TAAC: Task Allocation Meets Approximate Computing for Internet of Things

Wanli Yu∗, Ardalan Najafi†, Yarib Nevarez*, Yanqiu Huang† and Alberto Garcia-Ortiz∗
∗University of Bremen, Germany, {wuy, ardalan, nevarez, agarcia}@item.uni-bremen.de
†University of Twente, Netherlands, y.huang-3@utwente.nl

Abstract—Ultra-low-power operation, as required by Internet of Things (IoT) systems, requires to address energy consumption at all levels from circuit to system. Two of the promising solutions at circuit and system levels are approximate computing and energy aware task allocation, respectively. However, the existing task allocation approaches are designed without considering the aspect of approximate computing. For the first time, this work proposes an optimal Task Allocation algorithm taking the Approximate Computing into account (TAAC) to fill this gap. The problem of tasks assignment and executing modes selection (approximate or exact modes of the tasks) can be efficiently solved by formulating it as a linear programming problem. The extensive simulation results show that the proposed TAAC algorithm significantly outperforms the previous approaches.

I. INTRODUCTION

The emerging IoT technology can provide promising solutions for numerous present-day society issues, such as intelligent manufacturing, smart cities, remote healthcare and automatic agriculture [1]. As it is impractical to transport all data of the huge amount of IoT devices over today’s already congested backbone Internet, the concept of fog node (FN) has been proposed by Cisco [2] to provide nearby storage and computing services for IoT end devices (EDs). Due to the fact that the EDs and FN are typically powered by battery energy, reducing energy cost is always one of the primary concerns [3].

Recent years have seen plenty of research on palliating the issue of energy consumption reduction from the aspects of both the circuit and system levels. Approximate computing is one of the most promising solutions allowing inexact functionality by redesigning the logic circuits [4]. By executing the tasks in approximate modes instead of the exact modes, the energy consumption can be efficiently reduced with inexact functionality [5]. By matching the statistical error attributes of the hardware with the statistical processing requirements of the target application, a tremendous gain can be achieved. An example is the hybrid adders which have been proposed by the authors in [6] by combining the idea of small errors and infrequent errors. They show that with a wise selection of approximate adders up to 65% energy-saving can be reached. From the system level, energy aware task allocation has been widely used to achieve the energy efficiency by appropriately distributing the workload for different devices in the network. In [7], an optimal task allocation algorithm is designed for a cluster based network by formulating the optimization problem as a linear programming problem. DOTAM is proposed by [8] based on Dantzig-Wolfe decomposition to address the task allocation problem for multihop mesh network. However, the existing task allocation algorithms are only designed for tasks on the exact modes. They are not sufficient for the tasks with mixed selection options, i.e., choosing the tasks in the exact or the approximate modes.

This work fills the above-mentioned gap between task allocation and approximate computing in IoT, by proposing an optimal Task Allocation algorithm taking the Approximate Computing into account (TAAC). To the best of our knowledge, this is the first work that considers both the task allocation and approximate computing to achieve energy efficiency for IoT applications. The rest of this paper is organized as follows. Section II presents the system models. The next section illustrates the proposed TAAC algorithm. The performance evaluation is presented in Section IV. The last section summarizes this work.

II. SYSTEM MODELS

This section presents the system models: the model of the approximate computing based applications, the network structure, and the cost functions of IoT devices.

A. Modeling Applications with Approximated Tasks

An IoT application is typically made up of a set of dependent computing tasks and can be modeled by a Directed Acyclic Graph (DAG) [8]–[11]. In the formatted DAG, $G = (V, E)$, each vertex $v \in V$ stands for a task in the application. The tasks are connected with others by directed edges. Each edge $e \in E$ represents the communication from its source task to its direct destination task. Only when receiving the data from all predecessor tasks, a task is executed and the generated data is transmitted to its successor tasks. By integrating approximate computing, each task in the DAG can be executed either in the approximate mode to further reduce the energy consumption while producing results with errors, or in the original exact mode. The schematic diagram is illustrated in Fig. 1.

![DAG graph for application with approximated tasks.](image-url)

Fig. 1. DAG graph for application with approximated tasks.

Dealing with approximate computing, one of the most important concerns is the output quality. According to the way of the error generation, the current approximate computing
approaches can be classified into two clusters: producing 
infrequent large errors and small errors all the time. For 
IoT applications, it is crucial to guarantee the final system 
output quality when executing approximated tasks. This work 
provides linear formulations of the relation between the quality 
of the system output and each single approximated tasks.

In the cluster of producing infrequent large errors, once 
a executed approximated task makes error, the system output 
quality is not acceptable because of the large magnitude of the 
errors. Let $p_k$ denote the probability of producing error when 
executing task $v_k$ in the approximate mode. The probability 
of the successful system output for a DAG with $K$ tasks, $p_s$, 
can be calculated by:

$$p_s = (1-p_1)(1-p_2)...(1-p_K) = 1 - \sum_{k=1}^{K} p_k.$$  \hspace{1cm} (1)

Note that, due to the infrequent nature of errors, $p_k$ is typically 
very small and consequently the products of $p_k$s are negligible.

For the second cluster, this work takes the popular linear 
DSP system, in which the tasks produce uncorrelated errors, 
for example. According to [12], the error variance of the 
system output, $\sigma^2_{err}$, can be modeled by the error characteristic 
of each individual task as follows:

$$\sigma^2_{err} = \sum_{k=1}^{K} c_k \sigma^2_{k},$$  \hspace{1cm} (2)

where $c_k$ is the error coefficient of the EDs, which is in fact 
the impulse response of the device; $\sigma^2_{k}$ is the error variance 
by executing task $v_k$ in the approximate mode which can be 
calculated, as an example, for an optimal approximate adder producing small errors as presented in [13].

B. Network Structure and Cost Functions

The fog computing based IoT network consists of end 
deVICES (EDs), fog nodes (FNs) and remote cloud center. Each 
ED has to periodically finish its own application with the collaboration of the corresponding FN 1. EDs are responsible for sensing, pre-processing and transmitting data to the corresponding FNs; each FN is in charge of receiving the data from the EDs under its management and finishing the rest tasks of the applications. We assume a perfect time synchronization and a negligible interference between each fog group, which can be achieved by the time division multiple access (TDMA) based protocols [14].

The cost functions are illustrated based on one of the most popular IoT devices, wireless sensor node. The execution time of ED $i$ and FN for executing task $v$ are:

$$t_i(v) = w(v)/f_i \quad \text{and} \quad t_F(v) = w(v)/f_F$$  \hspace{1cm} (3)

where $w(v)$ stands for the computation workload (the number of CPU clock cycles) of task $v$, $f_i$ and $f_F$ are the working frequencies of ED $i$ and the FN, respectively. The corresponding processing energy cost can be formulated by:

$$e_i(v) = P_i t_i(v) \quad \text{and} \quad e_F(v) = P_F t_F(v)$$  \hspace{1cm} (4)

1Due to the high probability of intolerable delay and network congestion related with the remote cloud center, this work considers the applications are completed by each ED and its FN.

where $P_i$ and $P_F$ represent the average processing power of ED $i$ and the FN, respectively. The corresponding processing cost for executing task $v$ in approximate mode is expressed by $c_i(v)r_a$ and $e_F(v)r_a$, respectively, where $r_a$ is the energy ratio between executing the task in approximate and exact modes.

As reported in [9], the communication procedure of a wireless device includes not only the data packets communication but also other overhead activities. Thus, the energy cost spent on transmitting and receiving $L$ bits of data, $E_{tx}$ and $E_{rx}$, can be formed by:

$$E_{tx} = e_o + e_{tx}(d_i)L \quad \text{and} \quad E_{rx} = e_o + e_{rx}L$$  \hspace{1cm} (5)

where $e_o$ is the energy consumption by the overhead activities, $d_i$ is the distance between ED $i$ and the FN, $e_{tx}(d_i)$ and $e_{rx}$ are the energy dissipated by transmitting and receiving one bit of data package, respectively.

III. TAAC ALGORITHM

In this section, the problem modeling is firstly presented and then the TAAC algorithm is proposed by formulating the problem as a linear programming (LP) problem.

A. Problem Modeling

Each ED has to periodically complete all of the tasks of its own application (DAG) with the collaboration of the corresponding FN. As each task in the DAG can be executed in either the exact or the approximate modes, the task allocation problem is to partition the tasks for the ED and FN, and select the modes of the tasks at the same time in each round as shown in Fig. 2.

B. TAAC

This section firstly linearly formulates the computation and communication cost of EDs and FN and the error constraints. It then formulates the above problem as an LP problem.

For task $v_k$ in a given DAG $i$, it can be assigned to either the ED $i$ or FN. Meanwhile, it can be executed in either the approximate or exact modes. This work uses a vector variable, $\gamma_{ik} = [\gamma_{ie}(v_k), \gamma_{ia}(v_k), \gamma_{Fe}(v_k), \gamma_{Fa}(v_k)]$, to represent the result of task allocation and execution modes selection of task $v_k$. $\gamma_{ie}(v_k), \gamma_{ia}(v_k)$ and $\gamma_{Fe}(v_k), \gamma_{Fa}(v_k)$ are the probabilities of assigning the exact and approximate modes of task $v_k$ to ED $i$ and FN, respectively. They are subject to Constraint a): each element of $\gamma_{ik}$ is in the range of $[0, 1]$, i.e., $0 \leq \gamma_{ik} \leq 1$; and the summation of the four elements equals to 1, i.e., $\sum_{i}^{4} \gamma_{ik} = 1$. The task allocation and mode section of all the $K_i$ tasks in DAG $i$ can be expressed by $\Gamma_i = [\gamma_{i1}, \ldots, \gamma_{iK}]$.
According to Eq. (4), the energy cost of ED $i$ and FN for executing task $v_k$ can be expressed as $e_{ik}\gamma_{ik}$ and $E_{FK} = e_{FK}\gamma_{ik}$, respectively, where $e_{ik} = [e_{ic}(v_k), e_{id}(v_k), 0, 0]$ and $e_{FK} = [0, 0, e_{F}(v_k), e_{F}(v_k)]$. Therefore, the computation cost of ED $i$ and FN for DAG $i$ can be formulated by:

$$E_{p,i} = E_i\Gamma_i \quad \text{and} \quad E_{F,i} = E_{p,F,i}\Gamma_i$$  \hspace{1cm} (6)

where $E_i = [e_{i1}, \ldots, e_{iK}]$ and $E_{F,i} = [e_{F1}, \ldots, e_{FK}]$.

As shown in Fig. 2, the amount of transmitted data from ED to FN is the summation of the weights of edges which cross the partition cut. Let $l_{in}(v_k) = \sum_{c_{out}(v_k)} l(e) - \sum_{c_{in}(v_k)} l(e)$ denote the net generated data of task $v_k$ of the DAG, where $in(v_k)$ and $out(v_k)$ are the input and output edges of task $v_k$. The amount of data transmitted from ED to FN can be expressed by $L = LL\Gamma_i$, where $L = [l_n(v_1), l_n(v_1), 0, 0, \ldots, l_n(v_K), l_n(v_K), 0, 0]$. According to Eq. (5), the transmitting cost of ED $i$ and the receiving cost of FN for DAG $i$, $E_{tx,i}$ and $E_{rx,i}$, can be formulated by:

$$E_{tx,i} = e_a + e_{tx}(d_i) LL\Gamma_i \quad \text{and} \quad E_{rx,i} = e_o + e_{rx} LL\Gamma_i$$  \hspace{1cm} (7)

Based on Eqs. (6) and (7), the energy cost of ED $i$ is:

$$E_i(\Gamma_i) = E_i\Gamma_i + e_a + e_{tx}(d_i) LL\Gamma_i.$$  \hspace{1cm} (8)

As FN is in charge of all $n$ EDs, its energy cost can be expressed as:

$$E_F(\Gamma_1, \ldots, \Gamma_n) = \sum_{i=1}^{n} E_{p,F,i}\Gamma_i + e_o + e_{rx} LL\Gamma_i.$$  \hspace{1cm} (9)

Note that $\Gamma_i$ is the only variable in Eqs. (8) and (9).

Although there are different methods to calculate the errors generated by the approximated tasks, the errors can be formulated by linear functions as presented in Section II-A. Here we choose Eq. (2) to calculate the generated errors. The system error variance corresponding to DAG $i$ should not be beyond the error requirement, i.e., \textbf{Constraint b)}, can be formulated as:

$$\sigma_i^2 \Gamma_i \leq \tau_i$$  \hspace{1cm} (10)

where $\sigma_i^2 = [0, \sigma_1^2, 0, \sigma_2^2, \ldots, 0, c_K, \sigma_K^2, 0, c_K, \sigma_K^2]$; $\tau_i$ is the error threshold pre-defined by the users.

As network lifetime extension is the most frequently used metric for measuring the energy efficiency of the task allocation approaches in IoT, such as in [15]–[17], the objective of the task allocation and task mode selection problem in this work is to maximize the network lifetime. According to [18], the network lifetime is defined as the time when the first device dies. Based on Eqs. (8) to (10), the problem of task allocation and task modes selection for a Fog-IoT network with one FN and $n$ EDs can be formulated by the following LP format:

$$\arg\min_{\Gamma_i} \max_{i = 1, \ldots, n} \left\{ \frac{E_F(\Gamma_1, \ldots, \Gamma_n)}{Bat_F}, \frac{E_i(\Gamma_i)}{Bat_i} \right\} \quad \text{subject to:  \textbf{Constraints a) and b))}$$  \hspace{1cm} (11)

where $Bat_i$ and $Bat_F$ are the battery energy of ED $i$ and FN, respectively. By solving the above LP problem, the network lifetime can be maximized by the obtained results of task allocation and task modes selection, $\Gamma_1, \ldots, \Gamma_n$.

IV. EVALUATION

This section evaluates the proposed TAAC algorithm by extensive simulation results. The superiority of TAAC is demonstrated by comparing with the exhaustive search and reference [7]. For fair comparison, this work follows the simulation setup in [7]: we randomly generate the Fog-IoT network in a two-dimensional area of $100 \times 100$ m$^2$ with one FN at the center and $n$ randomly distributed EDs; the same values of the energy parameters are taken for EDs and FNs. The DAGs are randomly generated, in which the computing workload of each task in exact mode is within the range of $[100, 500]$ KCCs (kilo clock cycles) and the communication data on each edge in the range of $[100, 1000]$ bits. We consider the DAGs for all EDs have the same number of tasks. The error generated by each task in approximate mode and the corresponding coefficient, $\sigma_i^2$ and $c_K$, are randomly distributed within $[0, 0.01]$ and $[0.5, 1.5]$, respectively. The tasks executed in exact modes do not generate any error and the error limitation for the application is 0.02.

The performance on network lifetime extension and system error variance as well as the algorithm running time in Matlab 2017a are investigated by changing three configuration parameters: a) the number of EDs, $n$; b) the number of tasks, $K$; c) the energy ratio between approximate and exact modes, $r_o$. Note that only one parameter is changed in each simulation. The reported results correspond to the average values and the standard deviations of 500 test instances for each simulation.

The first set of simulations investigate the algorithm performance by changing the number of the EDs, $n$. Fig. 3a depicts the normalized network lifetime of TAAC and the exhaustive search with respect to [7]. It is clear that TAAC performs as well as the exhaustive search, and both of them significantly extend the network lifetime with respect to [7]. For example, they increase the network lifetime to 2.82 times of [7]. When $n$ changes from 5 to 40, the gain of TAAC slightly changes. This is due to the fact that the energy cost ratio between executing the tasks in the approximate and exact modes does not change. When looking at the generated errors by the selected approximated tasks in Fig. 3b, the errors generated by executing all the tasks in the approximate modes do not change with the increased $n$, since the number of tasks does not change. According to the error constraint, TAAC always satisfies the predefined error limitation. Moreover, TAAC requires very short algorithm runtime as shown in Fig. 3c, e.g., it only needs 0.0098 second when $n = 5$. As $n$ becomes 40, the algorithm runtime of TAAC slightly increases. The reason is that the increased number of EDs results in more variables in TAAC. Compared with the exhaustive search, the algorithm runtime of TAAC can be neglected.

In the second set of simulations, we estimate the impact of changing the number of tasks in the DAG, $K$, on the algorithm performance. As shown in Fig. 4a, both TAAC and the exhaustive search still perform the same and dramatically outperform [7] as expected. Different from the results in

\footnote{Reference [7] provides one optimal task allocation algorithm by considering all of the tasks are executed on the exact mode. The exhaustive search is to run [7] for all possible solutions and select the best one.}
For the first time, this work fills the gap between task allocation and approximate computing in IoT scenarios by proposing an optimal Task Allocation algorithm considering Approximate Computing, TAAC. TAAC formulates the problem of task allocation and execution modes selection as a linear programming problem, which is efficiently solvable. It improves the energy efficiency of IoT applications based on approximated tasks, while it can also guarantee the error limitations. We believe that the future IoT application can benefit a lot from this cross-level energy optimization from the circuit to system.

V. CONCLUSION

In addition to the number of EDs and tasks, a third set of simulations are conducted to investigate the impact of the energy ratio between the approximated and the exact tasks, $r_a$, on the performance of TAAC. According to Fig. 5, it is clear that the gains of both TAAC and the exhaustive search w.r.t. [7] significantly increase when $r_a$ changes from 0.9 to 0.1. For a smaller $r_a$, executing the tasks in approximate modes saves more energy. As the number of tasks does not change, the performance on generated errors and the algorithm runtime are very close to the results in Fig. 3b and Fig. 3c, respectively.
REFERENCES


