Optimizing Data Analytics in Energy Constrained IoT Networks

Apostolos Galanopoulos *, George Iosifidis* and Theodoros Salonidis†

*School of Computer Science and Statistics, Trinity College Dublin, Ireland
†IBM T.J. Watson Research Center, Yorktown Heights, NY, USA

***Abstract*—The emergence of delay sensitive and computationally demanding data analytic applications has burdened the core network with huge data transfers and increased computation load. Furthermore, the increasing number of Internet of Things deployments rely significantly on the execution of such applications. We propose an architecture where devices collaboratively execute data analytic tasks in order to improve their execution delay and accuracy. This is possible by exploiting the aggregate computation capabilities of the abundance of small devices. We design an optimization framework where the nodes decide where their data analytic tasks will be executed, in order to jointly optimize their average execution delay and accuracy, while respecting power consumption constraints. We propose a distributed dual ascent solution to the formulated convex problem, so that the nodes can make the outsourcing decisions by exchanging local information. The results indicate that the nodes can achieve better performance when collaborating than when they locally compute the tasks, depending on the network load.**

I. Introduction

A. Background

The Internet of Things (IoT) aspires to deliver a new set of services by integrating large numbers of heterogeneous devices in proximity with the end-users [1]. Among the most promising IoT services are those which collect and process data in order to make some type of inference or decision, often in real time. Indeed, today we see an increasing number of IoT scenarios that involve data analytics [2]. From interconnected sensors which detect various events or system operation anomalies, to IoT cameras performing face recognition or any other machine learning task, there is a plethora of suggestions (even deployments) for IoT-based analytics.

These services are as challenging to implement as they are important. On one hand, they create a huge amount of data that is very costly to transport to distant cloud servers where the analytics can be effectively executed. Large portions of this collected data is ephemeral with negligible payload and should ideally be filtered at the edge [2], [3]. On the other hand, these services require heavy-duty computing, and most often prior information that might require preprocessing and significant memory space (e.g., a large training dataset is needed for face recognition apps). Hence, implementing them at the IoT nodes, which are usually energy-constrained with limited computation capabilities, might induce significant performance degradation, e.g., low accuracy or high error rate.

A strong candidate solution for this problem is to leverage resources that are available at the edge, namely at nearby IoT devices or even at small edge servers (Fig. 1a), a concept also known as fog computing [2], [3]. For example, an IoT camera that performs face recognition, could transmit some of the captured pictures to another camera in range, in order to expedite the task execution or improve its accuracy (if the other camera has, say, a larger training dataset). This approach comes with several advantages: it is scalable, reduces data transfer costs and task completion times, satisfies privacy criteria since the data is not transferred to the cloud, and so on.

Nevertheless, enabling a group of resource-limited IoT devices to jointly execute their analytic tasks is an intricate goal [4]. Some devices might be able to execute certain functions with higher accuracy, but might not have enough computation resources to support all nearby devices. It is therefore difficult to decide where to execute the tasks, even more since their execution is often delay-sensitive. Clearly, there is here an inherent trade off between accuracy and task execution delay. On top of that, the IoT nodes usually have limited communication capabilities, e.g., small transmission range, and tight energy constraints (average power limitations). Hence, they need to wisely decide how to allocate their power budget among computations, data collection and transmissions.

Our goal in this work is to address the above challenges by *designing a framework for the collaboration of energy-constrained IoT devices which jointly optimize the execution delay and accuracy of their analytic tasks*.

B. Related Work

Mobile computation offloading. Existing mobile cloud offloading frameworks partition demanding mobile apps and offload tasks to the cloud [5], [6]. These techniques however cannot easily support real-time applications due to large delays between the users' devices and the cloud. Cloudlets reduce delay by bringing server infrastructure at the network edge, closer to mobile and IoT devices [2], [7], [8]. Such solutions however need to be pre-deployed in locations where a specific real-time application is needed. In contrast, we consider a collaborative computing model where proximal IoT devices can spontaneously execute data analytics.

Distributed collaborative mobile computing. Another class of solutions relies on collaborative execution of the tasks. Misco [9], and CWC [10] implement frameworks for parallel task execution on phones, catering to MapReduce-like batch processing models rather than tasks modeled as data streams. On the other hand, MobiStreams [11], Swing [12] and [13] enable collaborative computations on data streams. Nevertheless, these systems either do not optimize the task distribution [11], or use heuristic policies that do not cater for energy consumption or execution accuracy [12], [13].

Serendipity [14] focuses on distributed computing in mobile networks with intermittent connectivity, and proposes a greedy task allocation scheme to minimize completion time. However, it does not take into account accuracy or energy constraints, and does not provide optimality guarantees. Contrary to these important prior works, we introduce a multi-objective (for delay and accuracy) optimization framework and devise a provably-optimal cooperation policy.

Energy-aware Solutions. Energy constraints in IoT systems are very important and several prior works have proposed solutions for reducing their power consumption. In [15], for example, the authors propose a dynamic algorithm that minimizes energy costs by leveraging green energy sources; and similarly, [16] and [17] propose cooperative solutions that reduce energy costs. Contrary to our approach, these prior works do not consider data analytics (and the respective collaboration constraints) nor minimize the task execution delay. On the other hand, [8] decides the task offloading based on delay only, but does not account for the power consumption or the execution accuracy, which perhaps is the most important factor for such services.

C. Contributions

In this work, we introduce an optimization framework for devising a data analytic task execution policy in an energy-constrained IoT environment. Our policy determines how the tasks of each node are distributed to the network for execution to obtain optimal performance. The framework allows the policy designer to balance execution time and accuracy, based on the system's priorities (or, mission). Our model is detailed: it captures transmission and reception energy costs that might vary across nodes; caters for the inherent heterogeneity in link capacities; and uses a generic model for the task execution accuracy that covers a rich set of data analytic scenarios. We place emphasis on delay, which is apparently very crucial for this problem, and use a load-dependent delay metric, instead of average constant delay parameters.

In order to solve the resulting challenging mathematical program, we rely on Lagrange duality which effectively decomposes it to a set of subproblems (one per node). Furthermore, we use a primal-dual algorithm to allow the coordination of these subproblems in a distributed and scalable fashion. Indeed, we prove that our solution finds the optimal cooperation policy with minimal exchange of messages, involving only one-hop neighbours. Finally, we conduct a trace-driven evaluation of our solution, using measurements from a small testbed of Raspberry Pi 3 devices which run a Java-based face recognition app. Our experimental analysis reveals the potential benefits of such cooperative IoT solutions, and provides interesting insights for the arising trade offs. The contributions of this work can be summarized as follows:

- We introduce the problem of jointly optimizing the average execution delay and accuracy of data analytic tasks in IoT networks. Our model is generic in the sense that it captures a variety of network parameters.
- Our analytical multi-objective framework enables the designer to prioritize one objective over the other, based on the system's requirements.
- We design a scalable algorithm for solving the problem in a distributed fashion. Our algorithm requires minimal exchange of messages among only neighbouring nodes to converge to the optimal solution.
- We conduct an extensive trace-driven evaluation of our optimization framework, showing that the collaborative solution can be up to 60% faster and 23% more accurate than local task execution.

The rest of this paper is organized as follows. In Sec. II we introduce the system model and state formally the problem. In Sec. III we formulate and solve the optimization problem using a novel distributed algorithm. In Sec. IV we present an experimentally-driven evaluation of the proposed system and solution, and we conclude in Sec. V.

II. System Model and Problem Statement

A. System Model

Network. We consider a wireless IoT network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ of N nodes and L links. The nodes are IoT devices of low resource footprint. We assume they are equipped with half-duplex radios, hence each node can transmit to, or receive from at most one other node at each time instance. They also operate with existing IoT technologies such as Wi-Fi, Bluetooth or ZigBee. Each link $l_{i,j} \in L$, if node j is within communication range of node i. All nodes within range of node i, including i, are called its neighbours and are denoted by the set $\mathcal{N}_{(i)}$, which contains N_i nodes. Each link $l_{i,j}$ is characterized by its average capacity $c_{i,j}$ measured in bits/s. We assume that $c_{i,j}$ is measured by higher layer capacity estimation mechanisms [18] and accounts for channel fluctuations on the link, and MAC layer scheduling[1].

Each link $l_{i,j}$ is characterized by its average (Tx/Rx) energy consumption $e^t_{i,j}(c_{i,j}),\ e^r_{i,j}(c_{i,j})$ measured in Joules/bit, for transmission and reception respectively. We assume that energy consumption is dominated by the transmission at node i over (the attenuated) reception at node j. In general, energy consumption depends on the link capacity $c_{i,j}$ since it is affected by the channel quality and the transmitters' bit rate [19]. We assume that the nodes experience a certain average bit rate with each of their neighbours and, for simplicity, use $e^t_{i,j}$ and $e^r_{i,j}$ do denote the resulting energy consumption. We consider the general case where $c_{i,j} \neq c_{j,i}$, $e^t_{i,j} \neq e^t_{j,i}$, $e^r_{i,j} \neq e^r_{j,i}$ since nodes i and j may have different wireless adapters.

Computing and Data Analytics. We assume that $\mathcal{N}_T \subseteq \mathcal{N}$ is a subset of the network nodes that need to execute computationally intensive data analytic tasks. The tasks may correspond to two types of compute models. In a data stream processing model the tasks are data (e.g., images) that are sent to operator function modules (e.g., face recognition) deployed at the nodes. Alternatively, in a SPARK/MapReduce model the tasks may consist of operators that are sent to process data deployed at the nodes. Our model captures both scenarios.

Each node $i \in \mathcal{N}_T$ generates tasks for processing at a rate of λ_i tasks/sec. The tasks generated by each node i, are characterized by the amount of average input data S_i in

[1]Our focus is on data analytic task scheduling techniques at higher layers that are MAC agnostic. Optimizing MAC scheduling for IoT systems is an orthogonal problem that is beyond the scope of this paper.

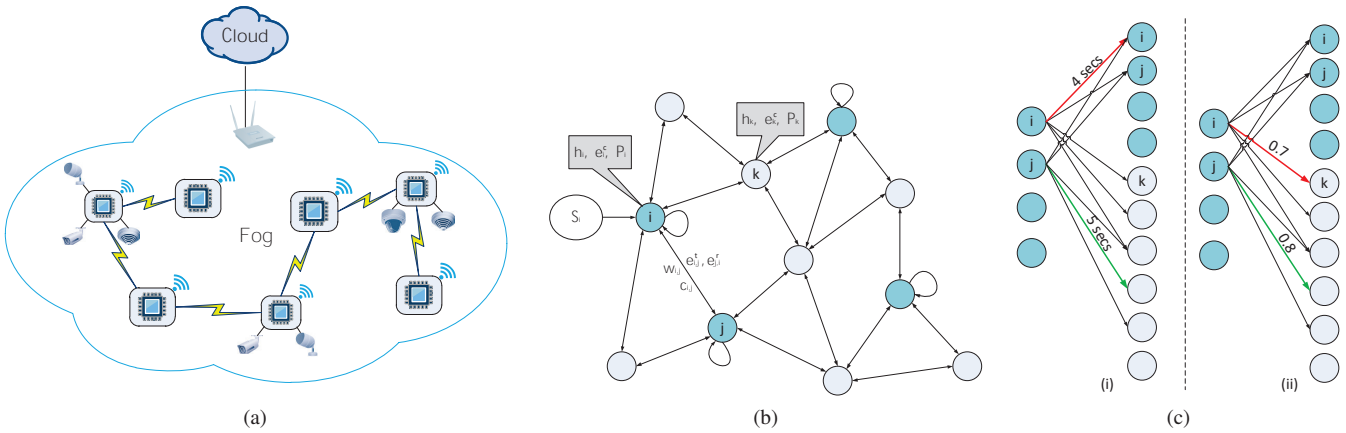

Fig. 1: (a) The IoT environment where devices have embedded different types of sensors. The computationally demanding applications that arise in such scenarios can be efficiently handled in the fog level by exploiting the increasing computing capabilities of the devices. (b) Example of network graph where the nodes with data analytic task arrivals are highlighted with blue color. (c) Network graph representation, where i and j's optimal decisions regarding (i) execution delay and (ii) execution accuracy, are highlighted with red and green for i and j respectively. Optimizing delays and execution accuracies may result to different (in case of i) or similar (in case of j) execution decisions.

bits.[2] Each node $j \in \mathcal{N}$ requires a number of ρ_j cycles to execute each task, depending on the algorithm it uses, based on its CPU frequency h_j. For example, in a face recognition application, node i is a camera that captures images of size S_i with an average rate of λ_i images/sec, and computing node j uses a specific recognition algorithm that yields a processing capacity of h_j/ρ_j tasks/sec. Combining the task arrival rate with its input size, we define each node's data generation rate $R_i = S_i\lambda_i$ (b/s). Furthermore, each node j has an average power budget of P_j Watts, which can be used for processing or data transfers. Such power budgets arise in IoT networks that employ energy harvesting mechanisms [17] or comprise small form-factor power-constrained devices. In addition, the nodes have energy requirement when performing computations e_j^c (Joules/cycle). An annotated view of the model is depicted in Fig. 1b.

Each data analytic task has a metric that determines the quality of its outcome. In this paper we focus on machine learning and predictive analytic tasks where the metric is the accuracy of the prediction. The output accuracy of the data analytic tasks depends on the executing node. We define by $w_{i,j} \in [0,1], \forall i \in \mathcal{N}_T,\ j \in \mathcal{N}_{(i)}$ the normalized output accuracy of node j when it executes a data analytic task requested by i. Consequently, when node i executes locally its own tasks, the execution accuracy is denoted by $w_{i,i}, \forall i \in \mathcal{N}_T$. Thus the notation $w_{i,j}$ is used to implicitly express j's efficiency to execute i's tasks, and not j's efficiency in general. In a face recognition application for example, the nodes can use different classifiers or different training data sets [20], which will yield different execution accuracy in the classification task. Note that our model is very generic as it allows performance to depend on the node that generated the tasks, e.g. the image resolution or the surrounding environment of the capturing camera-node may impact the accuracy of the classification.

[2]We assume that task properties are node-specific, e.g., in a face recognition application it is the camera configuration of each node that determines the size of the captured frames. However, our model can be directly extended to include a set of task classes, each one with different data sizes, computation load, and arrival rate.

B. Problem Statement

We design our system with the aim to devise a data analytic policy which will: *a)* minimize the overall execution delay, and *b)* maximize each node's average task execution accuracy. To this end, we formulate a rigorous mathematical program with a multi-objective optimization metric, that respects power budget, link and computation capacities in the IoT network. The solution of this problem determines the portion of tasks that the nodes will outsource to each of their neighbours. This is an interesting problem where several trade offs arise.

The reasons that outsourcing can improve the overall performance are multiple. The tasks arriving at the nodes can be too demanding with respect to the required computational capacity to be supported by their power budgets. In addition, these neighbouring nodes may be more specialized in executing these tasks, so outsourcing can increase execution accuracy. The additional communication delay when choosing to execute at another node can be large, however there can be some neighbouring nodes with large link capacity and powerful processing speed that can render outsourcing faster than local-only execution.

It is interesting to note that minimizing the execution delay and maximizing execution accuracy are *not* necessarily conflicting objectives. In some cases decreasing delays by routing tasks to good channel quality neighbours can lead to increasing execution accuracy, namely if the selected neighbours are very good at executing the requested tasks. In other cases however, it is impossible to optimize one criterion without degrading the other. The latter cases are obviously more interesting, since the produced solution will be determined by the importance of each objective.

To further illustrate the above, consider a representation of the network graph where task requesting nodes included in $\mathcal{N}_T$ are on one side and are connected to all the other nodes in $\mathcal{N}$ based on (i) average delay and (ii) average execution accuracy (Fig. 1c). For j, no matter how important each single objective is, the ideal decision is to outsource. In i's case however, local execution is better in terms of delay minimization, but

executing at k is optimal regarding execution accuracy. In this case, the routing decision will be significantly affected by the relative priority of the two objectives.

III. Problem Formulation and Solution

A. Problem Formulation

Let us define variable vector $\boldsymbol{x} = [x_{i,j} \in [0,1], \forall i \in \mathcal{N}_T,\ j \in \mathcal{N}]$. Its values denote the execution policy of the nodes so that $x_{i,j}$ represents the portion of tasks a node i outsources to some neighbouring node j. As a result, $x_{i,i}$ denotes local execution. In addition, we define a sub-vector of $\boldsymbol{x}$ as $\boldsymbol{x_i} = [x_{i,1}, x_{i,2}, \dots x_{i,N}]\ \forall i \in \mathcal{N}$, where $N = |\mathcal{N}|$ that contains the execution policy variables related to node i. The average execution delay per task involves two parameters, i.e., the communication delay for input/output data transfer and the computation delay.

In our analysis we consider load-dependent queuing and fixed transmission delays. Namely, the queuing delay $d^Q_{i,j}$ is:

$$d^Q_{i,j} = \frac{S_i}{c_{i,j} - x_{i,j} R_i},\ \forall\ i \in \mathcal{N}_T, j \in \mathcal{N} \tag{1}$$

as a result of Little's theorem and assuming that the tasks arrive in $M/M/1$ queues and experience a load dependent delay. Transmission delay $d^T_{i,j}$ is given by:

$$d^T_{i,j} = \frac{S_i}{c_{i,j}},\ \forall\ i \in \mathcal{N}_T, j \in \mathcal{N} \tag{2}$$

Since the output of the execution is much smaller in size than the input, it is omitted from (2) for simplicity. For instance in a face recognition application, the input size (photo) is in the order of a few MBytes while the result can be a few Bytes.

The task execution delay in node j is given by:

$$d^C_j = \frac{1}{\frac{h_j}{\rho_j} - \sum_{k \in \mathcal{N}_{(j)}} x_{k,j} \lambda_k} \forall\ j \in \mathcal{N} \tag{3}$$

We follow the same $M/M/1$ queuing approach in modeling the computation queuing delay at the nodes. The service and arrival rates are the task execution and arrival rates for all the neighbouring nodes that outsource tasks to some node j. The queuing delay at each node j considers the task's load and the CPU speed at the executing node to produce the average task execution rate of j. Notice how increasing variables $x_{k,j}$ increases the task arrival rate, hence affecting apart i, all of j's neighbours as well. As the task arrival rate reaches the execution rate, the queuing delay is infinite and the task queue becomes unstable. The average total (communication and computation) delay is then defined as:

$$D_i(\boldsymbol{x}) = \sum_{j \in \mathcal{N}_{(i)}} x_{i,j} \Big(d^T_{i,j} + d^Q_{i,j} + d^C_j \Big),\ \forall\ i \in \mathcal{N}\,. \tag{4}$$

Regarding the average computation accuracy of each node we use the following definition:

$$W_i(\boldsymbol{x_i}) = \sum_{j \in \mathcal{N}_{(i)}} x_{i,j} w_{i,j},\ \forall i \in \mathcal{N}\,. \tag{5}$$

The system's goal is to minimize the analytics' execution delay and maximize their execution accuracy. We notice that $D_i(\boldsymbol{x})$ is convex with $x_{i,j},\ j \in \mathcal{N}$ since the respective Hessian matrix is positive semi-definite, provided that the transmission and execution queues are stable. Instead of directly maximizing the accuracy, we use a convex utility approach for two main reasons. First we capture the diminishing returns effect that naturally arises in such systems. Namely, a certain improvement in accuracy is more important if the accuracy is low compared to the case that it is already high enough. The same holds for the delay since it is a convex function, in the sense that a certain delay deterioration is more negative/impactful if the delay is already high. Moreover, by using such utility for the accuracy we ensure a balanced dispersion of the collaboration benefits across the IoT nodes, satisfying this way a fairness criterion. In particular a fairness utility commonly used in network utility maximization is the a-fairness function [21]. Such function is the logarithmic, which we apply to the average execution accuracy, and by using its convex form $-\log W_i(\boldsymbol{x_i})$ we formulate the objective function as a sum of the two objectives:

$$U_i(\boldsymbol{x}) = \beta D_i(\boldsymbol{x}) - \gamma \log W_i(\boldsymbol{x_i}),\ \forall i \in \mathcal{N}_T \tag{6}$$

where β, γ are balancing parameters between the two objectives. For the balancing parameters we follow a scalarization approach where certain weights are assigned to the two objectives that we have. This ensures that the obtained solution is aligned with the priorities of the designer (delay or accuracy) and in any case they will be Pareto optimal [22, Sec. 4.7].

Regarding the total power consumption for any given node $j \in \mathcal{N}$ we define the following:

$$p^C_j(\boldsymbol{x}) = e^c_j \rho_j \sum_{i \in \mathcal{N}_{(j)}} x_{i,j} \lambda_i, \tag{7}$$

$$p^T_j(\boldsymbol{x}) = \sum_{i \in \mathcal{N}_{(j)}} e^t_{j,i} x_{j,i} R_j, \quad p^R_j(\boldsymbol{x}) = \sum_{i \in \mathcal{N}_{(j)}} e^r_{j,i} x_{i,j} R_i, \tag{8}$$

where p^C_j, p^T_j and p^R_j are the total power spent on computation, transmission and reception by node j respectively. Consequently, the joint execution delay and accuracy optimization problem is:

$$\boldsymbol{P_1}: \min_{\boldsymbol{x}} \sum_{i \in \mathcal{N}_T} U_i(\boldsymbol{x}) \tag{9a}$$

$$\text{s.t. } g_{1,j}(\boldsymbol{x}) = \sum_{i \in \mathcal{N}_{(j)}} \frac{x_{j,i} R_j}{c_{j,i}} - 1 \le 0,\ \forall\ j \in \mathcal{N}, \tag{9b}$$

$$g_{2,j}(\boldsymbol{x}) = \sum_{i \in \mathcal{N}_{(j)}} \frac{x_{i,j} R_i}{c_{i,j}} - 1 \le 0,\ \forall\ j \in \mathcal{N}, \tag{9c}$$

$$g_{3,j}(\boldsymbol{x}) = p^C_j(\boldsymbol{x}) + p^T_j(\boldsymbol{x}) + p^R_j(\boldsymbol{x}) - P_j \le 0,\ \forall j \in \mathcal{N} \tag{9d}$$

$$g_{4,i,j}(\boldsymbol{x}) = x_{i,j} - 1 \le 0,\ \forall\ i \in \mathcal{N}_T\ , j \in \mathcal{N}, \tag{9e}$$

$$g_{5,i,j}(\boldsymbol{x}) = -x_{i,j} \le 0,\ \forall\ i \in \mathcal{N}_T\ , j \in \mathcal{N}, \tag{9f}$$

$$h_i(\boldsymbol{x}) = \sum_{j \in \mathcal{N}_{(i)}} x_{i,j} - 1 = 0,\ \forall\ i \in \mathcal{N}_T. \tag{9g}$$

Eq. (9b)-(9c) guarantee the flow feasibility communication wise, i.e. they respect channel capacities [23]. The power budget constraint for each node is given by (9d). Note that it accounts for energy spent on both computation and communication. Finally, the sum of decision variables for each node i, over all its neighbours must be equal to 1 according to (9g) so that our routing policy is formed.

$\boldsymbol{P_1}$ is a convex yet challenging optimization problem. Note that $U_i(\boldsymbol{x})$ includes load-sensitive delay components and a logarithm of a summation. Moreover, the outsourcing and routing decisions of each node affect the resource consumption of its neighbours and through them, that of further-distant nodes. Thus, there is strong coupling among the decisions of nodes both in the objective and in the constraint set. Finally, often such IoT systems comprise hundreds of nodes, and their large size renders the above challenges more pronounced.

B. Problem Solution

In order to solve the problem, we rely on Lagrange duality that allows the decoupling of $\boldsymbol{P_1}$. Furthermore we define a new set of auxiliary variables for each node i, which serve as local copies of the decisions of the neighbors of j that affect its operation. This facilitates the decoupling of the objectives of the different nodes. Namely, we define variables $y_{i,j}$, and introduce the necessary constraints:

$$y_{i,j} = x_{j,i},\ \forall l_{j,i} \in \mathcal{L} \tag{10}$$

which ensure the consistent operation of the system. This transformation allows the scalable and distributed solution of the problem, which is particularly important for large IoT systems.

We first define the Lagrange function by relaxing all constraints:

$$\begin{aligned} L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\mu},\boldsymbol{v},\boldsymbol{\kappa}) &= \sum_{i\in\mathcal{N}_T} U_i(\boldsymbol{x},\boldsymbol{y}) + \sum_{k=1}^{3}\Big(\sum_{j\in\mathcal{N}} \mu_{k,j} g_{k,j}(\boldsymbol{x})\Big) \\ &+ \sum_{k=4}^{5}\Big(\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} \mu_{k,i,j} g_{k,i,j}(\boldsymbol{x})\Big) + \sum_{i\in\mathcal{N}} v_i h_i(\boldsymbol{x}) \\ &+ \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}} \kappa_{i,j}\Big(y_{i,j} - x_{j,i}\Big) \end{aligned} \tag{11}$$

where $\boldsymbol{\mu}, \boldsymbol{v}$ and $\boldsymbol{\kappa}$ are the Lagrange multiplier vectors for the inequality, equality and local variable copy constraints respectively. $\boldsymbol{y}$ is defined as: $\boldsymbol{y} = [y_{i,j},\ \forall i \in \mathcal{N},\ j \in \mathcal{N}_T]$ The dual problem is then defined as:

$$\max_{\boldsymbol{\mu}\geq\boldsymbol{0},\boldsymbol{v},\boldsymbol{\kappa}} \psi(\boldsymbol{\mu},\boldsymbol{v},\boldsymbol{\kappa}) = \inf_{x} L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\mu},\boldsymbol{v},\boldsymbol{\kappa}) \tag{12}$$

In order to solve (12), we follow a primal-dual decomposition method that gradually converges to the optimal solution. In each iteration, a gradient method is used to obtain improved values for the dual variables, and then use them to calculate the current gradient through the minimization of (11).

First, we observe that the dual problem has a separable structure and hence it favours a distributed solution. The latter is beneficial since it is scalable and requires only minimal circulation of coordination messages. Namely each node $i \in \mathcal{N}_T$ can optimize $\boldsymbol{x_i}, \boldsymbol{y_i}$, so the Lagrangian is written as a sum of partial Lagrangians L_i over all nodes $i \in \mathcal{N}_T$ such that:

$$L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\mu},\boldsymbol{v},\boldsymbol{\kappa}) = \sum_{i\in\mathcal{N}} L_i(\boldsymbol{x_i},\boldsymbol{y_i},\boldsymbol{\mu_i},\boldsymbol{v_i},\boldsymbol{\kappa_i}) \tag{13}$$

where $\boldsymbol{y_i}$ is defined as: $\boldsymbol{y_i} = [y_{i,1}, y_{i,2}, \dots y_{i,N}]\ \forall i \in \mathcal{N}$. $\boldsymbol{\mu}_i, \boldsymbol{v}_i$ and $\boldsymbol{\kappa_i}$ are sub-vectors of $\boldsymbol{\mu}, \boldsymbol{v}$ and $\boldsymbol{\kappa}$ that only contain the Lagrange multipliers associated with each node i, i.e. those appearing in each node's partial Lagrange function given by:

$$\begin{aligned} &L_i(\boldsymbol{x_i},\boldsymbol{y_i},\boldsymbol{\mu_i},\boldsymbol{v_i},\boldsymbol{\kappa_i}) = U_i(\boldsymbol{x_i},\boldsymbol{y_i}) + \\ &\sum_{k=1}^{3}\Big(\sum_{j\in\mathcal{N}} \mu_{k,j} g_{k,j}(\boldsymbol{x_i})\Big) + \sum_{j\in\mathcal{N}} \kappa_{i,j}\Big(y_{i,j} - x_{j,i}\Big) + \\ &\sum_{k=4}^{5}\Big(\sum_{j\in\mathcal{N}} \mu_{k,i,j} g_{k,i,j}(\boldsymbol{x_i})\Big) + v_i h_i(\boldsymbol{x_i}) \end{aligned} \tag{14}$$

The dual ascent method consists of a minimization step, where the Lagrangian (11) is minimized with respect to $\boldsymbol{x_i}, \boldsymbol{y_i}$ for some given $\boldsymbol{\mu_i}, \boldsymbol{v_i}, \boldsymbol{\kappa_i}$. After that, the Lagrange multipliers are updated using the solution obtained from the minimization step and sent to the neighbouring nodes before repeating the minimization step in an iterative way, until $\boldsymbol{x_i}$ converges to its optimal value $\boldsymbol{x_i}^*$. Formally we can write:

$$(\boldsymbol{x_i}^t, \boldsymbol{y_i}^t) = \arg\min_{\boldsymbol{x_i},\boldsymbol{y_i}} L_i(\boldsymbol{x_i},\boldsymbol{y_i},\boldsymbol{\mu_i}^{t-1},\boldsymbol{v_i}^{t-1},\boldsymbol{\kappa_i}^{t-1}) \tag{15a}$$

$$\boldsymbol{\mu_j}^t = [\boldsymbol{\mu_j}^{t-1} + \theta^t \boldsymbol{g_j^t}]^+, \tag{15b}$$

$$\boldsymbol{v_j}^t = \boldsymbol{v_j}^{t-1} + \theta^t \boldsymbol{h_j^t},\ \kappa_{i,j}^t = \kappa_{i,j}^{t-1} + \theta^t (y_{i,j} - x_{j,i}) \tag{15c}$$

where the superscript t indicates a variable's value at the t-th iteration and $\theta^t > 0$ is the step size. Vectors $\boldsymbol{g_j^t}$ and $\boldsymbol{h_j^t}$ contain the values of (9b)-(9f) and (9g) respectively of the constraints related to j, while $[x]^+$ denotes that $x = \max\{0, x\}$. Each node $i \in \mathcal{N}$ can then evaluate $\boldsymbol{x_i}^*, \boldsymbol{y_i}^*$ after receiving the dual variable updates in an iterative process, which is briefly described in Algorithm 1.

Algorithm 1 Distributed Collaborative Task Execution

1: **Input:** $\mathcal{N}, \mathcal{N}_T, \beta, \gamma, \theta, \boldsymbol{\mu_i^0}, \boldsymbol{v_i^0}, \boldsymbol{\kappa_i^0}$
2: **repeat**
3: **for each** node $i \in \mathcal{N}_T$ **do**
4: $t \leftarrow t + 1$
5: $(\boldsymbol{x_i^t}, \boldsymbol{y_i^t}) \leftarrow \arg\min_{\boldsymbol{x_i},\boldsymbol{y_i}} L_i(\boldsymbol{x_i},\boldsymbol{y_i},\boldsymbol{\mu_i^{t-1}},\boldsymbol{v_i^{t-1}},\boldsymbol{\kappa_i^{t-1}})$
6: Transmit $\boldsymbol{x_i^t}$ to neighbours
7: **end for**
8: **for each** node $j \in \mathcal{N}$ **do**
9: Update $\boldsymbol{\mu_j^t}, \boldsymbol{v_j^t}, \kappa_{j,i}^t$ by using (15b), (15c)
10: Transmit $\boldsymbol{\mu_j^t}, \boldsymbol{v_j^t}, \kappa_{j,i}^t$ to neighbours
11: **end for**
12: **until** $||\boldsymbol{x_i^t} - \boldsymbol{x_i^{t-1}}|| \leq \epsilon$

The algorithm's input is the number of nodes, the balancing parameters, the step size and the initial values for the Lagrange multipliers, all considered known to each node. The step size is chosen as a sufficiently small positive value to allow the algorithm to converge. Steps 2-12 describe the distributed iterative process of converging to the optimal $\boldsymbol{x}_i^*$. Each node $i \in \mathcal{N}_T$ evaluates $\boldsymbol{x}_i, \boldsymbol{y}_i$ (step 5), and transmits $\boldsymbol{x}_i$ to its neighbors, so they update their associated Lagrange multipliers (step 9) and send them to their neighbours (step 10) so that a new iteration can commence. Termination of the algorithm is achieved at some iteration t, for which it holds that $||\boldsymbol{x_i^t} - \boldsymbol{x_i^{t-1}}|| \leq \epsilon, \forall i \in \mathcal{N}_T$, where ϵ is a small positive, close to 0 value (step 12).

Proposition 3.1: Algorithm 1 solves problem $\boldsymbol{P_1}$ and converges to the optimal solution in a finite number of steps.

Proof: The Dual Ascent method solves the Lagrangian dual problem where the dual function is given in (12), by iteratively increasing the dual variables $\boldsymbol{\mu}, \boldsymbol{v}, \boldsymbol{\kappa}$ towards maximizing $\psi(\boldsymbol{\mu}, \boldsymbol{v}, \boldsymbol{\kappa})$. The maximum value of $\psi(\boldsymbol{\mu^*}, \boldsymbol{v^*}, \boldsymbol{\kappa^*}) = d^*$ provides a lower bound on the solution p^* of $\boldsymbol{P_1}$. Since the objective $\sum_{i \in \mathcal{N}_T} U_i(\boldsymbol{x_i}, \boldsymbol{y_i})$ is strictly convex, if Slater's conditions apply strong duality holds so that $d^* = p^*$ and solving the dual problem, using steps (15a)-(15c), solves the primal [22, Sec 5.3]. Algorithm 1 is the distributed implementation (dual decomposition) of the dual ascent method, which has been shown to converge linearly to $\boldsymbol{x^*}$ [24]. The communication overhead from steps 6, 10 is upper bounded by the maximum number of neighbours of each node, which is at most N. In that case $\boldsymbol{x_i}$ contains N variables, and the dual vectors $\boldsymbol{\mu}, \boldsymbol{v}, \boldsymbol{\kappa}$ contain $3+2N, 1$ and N variables, respectively. All together we get an exchange of $M = 3(N+1)+1$ variables, so the overhead is of $O(NM)$ order. ■

IV. Performance Evaluation

In this section we evaluate the performance of the proposed system and the associated algorithm devised to obtain optimal task outsourcing. We first describe the simulation setup, and then conduct a sensitivity analysis for selected key parameters of our model. Finally, we demonstrate the distributed algorithm's convergence and scalability properties for different network sizes.

Evaluation Setup. In order to evaluate our algorithm for relatively large network sizes,in our evaluation we use a custom simulator whose parameters are based on measurements on a small testbed of Raspberry Pi 3 Model B devices.

We measure compute and Wi-Fi power consumption using a digital multi-meter connected to the Pis. We measured compute power consumption by stressing the CPU of each Pi[3] using a methodology similar to [25]. Wi-Fi power consumption was measured by sending iperf UDP traffic at different data rates between two Pis connected with a 802.11g link in ad hoc mode. The power measurements are used to determine $e^t_{i,j}, e^r_{i,j}$ and e^c_j parameters in our simulator (See Table I).

The Pis are running a face recognition application, implemented in Java OpenCV [12]. The application has a source that generates frames from a video feed, detects faces in each frame and compares found faces to a local database. We ran the application locally at each Raspberry Pi and measured throughput and accuracy for three different image resolutions/sizes S_i (5, 8 and 32 KB) and classification algorithms (LBP, Haar), when the source sends frames in backlogged mode. Each classification algorithm corresponds to a different type of executing node in our simulator that affects the required cycles per task ρ_j.

We used Matlab R2015a in order to simulate the behavior of the system, and obtain a solution for $\boldsymbol{P_1}$. The IoT network is created using a random geometric graph model that is appropriate for such ad hoc networks. Namely, we have randomly placed the nodes in a $100m \times 100m$ area, assuming that they are in communication range when their distance is less than 50m, while their link capacities are calculated based on path loss and Rayleigh Fading. For each experiment we present results by averaging 100 simulations that differ in node locations.

[3]Each Pi has a 1.2 GHZ quad-core ARM Cortex A53 processor.

Description	Parameter	Value
Number of nodes, Number of nodes requesting tasks	N, N_T	25, 10
Transmission and reception energy consumption per bit	$e^t_{i,j}, e^r_{i,j}$	15-78, 5.5-38 nJ/b
Number of cycles per task	ρ_j	52-889 Mcycles
Energy consumption per computation cycle	e^c_j	0.2 nJ/cycle
Data per task	S_i	5-32 KB
Energy budget	P_j	0.4 W
Balancing parameters	β, γ	0.5
Task arrival rate	λ_i	1

TABLE I: Simulation parameters.

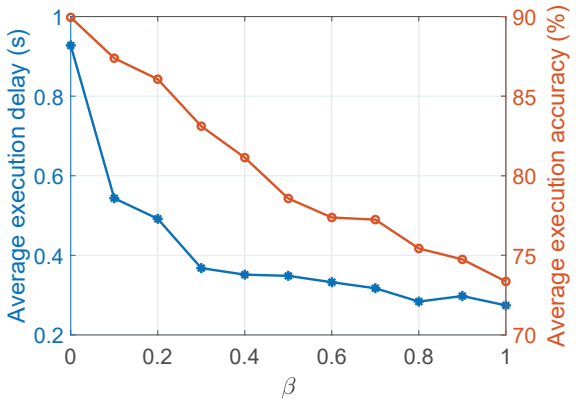

Fig. 2: The effect of balancing parameter β and $\gamma = 1 - \beta$ on the task execution time and task accuracy. Results are averaged for all nodes in each network, and for 100 different random network instances.

Balancing parameters β, γ and objective correlation. First we evaluate the impact of β and γ on the execution delay and accuracy. As it is expected, when the ratio β/γ is increased, the solution of $\boldsymbol{P_1}$ prioritizes delay (which is reduced) over accuracy (which deteriorates). Fig. 2 quantifies this effect, where we have averaged the delay and accuracy for all nodes in 100 different random networks with the same parameters except node location. We observe that when β increases up to 0.4, the delay is reduced to less than 0.4 secs (about 60% average improvement to $\beta = 0$), while for higher β values this improvement is smaller (still monotonic). On the other hand, the accuracy deteriorates with almost constant rate, reduced down to 73% for the extreme case the system gives full priority to delay ($\gamma = 0, \beta = 1$). Clearly, the exact values of average delay and task accuracy as β/γ changes depend also on the other system parameters shown in Table I. Moreover, this trade off is largely shaped by the network structure, in particular the properties of neighbouring nodes, and it is important to understand this dependency.

To this end, we go a further step and use the Kendal rank correlation coefficient [26] in order to predict, for a given network, how steep this trade off curve will be. In detail, for each node $i \in N_T$ we create two ordered lists of its neighbours, ranking them according to delay (lower to higher) and accuracy (higher to lower). In other words, the top-ranked node in the delay list is the one that can execute node i's tasks with the lower delay, while the top-ranked node in the accuracy list offers the highest accuracy. Then, for each node, we test for

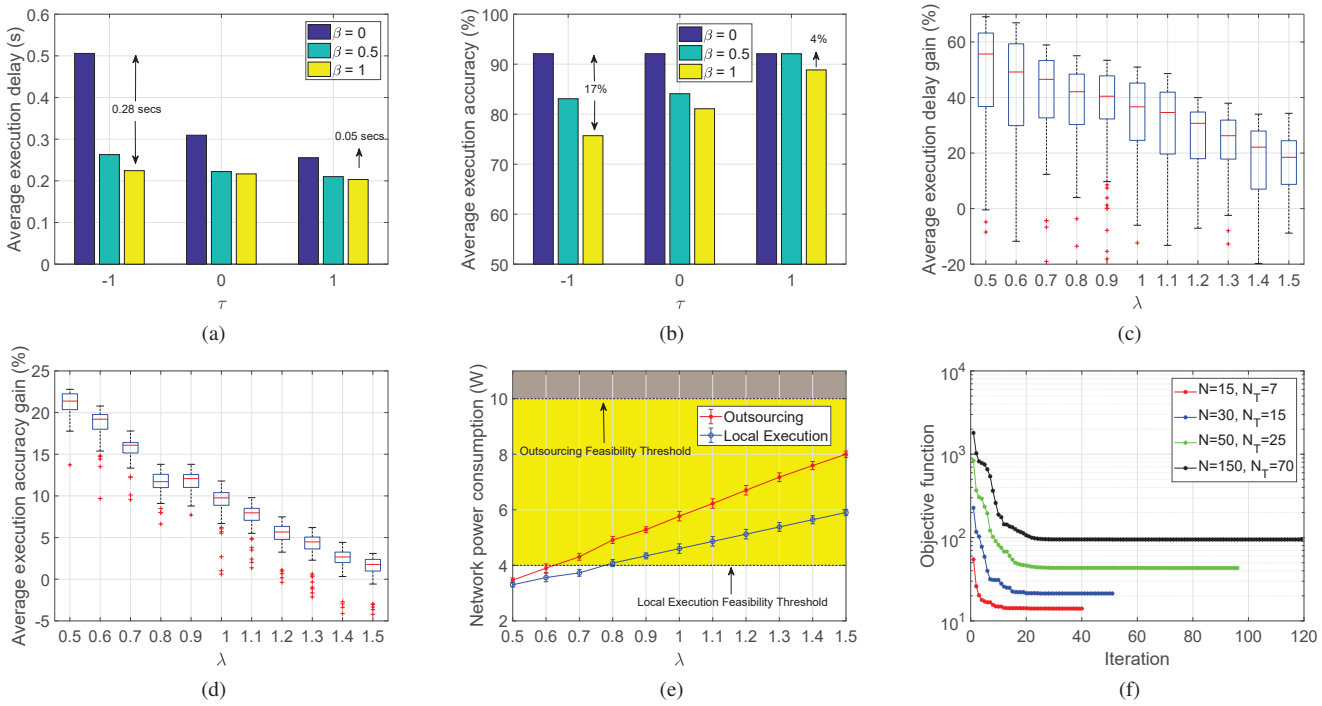

Fig. 3: Average network execution delay and accuracy for different levels of objective correlation in (a) and (b). The average gain in execution delay and accuracy as a function of the task arrival load λ in (c) and (d). Power consumption of outsourcing and local execution approaches displayed in (e) and the objective function at each algorithm iteration in (f).

all possible pairings between its neighbours, if they maintain their order in the two lists or not and count the number of concordant and discordant pairs. The Kendal rank correlation coefficient for a node i with N_i neighbours is given by:

$$\tau_i = \frac{P_i^c - P_i^d}{N_i(N_i-1)/2}, \ \forall i \in \mathcal{N}_T \tag{16}$$

where P_i^c is the number of concordant pairs and P_i^d the number of discordant pairs. Since $N_i(N_i-1)/2$ is the number of possible pairs it holds that: $-1 \leq \tau_i \leq 1$ and averaging for all $i \in \mathcal{N}_T$ we can obtain a unique objective similarity index τ for our network.

In Fig. 3(a-b) we depict the average execution delay and accuracy (mean values for 100 instances) of the solution of $\boldsymbol{P_1}$, when parameters $w_{i,j}$ are chosen so that $\tau \in \{-1, 0, 1\}$. We observe that when $\tau = -1$, i.e., the objectives are conflicting, the choice of β plays a significant role in the solution. For $\beta = 0$ we get high delay and accuracy, while for $\beta = 1$ the opposite. As the correlation of the objectives becomes stronger, the impact of β becomes smaller. In the extreme case of $\tau = 1$, the objectives are fully correlated and optimizing with respect to delay is roughly the same as optimizing with respect to accuracy.

Impact of load on cooperation benefits. Next, we explore the impact of the average task load λ on the *cooperation delay gain* of the IoT network. The latter is defined as the average task delay improvement (%) when the nodes collaborate based on the policy of $\boldsymbol{P_1}$, over when they execute the tasks locally. In Fig. 3c we plot this gain for different values of λ, where we have set $\lambda_i = \lambda, \forall i \in \mathcal{N}_T$. Note that in order to fairly compare collaborative with local execution, we have considered a large power budget ($P_j = 1W, \forall j \in \mathcal{N}$), so that neither mode (local or collaborative) violates the power constraints in $\boldsymbol{P_1}$.

There are some interesting observations here. First, note that the gain is positive, since some nodes expedite their tasks by outsourcing them to neighbours with high processing power with which they are connected with high-capacity links. For larger values of lambda however, the queuing delays increase fast (due to congestion) and hence our policy sacrifices performance (lower delay gains) in order to satisfy the power consumption constraints. Interestingly, in some cases, the cooperation might increase the delay for some nodes (negative delay gain) in order to achieve higher accuracy.

Similarly, in Fig. 3d we depict the *cooperation accuracy gain*, defined as the average task accuracy when the nodes collaborate, minus the average accuracy when all tasks are executed locally. Hence, the gain is positive if collaborative performance is better. We see again that there are important cooperation gains (few outliers are present), which are reduced for high load values (even by 20%).

These results show that even if the network is not energy-constrained, our algorithm has merit as it improves both the average delay and accuracy. However, as the load increases, the outsourcing of all tasks becomes very costly in terms of energy or bandwidth overheads, hence the average gains are reduced.

Outsourcing feasibility gains. Clearly, more often than not, the nodes will have tighter energy budgets than assumed above. In these cases, the cooperation among the nodes not only increases the system's performance, but also ensures the completion of the tasks even if they arrive at a (high)

rate that cannot be supported by local-only execution. This is demonstrated in Fig. 3e, where we have averaged results over 100 network instances. We observe that for $\lambda < 0.8$ both the outsourcing and the local execution solutions are feasible (white area), but for higher loads local execution exceeds the nodes' aggregate power allowance, i.e., $\sum P_i$. When task outsourcing is allowed however, all nodes in $\mathcal{N}$ contribute to the power budget which increases and allows the support of more tasks. Finally, it is interesting to note that the cooperative solution consumes more energy than the local-only execution (focus on the white area), as it tries to minimize the objective of $\boldsymbol{P_1}$ by achieving lower delay and higher accuracy than local solution.

Convergence of Algorithm 1. Finally, in Fig. 3f we evaluate the convergence of the distributed Algorithm 1, using 4 different network sizes and a fixed step size. We observe that the optimal solution is reached after few iterations, even for large networks of $N = 150$ nodes, which verifies the scalability properties of the algorithm. Furthermore, we see that in practice, the algorithm achieves approximately 95% of the optimal solution after only 25 iterations for all experiments.

V. Conclusion

In this work we studied how the collaborative execution of data analytic tasks in an IoT network can increase its performance. We formulated an optimization problem that minimizes the task execution delay and maximizes their accuracy, while respecting the devices power constraints. The proposed distributed algorithm converges fast, even for large networks and increases the performance compared to the local-only task execution. Furthermore, the algorithm provides feasible solutions for high load cases that local execution cannot support (due to lack of computation or energy resources). A trace-driven evaluation showed that indeed there is need for such cooperative solutions, and that the designer needs to carefully tune the priority parameters, especially for networks where delay and accuracy are conflicting.

Acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the ofcial policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon. The authors also acknowledge the support from the Science Foundation Ireland (SFI) under Grant Number 17/CDA/4760.

References

[1] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5G era: Enablers, architecture, and business models," *IEEE JSAC*, vol. 34, no. 3, pp. 510–527, 2016.

[2] F. Bonomi, R. Milito, Z. Zhang, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. of MCC*, 2012.

[3] Cisco, "Fog computing and the internet of things: Extend the cloud to where the things are," white paper, 2015.

[4] S. K. Sharma and X. Wang, "Live data analytics with collaborative edge and cloud processing in wireless iot networks," *IEEE Access*, vol. 5, pp. 4621–4635, 2017.

[5] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. of ACM EuroSys*, 2011.

[6] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, 2010.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, pp. 14–23, Oct. 2009.

[8] C. K. Tham and R. Chattopadhyay, "A load balancing scheme for sensing and analytics on a mobile edge computing network," in *Proc. of IEEE WoWMoM*, 2017.

[9] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikainen, and V. H. Tuulos, "Misco: A mapreduce framework for mobile systems," in *Proc. of ACM PETRA*, 2010.

[10] M. Y. Arslan, I. Singh, S. Singh, H. V. Madhyastha, K. Sundaresan, and S. V. Krishnamurthy, "Computing while charging: Building a distributed computing infrastructure using smartphones," in *Proc. of ACM CoNEXT*, 2012.

[11] H. Wang and L.-S. Peh, "Mobistreams: A reliable distributed stream processing system for mobile devices," in *Proc. of IEEE IPDPS*, 2014.

[12] S. Fan, T. Salonidis, and B. Lee, "A framework for collaborative sensing and processing of mobile data streams," in *Proc. of ACM MobiCom*, 2016.

[13] D. O'Keeffe, T. Salonidis, and P. Pietzuch, "Network-aware stream query processing in mobile ad-hoc networks," in *IEEE Military Communications Conference*, pp. 1335–1340, Oct 2015.

[14] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. of the Thirteenth ACM MobiHoc*, pp. 145–154, 2012.

[15] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive energy-aware computation offloading for cloud of things systems," *IEEE Access*, vol. 5, pp. 23947–23957, 2017.

[16] Z. Sheng, C. Mahapatra, V. C. M. Leung, M. Chen, and P. K. Sahu, "Energy efficient cooperative computing in mobile wireless sensor networks," *IEEE Transactions on Cloud Computing*, vol. 6, pp. 114–126, Jan 2018.

[17] S. Yang, Y. Tahir, P. y. Chen, A. Marshall, and J. McCann, "Distributed optimization in energy harvesting sensor networks with dynamic in-network data processing," in *Proc. of IEEE INFOCOM*, 2016.

[18] T. Salonidis, G. Sotiropoulos, R. Guerin, and R. Govindan, "Online optimization of 802.11 mesh networks," in *Proc. of ACM CoNEXT*, 2009.

[19] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11n/ac in smartphones," in *Proc. of IEEE INFOCOM*, 2015.

[20] M. A. Bagheri and Q. Gao, "An efficient ensemble classification method based on novel classifier selection technique," in *Proc of ACM WIMS*, 2012.

[21] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *Proc. of IEEE INFOCOM*, 2010.

[22] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[23] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels," *IEEE/ACM Transactions on Networking*, vol. 13, Aug 2005.

[24] Z.-q. Luo and P. Tseng, "On the convergence rate of dual ascent methods for linearly constrained convex minimization," *Math. Oper. Res.*, vol. 18, pp. 846–867, Nov. 1993.

[25] F. Kaup, P. Gottschling, and D. Hausheer, "Powerpi: Measuring and modeling the power consumption of the raspberry pi," in *39th Annual IEEE Conference on Local Computer Networks*, pp. 236–243, Sept 2014.

[26] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.