Kolomvatsos, K. and Anagnostopoulos, C. (2018) In-Network Decision Making Intelligence for Task Allocation in Edge Computing. In: 30th International Conference on Tools with Artificial Intelligence (ICTAI 2018), Volos, Greece, 5-7 Nov 2018, pp. 655-662. ISBN 9781538674499 (doi:10.1109/ICTAI.2018.00104).

This is the author's final accepted version.

There may be differences between this version and the published version. You are advised to consult the publisher's version if you wish to cite from it.

http://eprints.gla.ac.uk/166372/

Deposited on: 07 August 2018

Enlighten – Research publications by members of the University of Glasgow
http://eprints.gla.ac.uk

In-network Decision Making Intelligence for Task Allocation in Edge Computing

Kostas Kolomvatsos
School of Computing Science
University of Glasgow, UK
kostas.kolomvatsos@glasgow.ac.uk

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

***Abstract*—Humongous contextual data are produced by sensing and computing devices (nodes) in distributed computing environments supporting inferential/predictive analytics. Nodes locally process and execute analytics tasks over contextual data. Demanding inferential analytics are crucial for supporting local real-time applications, however, they deplete nodes' resources. We contribute with a distributed methodology that pushes the task allocation decision at the network edge by intelligently scheduling and distributing analytics tasks among nodes. Each node autonomously decides whether the tasks are conditionally executed locally, or in networked neighboring nodes, or delegated to the Cloud based on the current nodes' context and statistical data relevance. We comprehensively evaluate our methodology demonstrating its applicability in edge computing environments.**

***Index Terms*—Edge-centric task allocation, multi-criteria decision making, contextual reasoning, statistical data relevance.**

I. Introduction

Internet of Things (IoT) gives the opportunity for the development of intelligent analytics applications over sensing and computing devices. Such devices are interconnected to communicate while collecting contextual data from their environment. Moreover, they become knowledge producers as based on their (limited) computational capabilities they proceed with analytics tasks, knowledge extraction and inference. The inferred and/or exacted knowledge refers to the outcome of execution tasks issued by analytics (predictive, exploratory) queries defined by analysts and applications [1], [2], [13].

Legacy systems adopt the Cloud infrastructure, where various services are available. However, large-scale data centers present in Cloud are centralized systems which implies a large average separation between devices and their Clouds. This, in turn, increases the average latency and data migration [16], where delay sensitive applications can be negatively affected. For alleviating such problems, edge computing [15] is coming into play. It deploys Cloud-like capabilities in the network edge devices and gateways making them capable of processing the collected data, thus, becoming efficient knowledge producers [8]. Edge-centric analytics contributes to a distributed analytics tasks exploiting the interconnected heterogeneous resources controlled by edge nodes.

Challenge: Our aim is to keep multiple edge analytics and processing tasks as close to the nodes and their corresponding sources of contextual data as possible avoiding migrating data and tasks to the Cloud. For supporting edge-centric analytics tasks, edge nodes should compute/execute a set of tasks [5]. Such tasks are generated, possibly at high rates, and should be concluded immediately in terms of allocation and execution taking into consideration certain contextual parameters focusing explicitly on: (i) current node's computational load, (ii) communication cost within a group of edge nodes, and (iii) relevance with the data to be processed. Such tasks allocation process should be promptly determined in a distributed way within a group of nodes. The challenge is to maximize nodes' performance while minimizing the required resources and the induced communication overhead. Related research deals with centralized approaches, thus, the task allocation models suffer from the drawbacks reported for Cloud computing. Our challenge is to *push the task allocation intelligence into the edge network exploiting the nodes' context and data relevance*, thus, the edge nodes locally, based on the current contextual data reason about appropriate decisions for tasks allocation.

Contribution: We build on the *autonomous* nature of nodes by proposing a distributed methodology for *pushing* the analytics tasks allocation of a stream of incoming tasks to the network edge. Due to nodes' resource constraints, each node can execute a limited number of tasks, thus, selecting those that *maximize* its performance and meet specific resource constraints. Our two-level decision making methodology sequentially reasons on the best decision for task execution either on the node, or on its neighboring networking peer nodes, or at the Cloud. The rationale behind our methodology is that we distinguish two conditioned decisions on an edge node. The first decision is related to whether a task can be executed locally based on the current context of the node. The second decision is related to whether this task can be executed in a neighboring node with statistically similar data and more available resources, or on the Cloud conditioned on the result of the former decisions. Such sequential decision making is achieved by the fusion of a probabilistic classification with the multi-criteria optimization VIKOR methodology [17] w.r.t. the current load, the remaining resources, the collected data statistics, and task's characteristics (e.g., priority, execution requirements). Our contribution is: (i) a distributed sequential decision making mechanism for tasks allocation, (ii) a scheme for efficient selection of neighboring nodes when tasks are decided not to be executed locally; (iii) comprehensive experimental simulations and sensitivity analysis on the most significant methodology parameters.

The paper is organized as follows: Section II reports on related work and our contribution. Section III presents the problem of pushing tasks allocation to the edge and elaborates on the decision making methodology, while in Section IV, we provide the experimental evaluation of our scheme. Section V concludes the paper discussing future research plans.

II. Related Work

Prior Research Activities. Task allocation is studied in Wireless Sensor Networks (WSNs) as in [20], where the authors present a task mapping into WSN sensing devices taking into consideration energy constraints, communication and computation scheduling. The approach in [3] is based on a collaborative processing among nodes for task allocation adopting linear task clustering and a node assignment mechanism based on task duplication schemes. The model in [7] focuses on minimizing the task execution time in a clustered WSN. In [6], the authors propose a model for allocating the incoming tasks in WSN sensors according energy requirements. An Integer Linear Programming (ILP) formulation for task allocation is proposed in [24], where the time and energy costs of both computation and communication activities are considered. In [22], the authors propose a modified version of the binary Particle Swarm Optimization (PSO). The method adopts a different transfer function, a new position updating procedure and mutation to obtain the best solution. Another PSO-based solution is presented in [14] which allocates tasks into a number of robots decreasing the communication cost in a WSN. In [10], the authors present a mechanism of dynamic alliance based on a Genetic Algorithm (GA) to acquire the balance between energy consumption and accuracy considered in area sum method. In [11], the authors propose a QoS aware resource scheduling algorithm adopting PSO to derive the final scheduling. The aim is to reduce time and ensure the load balancing to maximize the performance.

A review on task allocation algorithms that fit in the Cloud is discussed in [18]. In [13], the authors propose a model for data distribution over computing devices. The model aims to eliminate the bandwidth and the storage constraints. The assigned tasks are executed over data while only one task is executed in each machine. However, this could not be the usual case when we consider real-time applications. Simulated annealing is adopted to solve the problem of task allocation in Cloud [12] by parallelizing various tasks in a multi-Cloud system. JarvSis is proposed as a distributed scheduler capable to automate the execution of multiple heterogeneous tasks in IoT [4]. Through JarvSis, developers can easily configure and deploy hierarchies of control tasks running in the Cloud. A distributed optimization approach manages to adopt multiple optimization results [19]. This approach can provide sub-optimal solutions when a centralized approach can not be used in a real environment.

Research Outcome: The major difference of our work compared to the above-mentioned approaches is that the related work to task allocation problems mainly focuses on energy constraints in the decision making process. Additionally, they focus on eliminating the communication overhead to reduce messages transmission, thus, to lower the number of collisions. They are, usually, *centralized* approaches meaning that a central unit decides on the final allocation based on the currently available information on the performance and the load of nodes. This inevitably conveys the disadvantages of any centralized system including communication overhead and a single point of failure. A baseline solution is to transfer the data to the specific node where the task will be executed. In general, research efforts adopting this methodology focus on 'univariate' contextual information meaning that the final decision for task allocation is delivered based only on a single parameter/perspective, e.g., either energy, or transmission requirements, or topology of the network. In addition, data migration techniques suffer from the increased migration cost especially in the network edge.

To the best of our knowledge our methodology firstly departs from the centralized task allocation intelligence and focuses on a distributed, local 'multivariate' case where the intelligence is pushed to the edge network. That said, the final decision is locally taken based on multiple parameters/perspectives, e.g., the current load of the nodes, their processing power, communication cost, remaining resources and data relevance. Our methodology casts as a local multi-criteria decision making mechanism for delivering the best task allocation decision. Apart from that, we further take into consideration the data collected at each node without adopting any data migration solutions, thus, avoiding redundant data communication overhead. Only meta-data information about a task and sufficient statistics over the collected data in nodes, which are relatively negligible comparing with raw data transfer, are disseminated into the network edge. Our sequential decision making methodology aims to eliminate the start-off time of tasks execution giving priority to local decisions, i.e., to the node where each task is initially assigned.

III. In-Edge Network Task Allocation

A. Definitions & Overview

Consider a set of N nodes, i.e., $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$. Nodes form a network represented by the graph $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{E}$ is the set of edges connecting the nodes. An edge $e_{ij} \in \mathcal{E}$ represents the communication channel between nodes n_i and n_j and is characterized by the communication cost $\kappa_{ij} > 0$. Given a node n_i, its neighborhood is defined by the subset of nodes $\mathcal{N}_i = \{n_j \in \mathcal{N} : e_{ij} \in \mathcal{E}\}$. Nodes are involved in certain tasks: sensing, collecting, computing and processing contextual data. Tasks are assigned to n_i at random intervals through a *task stream* $\mathcal{T}_i = (\langle T_{it}, \mathcal{C}_{it} \rangle)$ defined by a series of temporal ordered tuples $\langle T_{it}, \mathcal{C}_{it} \rangle$, where $t = 1, 2, \dots$ is the discrete time instance occurrence of the task T_{it} and $\mathcal{C}_{it}$ is a set of constraints for T_{it}. T_{it} is accompanied by the realization of a set of constraints $\mathcal{C}_{it}$, e.g., consider $\mathcal{C} = \{\text{priority, latency, lifetime}\}$ be the set of constraints and

$\mathcal{C}_{it} = \{0, 2\text{s}, 15\text{s}\}$ be their realization for T_{it}; $\mathcal{C}_{it} = \emptyset$ denotes no constraints for T_{it}.

A task stream $\mathcal{T}_i$ at node n_i yields an appropriate decision making for task allocation. Upon a task T_{it} reception at time instance t, node n_i creates the *context vector* $\mathbf{v}_i = [\lambda_i, \pi_i, \rho_i]^\top$, where $\lambda_i \in (0, 1)$ is the current percentage node load, $\pi_i \in (0, 1)$ is the priority of the task, and $\rho_i \in (0, 1)$ is the ratio of the remaining available resources of n_i; without loss of generality, ρ_i represents the available ratio of resources left for task execution in n_i, e.g., remaining energy budget. n_i, after receiving T_{it} at t, must decide w.r.t. $\mathcal{C}_{it}$ and vector $\mathbf{v}$ whether to: **(i)** Execute T_{it} locally - action a_l; **(ii)** Send T_{it} for execution to one of the networked neighboring nodes in the local neighbourhood $\mathcal{N}_i$ of n_i - action a_n; **(iii)** Send T_{it} to be executed in the Cloud - action a_c.

Our decision making methodology is a function $g : \mathbf{v} \rightarrow \{a_l, a_n, a_c\}$, where action a_l refers to task execution at n_i, a_n refers to task execution of a selected neighbouring node $n_j \in \mathcal{N}_i$ and action a_c refers to the task execution at Cloud since there is no node $n_j \in \mathcal{N}_j$ appropriate for the execution of T_{it}. The decision making function $g(\mathbf{v})$ secures the execution of task T_{it} as it sequentially selects the best decision among actions a_l, a_n, a_c that are mutually exclusive and exhaustively describe the universe of discourse (concerning the available actions for executing a task).

n_i senses and stores its d-dimensional contextual data vectors $\mathbf{x} \in \mathbb{R}^d$ in a dataset $\mathcal{X}_i = \{\mathbf{x}\}$ characterized by the specific statistics: expectation vector $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top]$. Each element σ_{kl} of matrix Σ is the covariance between data dimensions x_k and x_l from the data vector $\mathbf{x}$. At predefined intervals, n_i exchanges contextual information about its task requirements vector $\mathbf{v}_i$ and the current data statistics $(\boldsymbol{\mu}, \Sigma)_i$ to support the decision making and receives $(\boldsymbol{\mu}, \Sigma)_j$ from its neighbors $n_j \in \mathcal{N}_i$. Node n_i produces then the *information vector*: $\mathbf{p}_{ij} = [\lambda_j, \kappa_{ij}, \delta_{ij}]^\top$ for each of its neighbors n_j, where δ_{ij} is defined as the *data statistical difference* between statistics $(\boldsymbol{\mu}, \Sigma)_i$ and $(\boldsymbol{\mu}, \Sigma)_j$, defined by the convex sum of Euclidean norm and Frobenius norm over the expectation vectors and covariance matrices, respectively:

$$\delta_{ij} = \frac{1}{2}(\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_2 + \|\Sigma_i - \Sigma_j\|_F), \tag{1}$$

where the Frobenius norm is defined as: $\|\Sigma\|_F = \sqrt{\sum_{k=1}^d \sum_{l=1}^d |\sigma_{kl}|^2}$ and the Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\sum_{k=1}^d x_k^2}$. The advantage of applying a convex combination of both norms is that we can easily combine heterogeneous models exhibiting different characteristics. The δ_{ij} represents the dataset similarity between $\mathcal{X}_i$ and $\mathcal{X}_j$ in terms of their sufficient statistics. The rationale for the data statistical difference is that n_i takes into consideration the dataset similarities among their neighboring nodes to potentially assign its analytics tasks over similar data, thus, extracting approximately similar pieces of knowledge. This is reasoned when n_i cannot proceed with local tasks execution (w.r.t. context vector), thus, assigning the tasks to neighboring nodes n_j with *similar* data statistics achieves its goals and avoid transferring the task to the Cloud. Based on the received information vectors $\{\mathbf{p}_{ij}, n_j \in \mathcal{N}_i\}$ and the current context vector $\mathbf{v}_i$, n_i then decides on the actions $\{a_l, a_n, a_c\}$.

Methodology Overview: Upon the reception of $T_{it} \in \mathcal{T}_i$, n_i sequentially examines the available choices for the execution of T_{it}. Initially, n_i checks if T_{it} can be executed locally, thus, the communication cost is $\kappa_{ii} = 0$ and the start-off time is limited. Based on $\mathbf{v}_i$, n_i is then able to decide on action a_l. If n_i rejects action a_l, it checks if T_{it} can be executed by a node n_j from its neighborhood $\mathcal{N}_i$, i.e., action a_n. This decision is based on the processing of the information vectors and context vector, $\mathbf{p}_{ij}$ and $\mathbf{v}_i$, where the expected communication cost is κ_{ij}. If no node n_j can be assigned the T_{it}, n_i decides to send it to Cloud with inherent increased communication cost including e.g., data transfer; possibly higher than $\max(\kappa_{ij}), \forall i, j$. The decision function g decides the one-step optimal execution of T_{it} w.r.t. actions a_l, a_n, a_c. $g(\cdot)$ delivers the final action a_i based on a sequential processing applying the *principle of optimality*, which implies that every decision should be optimal for the remaining problem (initially, we select between three actions, next, we select between two actions).

B. Edge-centric Methodology

We analytically describe our methodology for concluding on an action to execute T_{it}. It is worth noting that the proposed techniques should deliver the decision in the minimum time since our methodology supports (near) real-time applications at the network edge. For deriving the action a_l, we rely on a probabilistic classifier [9], while for actions a_n and a_c, we adopt the multi-criteria VIKOR method [17] over the context and information vectors. The probabilistic classifier is computationally light and requires less training data compared to other classification methods while being highly scalable (linearly) with the number of data [23]. The VIKOR method solves decision problems with conflicting and noncommensurable criteria, assuming that compromise is acceptable for conflict resolution. In any case, it tries to approach a solution close to the optimal while the evaluation of a solution is based on a set of criteria. VIKOR ranks alternatives and determines the solution that is the closest to the optimal.

1) Probabilistic Local Task Allocation: The decision making for the action a_l depends on a Bayesian inference for deriving the posterior probability over the prior probabilities of specific events. Te probabilistic decision maker classifies the context vector $\mathbf{v}_i$ to the action class a_l and the complement $\bar{a}_1$, where the later represents the actions $\bar{a}_1 = \{a_n, a_c\}$. The output is the probability of T_{it} be executed at n_i or not. Node n_i requires a training set $\mathcal{I} = \{\mathbf{v}_k, a_k\}$ to train the classifier for estimating the probability $P(a_l|\mathbf{v}_i)$. Hence, the decision maker decides on a_l, i.e.,

$$g(\mathbf{v}_i) = \begin{cases} \{a_l\} & \text{if } P(a_l|\mathbf{v}_i) > P(\bar{a}_l|\mathbf{v}_i) \\ \{\bar{a}_l\} = \{a_n, a_c\} & \text{otherwise.} \end{cases} \tag{2}$$

If the decision is the a_l class, T_{it} is locally executed, otherwise, node n_i proceeds with the multi-criteria decision making to conclude either on a_n or a_c actions.

2) Multi-criteria Local Task Allocation: In our context, the VIKOR method results to an ordered list of the information vectors $(\mathbf{p}_{i1}, \mathbf{p}_{i2}, \dots \mathbf{p}_{iN_i})$ with ranking scores $z_1, z_2, \dots, z_{N_i}$ with $N_i = |\mathcal{N}_i|$. The required parameters and the steps if the VIKOR model are as follows. For the j-th candidate node $n_j \in \mathcal{N}_i$ with rank z_j, the rating of the k-th criterion dimension from the information vector $\mathbf{p}_{ij} = [\lambda_j, \kappa_{ij}, \delta_{ij}]$ is denoted by h_{jk}. The steps for deciding on an optimal action, i.e., the best candidate node for task execution are:

- **Step 1:** Determine the highest h^+_{jk} and lowest h^-_{jk} values for the k-th criterion dimension for all $n_j \in \mathcal{N}_i$;
- **Step 2:** Calculate the ratios per k-th criterion dimension:

$$S_k = \sum_{n_j \in \mathcal{N}_i} w_k \frac{h^+_{jk} - h_{jk}}{h^+_{jk} - h^-_{jk}},$$

$$R_k = \max_{n_j \in \mathcal{N}_i} \{ \frac{h^+_{jk} - h_{jk}}{h^+_{j} - h^-_{jk}} \},$$

  where $w_k \in (0, 1)$ is the weight for the k-th criterion dimension expressing its relative importance against the remaining criteria/dimensions. Based on the ratios, define the normalized convex criterion score:

$$q_k = \frac{1}{2} (\frac{S_k - \min\{S_k\}}{\max\{S_k\} - S_k} + \frac{R_k - \min\{R_k\}}{\max\{R_k\} - R_k}) \quad (3)$$

  This creates the criterion vector $\mathbf{q} = [q_1, q_2, q_3]^\top$ for load, communication cost and dataset statistics difference, respectively.
- **Step 3:** The score value for candidate n_j is then defined as $z_j = \mathbf{q}^\top \mathbf{p}_{ij}$, i.e., the vector information distance of node n_j from the criterion vector. The nodes are then ranked w.r.t. the inner products z_j in a descending order and the node $n_{j^*} = \arg\max_{n_j \in \mathcal{N}_i} \{z_j\}$ is candidate for selection.

Now, the candidate node n_{j^*} is to be assigned the task T_{it} if point-wise the elements of $\mathbf{p}_{ij^*}$ are greater than the pre-defined thresholds from the threshold vector $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^\top$ corresponding to the thresholds for the dimension criteria of interest (notated by: $\mathbf{p}_{ij^*} \succ \boldsymbol{\theta}$). In this case, action a_n is decided with node n_{j^*}. Otherwise, the action a_c is decided, that is, T_{it} is sent to the Cloud for execution. Our methodology is shown in Algorithm 1.

IV. Experimental Evaluation

We report on the performance of the proposed methodology and provide a sensitivity analysis on several performance metrics. We investigate whether the proposed in-edge network methodology is capable of *efficiently deciding on local task executions or transferring such responsibility to edge neighbouring nodes and/or Cloud.*

Algorithm 1 Task Decision Making Methodology on Node n_i

```
while task $T_{it} \in \mathcal{T}_t$ do
    Calculate context vector $\mathbf{v}_i$
    Receive information vectors $\mathbf{p}_{ij}; n_j \in \mathcal{N}_i$
    if $g(\mathbf{v}_i) = \{a_l\}$ then
        execute $T_{it}$ at node $n_i$
    else
        Calculate criterion vector $\mathbf{q}$ and scores $z_j = \mathbf{q}^\top \mathbf{p}_{ij}, \forall n_j$
        $z_{j^*} \leftarrow \arg\max_{n_j \in \mathcal{N}_i}(z_j)$
        if $\mathbf{p}_{ij^*} \succ \boldsymbol{\theta}$ then
            send $T_{it}$ to node $n_{j^*} \in \mathcal{N}_i$
        else
            send $T_{it}$ to Cloud
        end if
    end if
end while
```

A. Performance Metrics

We define the performance metrics over the context vector dimensions $\mathbf{v}_i = [\lambda_i, \pi_i, \rho_i]$ for load, task priority, ratio of remaining available resources, and task requirements along with the information vectors $\{\mathbf{p}_{ij}\}$ from neighbouring nodes. We focus on three aspects for evaluating the performance of our methodology. Specifically, for each node n_i, we:

- assess the correct identification of tasks that should be executed locally, i.e., given a specific task T_{it}, the current context vector $\mathbf{v}_i$ and the information vectors $\mathbf{p}_{ij}, n_j \in \mathcal{N}_i$, the methodology *correctly* concludes on the action a_l and not on the actions $\{a_n, a_c\}$.
- assess the correct identification of the appropriate neighbouring node $n_{j^*} \in \mathcal{N}_i$ to execute T_{it} given that the methodology correctly does not decide on a_c and it is not possible node n_i to execute T_{it} locally (not considering the action a_l).
- assess the 'closeness' of the proposed methodology to the optimal decision given that there would exist an *ideal node* in $n_i's$ neighbourhood, which should have been selected to task execution. We quantify this closeness as the distance (norm) of the context and information vectors from these corresponding to the ideal node, as will be elaborated later.

For the first two aspects, we adopt the widely known metrics *precision* P, *recall* R, *F-measure* F, and accuracy ϵ defined as: $P = \frac{TP}{TP+FP}$, $R = \frac{TP}{TP+FN}$, $F = 2\frac{PR}{P+R}$, and $\epsilon = \frac{TP+TN}{TP+TN+FP+FN}$, where TP refers to True Positive events, i.e., correctly decided actions that had to be identified (e.g., the methodology decides on a_l and this is the correct action w.r.t. context and information vectors), FP refers to False Positive events, i.e., actions that have not been correctly decided (e.g., the methodology decides on a_l but either action a_n or action a_c should have been decided), TN refers to True Negative events, i.e., incorrectly decided actions that had been identified, FN refers to False Negative events, i.e., actions that had to not been incorrectly decided.

For the third aspect, we assess the *appropriate* selection of the node n_{j^*} when node n_i *correctly* decides on the action a_n, i.e., the task should be transferred to the neighbourhood of n_i. In this context, we introduce the concept of the ideal node $n_\ell \in \mathcal{N}_i$ which is considered as the most appropriate (best) node for being assigned T_{it} given that n_i decides on the action a_n. Such node appropriateness is represented by the distance of the information vector $\mathbf{p}_{i,\ell}$ regarding the ideal node n_ℓ from the information vector $\mathbf{p}_{i,j^*}$ of the selected node n_{j^*} regarding the VIKOR optimal selection process in Section III. The information vector $\mathbf{p}_{i,\ell} = [\lambda_\ell, \kappa_{i\ell}, \delta_{i\ell}]$ of the ideal vector n_ℓ is constructed from (i) the lowest load $\lambda_\ell = \min_{n_j \in \mathcal{N}_i}\{\lambda_j\}$, (ii) the lowest communication cost $\kappa_{i\ell} = \min_{n_j \in \mathcal{N}_i}\{\kappa_{ij}\}$, and (iii) the closest dataset statistical distance $\delta_{i\ell} = \min_{n_j \in \mathcal{N}_i}\{\delta_{ij}\}$. Note that, $\lambda_\ell, \kappa_{i\ell}, \delta_{i\ell}$ do not necessarily correspond to the same neighbouring node from $\mathcal{N}_i$. It would be the ideal case to have the node n_ℓ in the neighbourhood of n_i, thus, immediately n_i would assign its task to node n_ℓ. However, this is not happening in reality. Hence, we assess how 'closely' the proposed methodology w.r.t. action a_n selects the node n_{j^*}, which is as close to the ideal node as possible w.r.t. information vector, i.e., the best possible decision we can obtain. We quantify this *closeness* by the Euclidean distance: $\omega_i = \|\mathbf{p}_{i,\ell} - \mathbf{p}_{i,j^*}\|$, $n_{j^*} \in \mathcal{N}_i$, in the information vectors space and desire $\omega_i \to 0$, i.e., the selected node n_{j^*} to be the best possible for selection, thus, enjoying the optimality achievement of our methodology. The metric ω represents the difference with the best node in the neighbouring network per task execution request as depicted by its computational, networking, loading and statistics attributes.

B. Experiment Setup

We adopt **(i)** a real dataset related to the prediction of companies bankruptcy from qualitative parameters as predicted by experts[1] to evaluate the classification 'power' of our methodology, i.e., if our methodology correctly classifies the context vectors to the best actions, and **(ii)** the real dataset[2] from the Intel Lab containing 3,000 contextual 4-dimensional data vectors $\mathbf{x} = [\text{temperature, humidity, light, voltage}]^\top$. The dataset is becomes the basis for each dimension of the data stored in every node in the network. From the decision making perspective, we train the probabilistic local decision making methodology using a training set of pairs: (context vectors, best actions), i.e., $\mathcal{I}_i = \{(\mathbf{v}_k, a_k)\}_{k=1}^K$ for each node n_i, $k = 1, \ldots, K = 300$. The training dataset $\mathcal{I}_i$ provides various combinations of the aforementioned vector parameters accompanied by the appropriate actions, i.e., the corresponding classes. By analyzing the class labels, approximately 65% of the training pairs are classified in action a_l, i.e., $P(a_l) = 0.65$ (35% of pairs are classified as $\bar{a}_1$). Node n_i decides on a_l if $P(a_l|\mathbf{v}_i) > P(\bar{a}_1|\mathbf{v}_i)$, while the classifier is trained with 10-fold cross validation having a split of 60% of the dataset as the training set $\mathcal{I}$ and 40% at the testing set in each node. If action a_l is not the outcome of the probabilistic decision maker, then the actions a_n and a_c are decided w.r.t. VIKOR methodology. A 'typical' running example is as follows: **(i)** the production of a contextual vector at a random node (data are retrieved by the aforementioned datasets); **(ii)** at pre-defined intervals, the node sends its information vectors to the neighbourhood; (iii) the execution of the decision function to reveal if the action a_l should be chosen; **(iv)** if a_l is selected, we store the incoming vector and update the dataset; otherwise, we apply the VIKOR methodology and get the rankings of the neighbours; **(v)** we get the maximum score and if this score is over the pre-defined threshold, we select the action a_n and update the dataset of the selected peer node; **(vi)** if a_n is not selected, T_{it} is sent to the Cloud.

We construct random neighbouring networking topologies of 5,000 nodes, where each node n_i has a neighbourhood size $|\mathcal{N}_i| = \{10, 50, 100, 1000\}$ nodes. The 'randomness' in the topology of the network does not affect the proposed model and is adopted to 'spread' the nodes in the area under consideration. The communication cost $\kappa_{ij} \sim U(0,1)$ is uniformly distributed for $e_{ij} \in \mathcal{E}$, the task priority per node n_i is drawn uniformly $\rho_i \sim U(0,1)$. The uniformity of the task priority gives the necessary randomness in the characteristics of tasks arriving at any node in the network. Based on this, we investigate how the number of neighbouring nodes in the edge network affects the results of our decision making mechanism. We expect that the higher the number of nodes, the more the options of selecting a node which can be assigned tasks, thus, avoiding transferring the task to the Cloud, i.e., a_c. However, on the other hand, this potentially increases the expected load of neighbouring nodes, if they are assigned relatively a huge number of tasks for possible execution. Hence, the decision on action a_c is not entirely avoidable, but it is considered as a last resort after each neighbourhood can successfully satisfy the assigned task executions. We also examine the importance of each criterion dimension $w_k \in (0,1)$, $\sum_k w_k = 1$, in the VIKOR method and investigate their impact on the closeness of our methodology to the best decision w.r.t. action a_n. Hence, we consider four experimental scenarios with different importance weight distributions on the criteria: *load*, *communication cost*, *remaining available resources*, and *dataset statistical distance* (data similarity):

TABLE I
WEIGHTS IMPORTANCE DISTRIBUTION OVER CRITERIA.

Scenario	load (λ)	comm. (κ)	resources (ρ)	distance (δ)
Scenario A	0.25	0.25	0.25	0.25
Scenario B	0.70	0.10	0.10	0.10
Scenario C	0.10	0.10	0.40	0.40
Scenario D	0.10	0.10	0.10	0.70

Through the aforementioned scenarios, we pay attention on different parameters when selecting nodes for transferring a task. Scenario A pays equal attention on all criteria while Scenario B pays more attention on the load of each node.

[1]https://archive.ics.uci.edu/ml/datasets/qualitative_bankruptcy
[2]http://db.csail.mit.edu/labdata/labdata.html

Scenario C pays more attention on the remaining resources and dataset statistical distance, while Scenario D pays attention on the similarity of data in terms of statistics that nodes have collected from their environment.

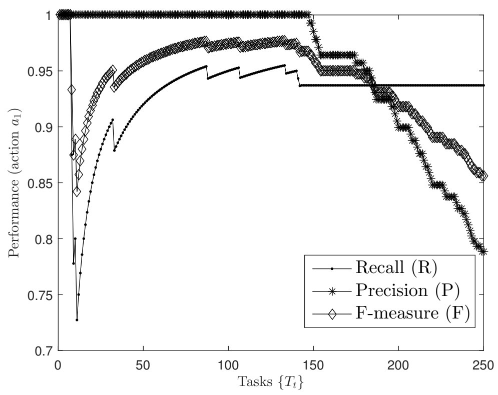
C. Performance Evaluation

Firstly, we evaluate the proposed mechanism concerning the decision of the local task execution in all edge nodes, i.e., concerning the action a_l. For a workload of $|\mathcal{T}| = 250$ tasks per node, we obtain an average precision $P = 78.8\%$, average recall $R = 93.7\%$, average F-measure $F = 85.6\%$ and expected accuracy $\epsilon = 90.71\%$ (deviation: 0.047). Such statistics indicate that our methodology is capable of identifying when a task should be locally executed. The classifier manages to eliminate false negative events, thus, we enjoy an increased R. However, we observe that there is room for reducing false positives, thus, we could enjoy P close to unity. The expected accuracy per node ϵ is also at high levels indicating the certainty of a node when concludes on action a_l, thus, avoiding further communication with its neighbors.

Figure 1 (upper) shows the evolution of P per node aligned with the tasks stream $T_{i1}, T_{i2}, T_{i3}, \ldots$. One can observe that for the first 150 tasks, the precision is equal to 1.0 (100%). This is because in the test dataset, the local tasks execution events are stored in the first places of the tasks stream and the proposed scheme correctly identifies them. Afterwards, the proposed scheme decides that the incoming tasks should not be executed locally. We observe that P decreases as false positives are identified. In any case, all the tasks that should be locally executed are correctly identified obtaining high accuracy and precision. Moreover, Figure 1 (upper) illustrates the evolution for the recall per node. The recall is characterized by stability after the allocation of the first 150 tasks. Figure 1 (lower) illustrates the probability density function of the accuracy metric regarding the action a_l indicating that the node n_i correctly concludes on a_l and rejects the actions a_n and a_c provided the current context vector.

We now assess the proposed methodology regarding the decision on actions a_n and a_c given that the action a_l is rejected by the node. Our methodology decides to execute locally 50% of the total tasks with average node load $\mathbb{E}[\lambda] = 0.325$ and average ratio of remaining resources $\mathbb{E}[\rho] = 0.459$. In this context, the average node load deciding on locally executing the tasks is relatively low, while the remaining resources are at a medium level. Moreover, 25% of the tasks on average were assigned to neighbouring nodes (action a_n), while 25% of the tasks were transferred to Cloud (action a_c).

Figure 2 (lower) shows the $\boldsymbol{\theta}$ normalized threshold vector space (for the criteria: dataset similarity, load, communication cost) for the VIKOR method and the corresponding percentages of the tasks assigned and executed: locally, in the neighborhood, at the Cloud. Based on the VIKOR decision making, the criteria thresholds θ_k were set to (0.4, 0.4, 0.2) for the criteria dimensions λ_j, κ_{ij} and δ_{ij}, respectively; this is the methodology that selects the node with the highest score u_{j^*} iff $\lambda_{j^*} \leq 0.4$, $\kappa_{ij^*} \leq 0.4$ and $\delta_{ij^*} \leq 0.2$. We also



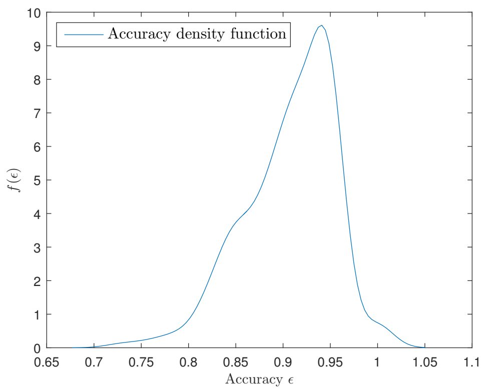
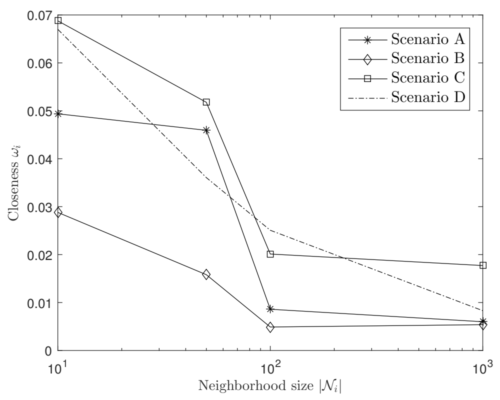


Fig. 1. (Upper) Precision, recall, and F-measure against tasks stream for action a_l; (lower) probability density function $f(\epsilon)$ of the accuracy in concluding on action a_l.

experiment with other threshold values. Specifically, with θ_k values defined in (0.455, 0.455, 0.05), we observed that no task will be executed in any neighbouring node, thus, 80% of the tasks were to be transferred to the Cloud (and 20% locally). In addition, with θ_k values defined in (0.33, 0.33, 0.33), on average 30% of the tasks would have been executed in the neighbourhoods and 20% in the Cloud. Hence, to avoid having many tasks transferred to the Cloud and avoid not assigning certain tasks in the neighbourhoods, we select the (0.4, 0.4, 0.2) normalized threshold values for the VIKOR method regarding the action a_n.

We also plot the closeness metric ω_i in Figure 2 (upper) demonstrating that in all scenarios the size of the neighbourhood results to the selection of the most appropriate node for tasks assignments in all the considered criteria (load, communication cost, dataset similarity). Hence, the proposed methodology efficiently and successfully finds the best neighbouring node upon concluding on the action a_n.

We further assess the behaviour of our methodology based on the closeness to the optimal (ideal) solution regarding action a_n. Specifically, we assess the closeness of our methodology concerning the selection of the appropriate neighbouring node for executing a task w.r.t. the ideal node as described in Section III. Recall that the ideal node is characterized by the lowest load, lowest communication cost and closest distance



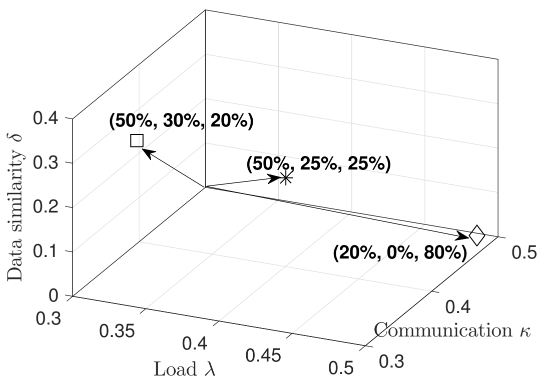
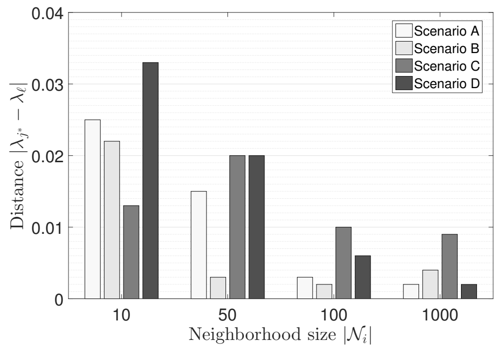


Fig. 2. (Upper) The closeness metric ω_i from the ideal node against neighborhood size for all scenarios; (lower) the $\boldsymbol{\theta}$ threshold space in the VIKOR method along with the percentages of tasks executed locally, in the neighbourhood, or at the Cloud.

of the collected data (all these, at the same time) in a specific neighbourhood.

Figure 3 (upper) presents our results concerning the distance (absolute difference) of the load parameter $|\lambda_{j^*} - \lambda_\ell|$ between λ_{j^*} of the selected node n_{j^*} and the lowest load λ_ℓ of the ideal node versus the size of the neighbourhood $|\mathcal{N}_i|$ for all scenarios (Table I). We observe that the neighbourhood size influences the final difference since an increased number of neighbouring nodes leads to more opportunities for assigning tasks. This stands for all the examined scenarios. The difference is below 0.04 (4%; quantities are normalized in (0,1)), which depicts the efficiency of the proposed scheme. The performance of our mechanism increases as tasks are allocated to nodes with low load, thus, no node is expected to be overloaded creating a *bottleneck* in the neighborhood. One can also observe that when $|\mathcal{N}_i| \in \{50, 100\}$, the distance from the optimal load is less than 1% especially in Scenario B. Note that, in Scenario B we pay increased attention on the load of the nodes, which this is reflected by our methodology for finding the most appropriate neighbouring node.

Regarding the dataset statistical relevance in terms of the sufficient statistics distance δ_{ij^*} and from the corresponding ideal node $\delta_{i\ell}$, Figure 3 (lower) illustrates the absolute difference $|\delta_{ij^*} - \delta_{i\ell}|$ for all scenarios and neighbourhood size. Similarly to the aforementioned results, an increased number of nodes positively affects the results. When $|\mathcal{N}_i| \to 1000$, our methodology exhibits the best performance resulting to a distance from the ideal case close to zero. This indicates that our scheme assigns tasks to the most similar dataset.



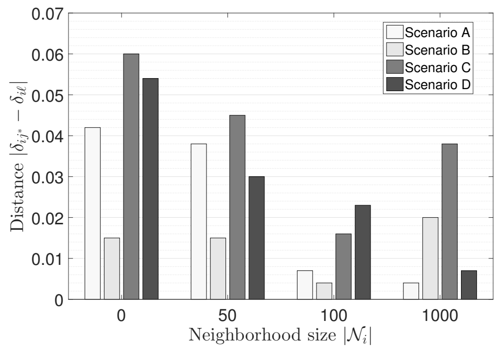


Fig. 3. (Upper) Distance of the load criterion and (lower) distance of the dataset statistical relevance from the ideal node/decision against neighborhood size for all scenarios.

In addition, we perform experiments to measure the difference in the communication cost of the selected neighbouring node compared to the optimal (ideal) communication cost in the neighbourhood. The distances $|\kappa_{i\ell} - \kappa_{ij^*}|$ for all scenarios against different neighbourhood sizes are presented in Table II. In the majority of the cases, such differences are close to zero. Again, a limited number of selected neighbouring nodes may lead to high values for such difference, thus, our methodology requires a relatively high number of nodes to efficiently decide on the most appropriate node for transferring the incoming tasks.

TABLE II
AVERAGE DIFFERENCE IN THE COMMUNICATION COST.

$|\mathcal{N}_i|$	$|\kappa_{i\ell} - \kappa_{ij^*}|$			
	Scenario A	Scenario B	Scenario C	Scenario D
10	0.007	0.011	0.031	0.022
50	0.021	0.004	0.016	0.001
100	0.004	0.002	0.007	0.008
1000	0.004	0.009	0.018	0.004

Based on the above, we can safely conclude on the efficiency of the proposed scheme as it can select the best possible node based on the entire set of parameters and not on a single one.

Expected Overhead: Our methodology mainly aims to stochastically avoid the migration of the data $\mathcal{X}$ that should be transferred from the edge network to the Cloud. Instead of migrating data, the methodology intelligently decides on assigning tasks to neighboring nodes over *statistically similar* data residing on different edge nodes, if the option of local task execution is rejected. The expected overhead in terms of messages sent within the edge network relates to **(i)** tasks transfer/assignment in action a_n and **(ii)** data transfer from the node to the Cloud in action a_c, i.e., the overhead is faced only when the task itself is decided to be relocated.

Let $p_1 = P(a_l|\mathbf{v}_i)$ and $p_2 = P(\mathbf{p}_{ij^*} \succ \boldsymbol{\theta}|\bar{a}_l)$ be the probability of selecting the action a_l for local task execution (given context vector $\mathbf{v}_i$) and the probability of assigning a task to neighbouring node $n_{j^*} \in \mathcal{N}_i$ as inferred from the VIKOR method given that action a_l is rejected, respectively. In the former case, the expected communication overhead is zero, while in the latter case, node n_i should first collect the information vectors $\mathbf{p}_{ij}$ and then assign the task to the selected n_{j^*}, thus, incurring $|\mathcal{N}_i| + 1$ communication messages. The expected overhead for both actions ensuring task execution in the network edge is then $\sum_{n_i \in \mathcal{N}}(1 - p_1)p_2(|\mathcal{N}_i| + 1)$, which depends linearly on the neighbourhood side, as expected. Concerning the action a_c, we face the case as in the baseline/centralized approaches, where the node n_i decides to migrate the data $\mathcal{X}_i$ to the Cloud. In the centralized approaches this happens with probability 1, while in our case, this occurs with probability $(1-p_1)(1-p_2)$. Indicatively, based on our experimental setting and results, this corresponds to maximum 25% of the cases on average, which denotes the communication gain of our methodology by pushing the task execution at the network edge taking into consideration the nodes' context and data relevance.

V. Conclusions

Recent advances in edge computing involve the execution of analytics tasks close to the sensing/computing devices challenging with the limited computational capabilities. Our methodology faces with the problem: *which tasks each node should execute to avoid data migration and maximize the expected performance*. We contribute with an efficient, intelligent, distributed scheme adopted by every node that pushes the task allocation to the edge providing a two-level decision making mechanism. Our scheme sequentially examines possible actions to efficiently decide the place where tasks are executed: locally, in the node neighborhood, and at the Cloud, considering context and information vectors. We evaluate our scheme with real data and the experimental evaluation shows that the selected nodes for task allocation are among the best possible solutions minimizing the average difference. Our future agenda involves a time-optimized scheme for handling the uncertainty in the decision making process.

Acknowledgment

This work is funded by the EU/H2020 Marie Sklodowska-Curie (MSCA-IF-2016) under the INNOVATE; Grant#745829.

References

[1] C. Anagnostopoulos, P. Triantafillou. 'Query-Driven Learning for Predictive Analytics of Data Subspace Cardinality' ACM TKDD, 11(4), 2017.

[2] N. Harth, C. Anagnostopoulos, D. Pezaros, 'Predictive intelligence to the edge: impact on edge analytics', Evolving Systems, 9(2):95–118, June 2018.

[3] Awadalla, M. H. A., 'Task Mapping and Scheduling in Wireless Sensor Networks', JCS, 440(4), 2013.

[4] De Benedetti, M., Messina, F., Pappalardo, G., Santoro, C., 'JarvSis: A Distributed Scheduler for IoT Applications', Cluster Computing, 20:1775–1790, 2017.

[5] N. Harth, C. Anagnostopoulos, 'Edge-centric Efficient Regression Analytics'. IEEE EDGE, 2018.

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

[7] Dai, L., Chang, Y., Shen, Z., 'An Optimal Task Scheduling Algorithm in Wireless Sensor Networks', JCCC, 1:101–112, 2011.

[8] 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 Computer Communication Review, 45(5), 2015.

[9] Han, J., Kamber, M., Pei, J., 'Data Mining, Concepts and Techniques', Morgan Kaufmann Publishers, 2012.

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

[11] Krishnapriya, S., Joby, P. P., 'QoS Aware Resource Scheduling in Internet of Things-Cloud Environment', Scientific & Engineering Research, 6(4), 2015.

[12] Moschakis, I., Karatza, H., 'Towards Scheduling for Internet-of-Things Applications on Clouds: A Simulated Annealing Approach', Concurrency and Computation, 27(8):1886–1899, 2015.

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

[14] Razavinegad, A., 'Task Allocation In Robot Mobile Wireless Sensor Networks', IJSTR, 3(6), 2014.

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

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

[17] Sayadi, M. K., Heydari, M., Shahanaghi, K., 'Extension of VIKOR Method for Decision Making Problem with Interval Numbers', Applied Mathematical Modelling, 33(5):2257–2262, 2009.

[18] Shanthan, BJ., Kumar, A. D. V., Govindrajan, E., Arockian, L., 'Scheduling for Internet of Things Applications on Cloud: A Review', Imperial Journal of Interdisciplinary Research, 3(1), 2017.

[19] Sebastio, S., Gnecco, G., Bemporad, A., 'Optimal distributed task scheduling in volunteer clouds', Computers & Operations Research, 81:231–246, 2017.

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

[21] Xu, J., Palanisamy, B., Ludwig, H., Wang, Q., 'Zenith: Utility-Aware Resource Allocation for Edge Computing', IEEE EDGE 2017.

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

[23] Yu, Y., Wang, A., 'Extension of VIKOR method for multi-criteria group decision making problem with linguistic information', Applied Mathematical Modelling, 37(5):3112–3125, 2013.

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