching' queries with edge nodes. In the first place of our future research plans is the incorporation of more parameters in the fuzzy model and the study of the hidden aspects of the fuzzy decision making. Hence, we will be able to offer a more complex, however, efficient model for delivering the envisioned allocations.

Acknowledgment

This research received funding from the European's Union Horizon 2020 research and innovation programme under the grant agreement No. 745829.

References

References

[1] Aazam, M., Hung, P. P., Huh, E. N., 'Smart Gateway based Communication for Cloud of Things', 9th ICSSNIP, 2014.

[2] Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D. J., Rasin, A., Silberschatz, A., 'HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads', PVLDB, 2(1), 2009.

[3] Agrawal, S., Milner, H., Kleiner, A., Talwalkar, A., Jordan, M., Madden, S., Mozafari, B., Stoica, I., 'Knowing When You're Wrong: Building Fast and Reliable Approximate Query Processing Systems', ACM SIGMOD, USA, 2014.

[4] Adami, D., Gabbrielli, A., Giordano, S., Pagano, M., Portaluri, G., 'A Fuzzy Logic Approach for Resources Allocation in Cloud Data Center', IEEE Globecom Workshops, 2015.

[5] Artail, H., El Amine, H., Sakkal, F., 'SQL query space and time complexity estimation for multidimensional queries', International Journal of Intelligent Information and Database Systems, vol. 2(4), 2008, pp. 460—480.

[6] Auria, L., Moro, R., 'Support Vector Machines (SVM) as a Technique for Solvency Analysis', Position Paper, DIW Berlin, 2008.

[7] Awadalla, M. H. A., 'Task Mapping and Scheduling in Wireless Sensor Networks', Int. Journal of Computer Science, 440(4), 2013.

[8] Azar, A. T., 'Overview of Type-2 Fuzzy Logic Systems', International Journal of Fuzzy Systems Applications, IGI Global, vol. 2(4), 2012, pp. 1-28.

[9] Balkensen, C., Tatbul, N., 'Scalable Data Partitioning Techniques for Parallel Sliding Window Processing over Data Streams', in Proc. of 8th Int. Workshop on Data Management for Sensor Networks, 2011.

[10] Bharti, S., Pattanaik, K., 'Task Requirement Aware Pre-processing and Scheduling for IoT Sensory Environments', Ad Hoc Networks, 50, 2016, pp. 102–114.

[11] Bullen, P., 'Quasi-Arithmetic Means', Handbook of Means and Their Inequalities', vol. 560, 2003.

[12] Cao, L., Rundensteiner, E. A., 'High Performance Stream Query Processing with Correlation-Aware Partitioning', VLDB Endowment, 7(4), 2013, pp. 265–276.

[13] Caruccio, L., Deufemia, V., Polese, G., 'Learning Effective Query Management Strategies from Big Data', in Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017.

[14] Castillo, O., Melin, P., 'Type-2 fuzzy logic theory and applications', Berlin, Germany: Springer-Verlag, 2008.

[15] Chen, Y., 'Support Vector Machines and Fuzzy Systems', In: Maimon O., Rokach L. (eds) Soft Computing for Knowledge Discovery and Data Mining. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-69935-6-9.

[16] Chen, X., Li, Y., Harrison, R., Zhang, Y. Q., 'Type-2 Fuzzy Logic-based Classifier Fusion for Support Vector Machines', Applied Soft Computing, vol. 8(3), 2008, pp. 1222-1231.

[17] Chen, W., Zhou, K., Yang, S., Wu, C., 'Data quality of electricity consumption data in a smart grid environment', Renew. Sustain. Energy Rev., vol. 75, 2017, pp. 98-–105.

[18] Çiğdem İnan, A., Akay, M. F., 'A hybrid congestion control algorithm for broadcast-based architectures with multiple input queues', Journal of Supercomputing, 2015, 71: 1907.

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

[20] Dittrich, J., Quiane-Ruiz, J. A., Jindal, A., Kargin, Y., Setty, V., Schad, J., 'Hadoop++: Making a Yellow Elephant Run Like a Cheetah', PVLDB, 3(1), 2010.

[21] Edalat, N., Xiao, W., Tham, C.-K., Keikha, E., Ong, L.-L., 'A price-based adaptive task allocation for Wireless Sensor Network', 6th IC-MASS, 2009.

[22] Gal, L., Lovassy, R., Rudas, I., Koczy, L., 'Learning the optimal parameter of the Hamacher t-norm applied for fuzzy-rule-based model extraction', Neural Computing and Applications, 24, 2014, pp. 133–142.

[23] Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., Riviere, E., 'Edge-centric Computing: Vision and Challenges', ACM SIGCOMM Computer Communication Review, 45(5), 2015.

[24] Gedik, B., 'Partitioning Functions for Stateful Data Parallelism in Stream Processing', VLDB Journal, vol. 23(4), 2014, pp. 517–539.

[25] Gedik, B., 'Partitioning Functions for Stateful Data Parallelism in Stream Processing', VLDB Journal, vol. 23(4), 2014, pp. 517–539.

[26] Greenberg, A., Hamilton, J., Maltz, D., Patel, 'The Cost of a Cloud: Research Problems in Data Center Networks', ACM SIGCOMM, 39(1), 2008, pp. 68–73.

[27] Gualtieri, M., Yuhanna, N., 'The Forrester Wave: Big Data Hadoop Solutions', Technical Report, 2014.

[28] Han, J., Kamber, M., Pei, J., 'Data Mining, Concepts and Techniques', 3rd Edition, Morgan Kaufmann Publishers, Elsevier, 2012.

[29] Haßler, M. Jeschke, S. Meisen, T., 'Similarity Analysis of Time Interval Data Sets Regarding Time Shifts and Rescaling', In Proceedings International work-conference on Time Series, 2017.

[30] Hearst, M. A., 'Support Vector Machines', IEEE Intelligent Systems, vol. 13(4), 1998, pp. 18—28.

[31] Hossain, K., Raihan, Z., Hashem, M., 'On Appropriate Selection of Fuzzy Aggregation Operators in Medical Decision Support System', In Proc. of the 8th Int. Conf. on Comp. and Inf. Technology, 2005.

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

[33] Jiang, D., Ooi, D. C., Shi, L., Wu, S., 'The Performance of MapReduce: An In-depth Study', PVLDB, 3(1), 2010.

[34] Jiang, D. L., Wang, K. W., Wang, X. D., 'Clustering method of fuzzy equivalence matrix to bad-data detection and identification', Power Syst. Protection Control, vol. 39(21), 2011, pp. 1—6.

[35] Kalapanidas, E., Avouris, N., Cracium, M., Neagu, D., 'Machine Learning Algorithms: a Study on Noise Sensitivity', in Proc. of the 1st Balkan COnference on Informatics, 2003.

[36] Karanika, A., Oikonomou, P., Kolomvatsos, K., Loukopoulos, T., 'A Demand-driven, Proactive Tasks Management Model at the Edge', in IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE World Congress on Computational Intelligence (WCCI), Glasgow, UK, 2020.

[37] Karanika, A., Soula, M., Anagnostopoulos, C., Kolomvatsos, K., Stamoulis, G., 'Optimized Analytics Query Allocation at the Edge of the Network', in 12th International Conference on Internet and Distributed Computing Systems, Naples, Italy, Oct. 10-12, 2019.

[38] Kolomvatsos, K., 'An Intelligent Scheme for Assigning Queries', Springer Applied Intelligence, https://doi.org/10.1007/s10489-017-1099-5, 2018.

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

[40] Kolomvatsos, K., Anagnostopoulos, C., 'Multi-criteria Optimal Task Allocation at the Edge', Elsevier Future Generation Computer Systems, vol. 93, 2019, pp. 358-372.

[41] Kolomvatsos, K., Anagnostopoulos, C., 'In-Network Edge Intelligence for Optimal Task Allocation', 30th International Conference on Tools with Artificial Intelligence, Nov. 5-7, Volos, Greece, 2018.

[42] Kolomvatsos, K., Anagnostopoulos, C., 'An Edge-Centric Ensemble Scheme for Queries Assignment', in 8th International Workshop on Combinations of Intelligent Methods and Applications in conjunction with the 30th International Conference on Tools with Artificial Intelligence, 2018.

[43] Kolomvatsos, K., Anagnostopoulos, C., 'Reinforcement Machine Learning for Predictive Analytics in Smart Cities', Informatics, MDPI, 4, 16, 2017.

[44] Kolomvatsos, K., Anagnostopoulos, C., Hadjiefthymiades, S., 'Data Fusion & Type-2 Fuzzy Inference in Contextual Data Stream Monitoring', IEEE Transactions on Systems, Man and Cybernetics: Systems, vol. 47(8), 2016, pp. 1–15.

[45] Kolomvatsos, K., Anagnostopoulos, C., Hadjiefthymiades, S., 'A Time Optimized Scheme for Top-k List Maintenance over Incomplete Data Streams', Elsevier Information Sciences (INS), vol. 311, pp. 59-73, 2015.

[46] Kolomvatsos, K., Hadjiefthymiades, S., 'Learning the Engagement of Query Processors for Intelligent Analytics', Applied Intelligence, vol. 46, 2017, pp. 96–112.

[47] Kumar, K. N., Murthy, N. V., Reddy, C. S., 'An Effective Parallel XML Fuzzy Query Processing', International Journal of Computer Applications, vol. 86(9), 2014.

[48] Liu, X., Cheng, G., Wu, J., 'Analyzing Outliers Cautiously', IEEE Transactions on Knowledge and Data Engineering, vol. 14(2), 2002, pp. 432–437.

[49] Mendel, J. M., 'Type-2 Fuzzy Sets and Systems: An Overview', IEEE Computational Intelligence Magazine, 2(2), 2007.

[50] Mendel, J. M. Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Upper Saddle River, Prentice-Hall, 2001.

[51] Orfila, A., Carbo, J., Ribagorda, A., 'Fuzzy Logic on Decision Model for IDS', in proceedings of the 12th IEEE International Conference on Fuzzy Systems, 2003.

[52] Pasteris, S., Wang, S., Makya, C., Chan, K., Herbster, M., 'Data Distribution and Scheduling for Distributed Analytics Tasks', IEEE SWC Conference, 2017.

[53] Raman, V., Raman, B., Hellerstein, J. M., 'Online dynamic reordering for interactive data processing', In VLDB, 1999.

[54] Raipurkar, A. R., Bamnote, G. R., 'Fuzzy Logic Based Query Optimization in Distributed Database', International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, 2013.

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

[56] Roman, R., Lopez, J., Mambo, M., 'Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges', Future Generation Systems, 78(2), 2018, pp. 680–698.

[57] Saleh, A. I., Rabie, A. H., Abo-Al-Ez, K. M., 'A data mining based load forecasting strategy for smart electrical grids', Adv. Eng. Inform., vol. 30(3), 2016, pp. 422-–448.

[58] Satyanarayanan, M., 'A brief history of cloud offload: A personal journey from Odyssey through cyber foraging to cloudlets', Mobile Computing Communications, 18(4), 2015, pp. 19–23.

[59] Schwaab, A. A., Nassar, S. M., Filho, J., 'Automatic Methods for Generation of Type-1 and Interval Type-2 Fuzzy Membership Functions', Journal of Computer Sciences, vol. 11(9), 2015, pp. 976–987.

[60] Simon, M., Pataki, N., 'SQL Code Complexity Analysis', In proceedings of the 8th International Conference of Applied Informatics, 2010.

[61] Singh, S., Singh, N., 'Big Data Analytics', In Proc. of the International Conference on Communication, Information and Computing Technology, 2012.

[62] Singh, S. S., Sayal, R., Rao, V., 'Analysis and Usage of Fuzzy Logic for Optimized Evaluation of Database Queries', International Journal of Computer Applications, vol. 16(3), 2011.

[63] Sola, H. B., Fernandez, J., Hagras, H., Herrera, F., Pagola, M., Barrenechea, E., 'Interval Type-2 Fuzzy Sets are Generalization of Interval-Valued Fuzzy Sets: Toward a Wider View on Their Relationship', IEEE Transactions on Fuzzy Systems, vol. 23(5), 2015.

[64] Tian, Y., Ekici, E., Ozguner, F., 'Energy-Constrained Task Mapping and Scheduling in Wireless Sensor Networks', IEEE ICMASS, 2005.

[65] Tsanas, A., Xifara, A., 'Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools', Energy and Buildings, vol. 49, 2012, pp. 560–567.

[66] Vashistha, A., Jain, S., 'Measuring Query Complexity in SQLShare Workload', in proceedings of the International Conference on Management of Data, 2016.

[67] Vigler, C., Goldenstein, S., Stolfi, J., Pavlovic, V., Metaxas, D., 'Outlier rejection in high-dimensional deformable models', Image and Vision Computing, vol. 25, 2007, pp. 274-–284.

[68] Walker, C. L., Walker, E. A., 'The algebra of fuzzy truth values', Fuzzy Sets Systems, 149, pp.309-347, 2005.

[69] Yang, T., Wei, J., Fan, B., Wang, X., Zhang, H., 'Structural Similarity Computation Based On Extended Edge Matching Method', International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

[70] Yang, J., Zhang, H., Ling, Y., Pan, C., Sun, W., 'Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization', IEEE Sensors Journal, 14(3), 2014, pp. 882–892.

[71] 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.

[72] Zeitler, E., Risch, T., 'Scalable Splitting of Massive Data Streams', in DASFAA 2010, vol 5982, Springer, 2010.

[73] Zimmermann, H.-J., 'Fuzzy Set Theory—and its Applications', Kluwer Academic Publishers, Boston, 2nd edition, 1991.

[74] Zoghdy, S., Nofal, M., Shohla, M., El-sawy, 'An Efficient Algorithm for Resource Allocation in Parallel and Distributed Computing Systems', in International Journal of Advanced Computer Science and Applications, vol. 4(2), 2013.