This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2873239, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

QoS-Adaptive Approximate Real-Time Computation for Mobility-Aware IoT Lifetime Optimization

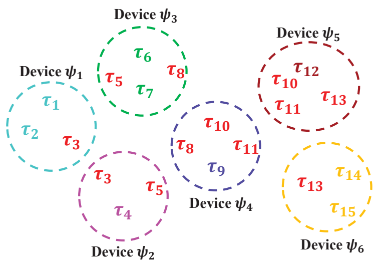
Kun Cao, Guo Xu, Junlong Zhou, Tongquan Wei, Mingsong Chen, and Shiyan Hu

***Abstract*—In recent years, the Internet of Things (IoT) has promoted many battery-powered emerging applications such as smart home, environmental monitoring, and human healthcare monitoring, where energy management is of particular importance. Meanwhile, there is an accelerated tendency towards mobility of IoT devices, either being transported by humans or being mobile by itself. Existing energy management mechanisms for battery-powered IoT fail to consider the two significant characteristics of IoT: the approximate real-time computation, and the mobility of IoT devices, resulting in unnecessary energy waste and network lifetime decay. In this paper, we explore mobility-aware network lifetime maximization for battery-powered IoT applications that perform approximate real-time computation under the Quality-of-Service (QoS) constraint. The proposed scheme is composed of offline and online stages. At offline stage, an optimal mobility-aware task schedule that maximizes network lifetime is derived by using mixed-integer linear programming (MILP) technique. Redundant executions due to mobility-incurred overlapping of a single task on different IoT devices are avoided for energy savings. At online stage, a performance-guaranteed and time-efficient QoS-adaptive heuristic based on cross-entropy method is developed to adapt task execution to the fluctuating QoS requirements. Extensive simulations based on synthetic applications and real-life benchmarks have been implemented to validate the effectiveness of our proposed scheme. Experimental results demonstrate that the proposed technique can achieve up to 169.52% network lifetime improvement compared to benchmarking solutions.**

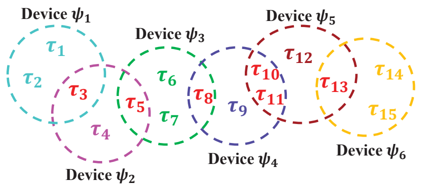
***Index Terms*—Internet of Things (IoT), network lifetime optimization, approximate real-time computation, Quality-of-Service (QoS), mobility.**

I. Introduction

Internet of Things (IoT), providing ubiquitous connectivity for anyone and anything at any time and any place, has been envisioned as one of the most promising networking paradigms for the information society [1]. In the past few years, IoT has promoted a variety of emerging applications, such as intelligent transportation, smart home, environmental monitoring, etc. According to a report from McKinsey Global Institute, the potential economic impact of IoT applications could be as much as \$11.1 trillion per year in 2025 [2]. Nonetheless, various challenges in IoT remain to be addressed, one of them being the efficient energy management [3]. Due to the considerations of either portability requirements, or the low-cost installation, deployment and maintenance, many IoT devices are powered by batteries that usually provide only a limited supply of energy. Therefore, efficient energy management is a key concern in battery-powered IoT, which aims at prolonging the network lifetime while meeting the Quality-of-Service (QoS) requirement.


Device ψ_1
τ_1 τ_2 τ_3
Device ψ_2
τ_3 τ_4 τ_5
Device ψ_3
τ_5 τ_6 τ_7 τ_8
Device ψ_4
τ_8 τ_9 τ_{10} τ_{11}
Device ψ_5
τ_{10} τ_{11} τ_{12} τ_{13}
Device ψ_6
τ_{13} τ_{14} τ_{15}

(a) The initial distribution of device locations.



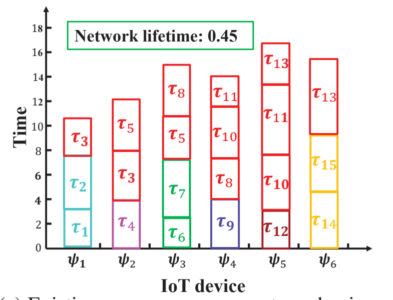
(b) The current distribution of device locations due to device mobility.

Fig. 1: An illustration to demonstrate IoT device mobility.

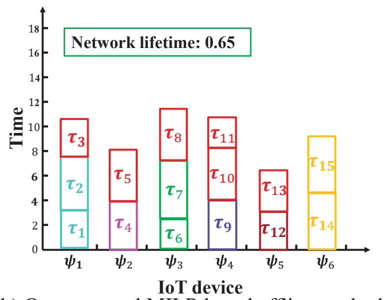
Several novel energy-efficient management mechanisms for battery-powered IoT have been proposed in literatures [4]–[7]. Colistra et al. [4] presented a consensus protocol for a distributed decision on resource allocation in a simulated IoT scenario. The proposed consensus protocol can improve network lifetime while preserving the required QoS. Luo et al. [5] introduced the concept of "equivalent node" to select relay node for optimal data transmission and energy conservation in WSN-based IoT. A probabilistic dissemination algorithm is designed to choose the optimal energy strategy and prolong the lifetime of whole network. Samie et al. [6] put forward a technique of computation offloading in a local IoT network for network lifetime optimization under bandwidth constraints. A hierarchical clustering approach was presented by Li et al. [7] to maximize the network lifetime of battery-powered IoT. This approach assigns different duty cycle ratios to the IoT devices that are located at different layers, resulting in the balance of energy consumption among these devices.

K. Cao, G. Xu, and T. Wei are with the Department of of Computer Science and Technology at East China Normal University, J. Zhou is with the School of Computer Science and Engineering at Nanjing University of Science and Technology, M. Chen is with the Shanghai Key Laboratory of Trustworthy Computing, and S. Hu is with the Department of Electrical and Computer Engineering at Michigan Technological University. T. Wei is the corresponding author: tqwei@cs.ecnu.edu.cn.

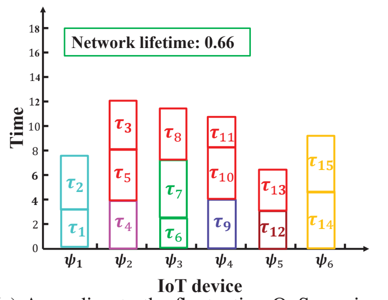
This work is partially supported by Shanghai Municipal Natural Science Foundation under Grant 16ZR1409000, National Natural Science Foundation of China under Grant 61802185 and 61872147, and Natural Science Foundation of Jiangsu Province under Grant BK20180470.



(a) Existing energy management mechanisms perform redundant execution of overlapping tasks τ_3, τ_5, τ_8, τ_{10}, τ_{11}, and τ_{13}. It is clear that these mechanisms lead to unnecessary energy waste.



(b) Our proposed MILP-based offline method avoids redundant execution of overlapping tasks τ_3, τ_5, τ_8, τ_{10}, τ_{11}, and τ_{13}. Any overlapping task is only executed once for energy saving.



(c) According to the fluctuating QoS requirement at runtime, our proposed online method quickly reallocates overlapping task τ_3 from IoT device ψ_1 to ψ_2 for achieving further energy saving.

Fig. 2: An illustration of existing mechanisms and our proposed solution to deal with the mobility of IoT devices shown in Fig. 1. Overlapping tasks are marked with red rectangles, while exclusive tasks are marked with rectangles of other colors that are consistent with the colors used in Fig. 1.

All the above works [4]–[7] on IoT energy management fail to consider two significant characteristics of IoT. One is the approximate task execution of IoT applications, and the other is the mobility of IoT devices. In real-world, many IoT applications can accept an inaccurate or approximate result in addition to a guaranteed accurate result [8]. For an approximate computation modeled task, its execution cycles are logically decomposed into two parts: one is a mandatory part and the other is an optional part [9], [10]. The mandatory part must execute to completion before the deadline and generate an acceptable result. The optional part refines and improves the result produced by the mandatory part. Some works [11]–[13] have utilized the concept of approximate computation to design task scheduling schemes for system QoS maximization. However, these works are tailored to maximize system QoS for an individual IoT device, and thus they cannot be directly applied to the lifetime optimization of the network consisting of multiple IoT devices.

Mobility of IoT devices is another important characteristic of IoT. There is an accelerated tendency towards mobility of IoT devices, either being transported by humans, like smartphones, or being mobile by itself, like a robot or an unmanned aerial vehicle. Fig. 1 illustrates an example of device mobility. As shown in Fig. 1a, tasks τ_3, τ_5, τ_8, τ_{10}, τ_{11} and τ_{13} appear in more than one IoT device in the initial distribution of device locations. For example, task τ_3 appears not only in device ψ_1, but also in device ψ_2. In this situation, device ψ_1 and device ψ_2 need to process task τ_3 separately because task τ_3 has different collected input data on different devices. Due to device mobility, the initial distribution of device locations may change, as demonstrated in Fig. 1b. In Fig. 1b, any one of tasks τ_3, τ_5, τ_8, τ_{10}, τ_{11} and τ_{13} has the same collected input data on different devices. We divide these tasks into two categories: one for exclusive task and one for overlapping task. An exclusive task is the task that can only be executed by the specific IoT device, and an overlapping task refers to the task that can be handled by any one of the IoT devices containing this task. For instance, as shown in Fig. 1b, task τ_1 and task τ_2 are exclusive tasks whereas task τ_3 is an overlapping task. This is because tasks τ_1 and τ_2 can only be executed by device ψ_1, while task τ_3 can be executed either by device ψ_1 or by device ψ_2.

However, as demonstrated in Fig. 2a, existing energy management mechanisms fail to take into consideration the mobility of IoT devices, and they allow overlapping task τ_3 to be executed once on IoT device ψ_1 in addition to being executed once on IoT device ψ_2. This results in overlapping task τ_3 being executed twice. In fact, overlapping task τ_3 only needs to be executed once. Obviously, redundant execution of overlapping tasks leads to unnecessary energy waste, thereby decaying the network lifetime. Meanwhile, since network QoS requirement may fluctuate at runtime, task execution should quickly adapt to the fluctuating QoS requirement when optimizing network lifetime. In this paper, we propose a QoS-adaptive mobility-aware scheme to optimize network lifetime of battery-powered IoT applications with approximate real-time computation requirements. To the best of our knowledge, this is the first attempt to optimize network lifetime with joint considerations of IoT device mobility and approximate real-time computation. As illustrated in Fig. 2b and Fig. 2c, our developed solution can not only avoid redundant execution of overlapping tasks for energy saving but also meet the fluctuating QoS requirement at runtime. The major contributions of this paper are summarized as follows.

- We investigate the problem of network lifetime optimization for battery-powered IoT applications that perform approximate real-time computation under the QoS constraint. In particular, the mobility of IoT devices is taken into consideration.
- We present a mixed-integer linear programming (MILP) based mobility-aware offline solution to the QoS-constrained network lifetime optimization, and propose a cross-entropy method based QoS-adaptive online heuristic that can quickly adapt task execution to the fluctuating QoS requirement at runtime.
- We conduct extensive simulation experiments to validate

the effectiveness of our proposed algorithms. Simulation results demonstrate that the proposed algorithms can achieve up to 169.52% network lifetime improvement compared to benchmarking schemes.

The remainder of the paper is organized as follows. Section II introduces network architecture and models. Section III describes problem definition and the overall framework of our optimization solution. Section IV presents the offline approach based on MILP technique while Section V shows the proposed online scheme based on cross-entropy method. The effectiveness of the proposed solution is verified in Section VI and concluding remarks are given in Section VII.

II. Network Architecture and Models

In this section, we briefly introduce our network architecture and models including task model, energy model, and network lifetime and QoS model.

A. Network Architecture

We consider a local network composed of N IoT devices $\psi = \{\psi_1, \psi_2, \cdots, \psi_N\}$ and a gateway, as shown in Fig. 3. All IoT devices are connected to the gateway by using a specific IoT wireless technology (e.g., HaLow, LPWAN, BLE [6]). Fig. 4 presents the architecture diagram of any IoT device in the local network. As illustrated in the figure, each IoT device ψ_i $(1 \leq i \leq N)$ is equipped with two major modules: one is an energy source module, and the other is an energy dissipation module. The energy source module is typically in the form of a battery (e.g., Li-ion). The energy dissipation module drains energy from the energy source module and consists of three parts: sensors, a heterogeneous multiprocessor system-on-chip (MPSoC), and a signal transceiver. The sensors sense physical phenomena and acquire input data required for tasks. The MPSoC system performs task execution, and the signal transceiver implements communication between the IoT device and the gateway. Assume that the MPSoC system on IoT device ψ_i is equipped with L_i heterogeneous processors, denoted by $\Theta_i = \{\Theta_{i,1}, \Theta_{i,2}, \cdots, \Theta_{i,L_i}\}$, where every processor $\Theta_{i,l}$ $(1 \leq l \leq L_i)$ is characterized by a given supply voltage/frequency pair $(v_{i,l}, f_{i,l})$.

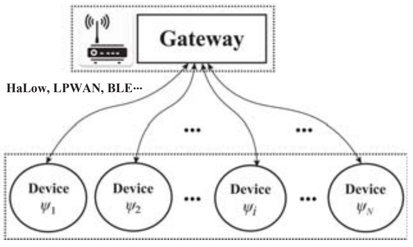


Fig. 3: A local network with N IoT devices and a gateway.

B. Application Model

Assume that real-time periodic Bag-of-Tasks (BoT) applications [14] are to be executed in the local network. In such a periodic application, tasks are independent, activated

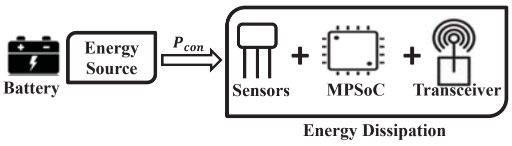


Fig. 4: The architecture diagram of any IoT device.

with a common period, and share a common deadline. Tasks are independent in the sense that there is no precedence or communication among tasks. Let T and D represent the period and deadline of the application, respectively. Suppose that ξ_i tasks of the application can be performed by IoT device ψ_i, denoted by $\Gamma_i = \{\tau_{1,i}, \tau_{2,i}, \cdots, \tau_{m,i}, \cdots, \tau_{\xi_i,i}\}$. Let C_i be a $\xi_i \times N$ matrix, where $c_i[m][j] \in C_i$ $(1 \leq j \leq N, j \neq i)$ gives the information whether task $\tau_{m,i} \in \Gamma_i$ $(1 \leq m \leq \xi_i)$ is an overlapping task or not. If $\tau_{m,i} \in \Gamma_j$ holds, then task $\tau_{m,i}$ is an overlapping task and hence $c_i[m][j]$ takes the value of 1. Otherwise task $\tau_{m,i}$ is an exclusive task and thus $c_i[m][j]$ is set to 0. Tasks are assumed to be heterogeneous and therefore the activity factor of a task, denoted by μ (ranging in (0,1]), is introduced to capture how intensively functional units are being utilized by the task [15].

In this work, we consider approximate computation real-time tasks [9], [10]. Every task $\tau_{m,i}$ is logically decomposed into two parts: i) a mandatory part with execution cycles $M_{m,i}$, which must execute to completion before the deadline and generate an acceptable result, and ii) an optional part with maximum execution cycles $O_{m,i}$, which refines and improves the result produced by the mandatory part[1]. We use a tuple $\tau_{m,i} : \{\mu_{m,i}, R_{m,i}, V_{m,i}, M_{m,i}, O_{m,i}, \alpha_{m,i}\}$ to characterize the approximate computation modeled task $\tau_{m,i}$. In the tuple, $\mu_{m,i}$ is the activity factor of task $\tau_{m,i}$. $R_{m,i}$ represents the amount of input data needed for task $\tau_{m,i}$ to start its execution. $V_{m,i}$ denotes the amount of data acquired by IoT device ψ_i for task $\tau_{m,i}$ in a sampling cycle. The $\alpha_{m,i}$ (ranging in [0,1]) is the optional execution factor of task $\tau_{m,i}$, which denotes the proportion of executed optional cycles to maximum optional cycles of task $\tau_{m,i}$. Therefore, the actual length $W_{m,i}$ of task $\tau_{m,i}$, measured by the total execution cycles, can be given by

$$W_{m,i} = M_{m,i} + \alpha_{m,i} \times O_{m,i}. \quad (1)$$

C. Energy Model

The overall power consumption P_{con} of any IoT device in the local network includes the power consumption P_{sen} of sensors, the power consumption P_{exe} of the MPSoC system, and the power consumption P_{com} of the signal transceiver. Therefore, P_{con} is expressed as

$$P_{con} = P_{sen} + P_{exe} + P_{com}. \quad (2)$$

The power consumption P_{exe} of an MPSoC system can be modeled as the sum of static power consumption P_{sta} and dynamic power consumption P_{dyn}, that is, [16]

$$P_{exe} = P_{sta} + P_{dyn}. \quad (3)$$

[1]For the sake of generality, we assume the weight of task $\tau_{m,i}$ that indicates the relative importance of the task has been integrated into $O_{m,i}$.

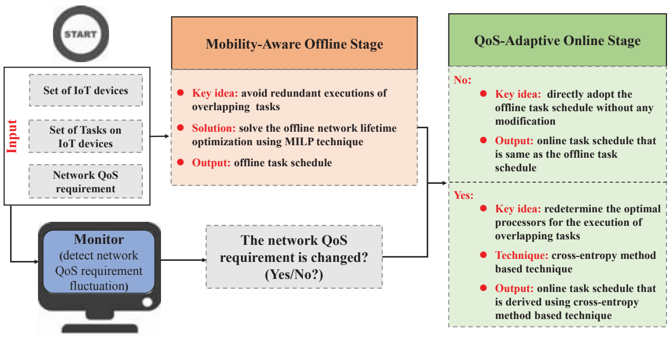

START
Input
Set of IoT devices
Set of Tasks on IoT devices
Network QoS requirement
Mobility-Aware Offline Stage
Key idea: avoid redundant executions of overlapping tasks
Solution: solve the offline network lifetime optimization using MILP technique
Output: offline task schedule
Monitor (detect network QoS requirement fluctuation)
The network QoS requirement is changed? (Yes/No?)
QoS-Adaptive Online Stage
No:
Key idea: directly adopt the offline task schedule without any modification
Output: online task schedule that is same as the offline task schedule
Yes:
Key idea: redetermine the optimal processors for the execution of overlapping tasks
Technique: cross-entropy method based technique
Output: online task schedule that is derived using cross-entropy method based technique

Fig. 5: An overview of our proposed solution.

P_{sta} is independent of switching activity and can be regarded as a processor-dependent constant while P_{dyn} is related to charging and discharging of gates in the circuits. The dynamic power consumption of processor $\Theta_{i,l}$ when executing task $\tau_{m,i}$ at the supply voltage/frequency pair $(v_{i,l}, f_{i,l})$ is given by [16]

$$P_{dyn}(m,i,l) = \mu_{m,i} C_{i,l}^{eff} v_{i,l}^2 f_{i,l}, \tag{4}$$

where $C_{i,l}^{eff}$ is the effective switching capacitance of processor $\Theta_{i,l}$. Therefore, the overall energy consumption of IoT device ψ_i during scheduling horizon T is hence calculated as

$$\begin{aligned}\mathbb{E}_{con}^{energy}(i) = P_{com}(i) \times T + \sum_{l=1}^{L_i} P_{sta}(i,l) \times T + \\ \sum_{l=1}^{L_i} \sum_{\tau_{m,i} \in \Gamma_{i,l}} (\mu_{m,i} C_{i,l}^{eff} v_{i,l}^2 W_{m,i} + P_{sen}(i) \frac{R_{m,i}}{V_{m,i}}),\end{aligned} \tag{5}$$

where $P_{sen}(i)$ and $P_{com}(i)$ are the power consumption of sensors and the signal transceiver on IoT device ψ_i, respectively. $P_{sta}(i,l)$ is the static power consumption of processor $\Theta_{i,l}$. $\Gamma_{i,l} \subseteq \Gamma_i$ is the task subset consisting of these tasks that are executed on processor $\Theta_{i,l}$.

D. Network Lifetime and QoS Model

The lifetime of an IoT device can be defined as the ratio of its remaining energy to its power consumption [6]. Therefore, the lifetime $\mathbb{T}_i$ of IoT device ψ_i is given by

$$\mathbb{T}_i = \frac{\mathbb{E}_{rem}^{energy}(i)}{\mathbb{E}_{con}^{energy}(i)/T}, \tag{6}$$

where $\mathbb{E}_{rem}^{energy}(i)$ is the remaining energy of IoT device ψ_i and it is expressed as the difference between the available energy supply $\mathbb{E}_{sup}^{energy}(i)$ provided by the energy source module and energy consumption $\mathbb{E}_{con}^{energy}(i)$:

$$\mathbb{E}_{rem}^{energy}(i) = \mathbb{E}_{sup}^{energy}(i) - \mathbb{E}_{con}^{energy}(i). \tag{7}$$

The network lifetime, denoted by $\mathbb{T}$, can be thus represented as the minimal lifetime of the IoT device in the local network

$$\mathbb{T} = \min\{\mathbb{T}_1, \mathbb{T}_2, \cdots, \mathbb{T}_i, \cdots, \mathbb{T}_N\}. \tag{8}$$

It has been shown that the QoS of a task can be represented as a linear or concave function of optional cycles of the task [9]. The more cycles the optional part of the task executes, the higher QoS the task generates. Thus the network QoS can be defined as the sum of the executed CPU cycles of optional parts of all the approximate computation real-time tasks in the network. It is denoted by Q and is expressed as

$$Q = \sum_{i=1}^{N} Q_i = \sum_{i=1}^{N} \sum_{m=1}^{\xi_i} \alpha_{m,i} O_{m,i}, \tag{9}$$

where Q_i is the QoS achieved by IoT device ψ_i, $\alpha_{m,i}$ is the optional execution factor of task $\tau_{m,i}$, and $O_{m,i}$ is the the maximum optional cycles of task $\tau_{m,i}$.

III. Problem Definition and Overall Framework

In this section, we first give a definition of the network lifetime optimization problem, and then show the overall framework of our proposed solution.

A. Problem Definition

In this paper, we concentrate on the scenario where the available energy supply of each IoT device during the scheduling horizon is sufficient to complete the execution of the mandatory parts of all approximate tasks on the IoT device, but sufficient or insufficient to perform the execution of the optional parts of all approximate tasks on the IoT device. The studied problem is formally defined as follows. Given a mobility-aware local network, the target application, and the network QoS requirement, design a task allocation and scheduling scheme to maximize network lifetime under the constraints that i) the mandatory parts of tasks must be completed before the deadline; ii) the network QoS requirement is satisfied; and iii) for each IoT device, its energy consumption cannot exceed its available energy supply during the scheduling horizon.

B. Overview of Our Proposed Solution

Fig. 5 presents an overview of our proposed solution to QoS-constrained network lifetime optimization. As shown in the figure, our solution is composed of two stages: a mobility-aware offline stage and a QoS-adaptive online stage. At offline stage, an optimal mobility-aware task schedule that can maximize network lifetime while satisfying all design constraints is derived by using MILP technique. Redundant executions due to mobility-incurred overlapping of a single task on different IoT devices are avoided for energy savings. Since the network QoS requirement may fluctuate at runtime, the offline task schedule needs to be adjusted at runtime at the minimum cost. Therefore, we provide a low-cost yet high-performance QoS-adaptive online scheme. In the proposed online scheme, if the gateway monitors no change in network QoS requirement [2] at runtime during the current time interval $[t, t+T]$ for executing new arriving task instances, the optimal task schedule produced at offline stage will be directly adopted without any modification to schedule tasks; else an online heuristic will be invoked. The main idea of the heuristic is

[2]We assume that the time granularity of changes in QoS requirements at runtime cannot be smaller than the application period.

to redetermine the optimal processors for the execution of overlapping tasks by using cross-entropy method such that the network lifetime is optimized and task execution quickly adapts to the fluctuating QoS requirement.

IV. MILP Based Mobility-Aware Offline Network Lifetime Optimization

In this section, we present an MILP based mobility-aware offline approach to solve the QoS-constrained network lifetime optimization problem.

A. MILP Formulation

For ease of presentation, let $\Gamma = \{\Gamma_1 \cup \cdots \cup \Gamma_i \cup \cdots \cup \Gamma_N\}$ be the task set consisting of all exclusive tasks and overlapping tasks on N IoT devices. An overlapping task on different IoT devices only appears once in task set Γ. Let M_ψ^m be the set of the indexes of these IoT devices that can execute task $\tau_m \in \Gamma$. Let M_Θ^i be the set of the indexes of these processors on IoT device ψ_i. We define the following variables.

$$A_{m,i,l} = \begin{cases} 1 & \text{if task } \tau_m \in \Gamma \text{ is assigned to} \\ & \text{processor } \Theta_{i,l},\ i \in M_\psi^m, l \in M_\Theta^i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$S_m : \text{the start time of task } \tau_m \quad (11)$$

$$\alpha_m : \text{the optional execution factor of task } \tau_m \quad (12)$$

1) Objective: The objective function is expressed as

$$\text{maximize: } \mathbb{T}(A_{m,i,l}, S_m, \alpha_m), \forall \tau_m \in \Gamma, i \in M_\psi^m, l \in M_\Theta^i. \quad (13)$$

2) Constraints: The constraints are summarized below.

- *Every task τ_m is assigned to exactly one processor.*

$$\sum_{i \in M_\psi^m} \sum_{l \in M_\Theta^i} A_{m,i,l} = 1, \forall \tau_m \in \Gamma. \quad (14)$$

- *The optional execution factor α_m of task τ_m takes the value from interval $[0,1]$.*

$$\alpha_m \in [0,1], \forall \tau_m \in \Gamma. \quad (15)$$

- *The mandatory part of every task τ_m meets its deadline.*

$$S_m + \sum_{i \in M_\psi^m} \sum_{l \in M_\Theta^i} A_{m,i,l}\left(\frac{R_m}{V_{m,i}} + \frac{M_m}{f_{i,l}}\right) \leq D, \forall \tau_m \in \Gamma. \quad (16)$$

- *For each IoT device, its energy consumption cannot exceed its available energy supply during the scheduling horizon.* This constraint is equivalent to the condition that the lifetime of any IoT device should not be less than zero during the scheduling horizon.

$$\mathbb{T}_i(A_{m,i,l}, S_m, \alpha_m) \geq 0, \forall \tau_m \in \Gamma, i \in M_\psi^m, l \in M_\Theta^i. \quad (17)$$

- *The offline QoS requirement Q_{goal}^{off} should be met.*

$$\sum_{\tau_m \in \Gamma} \alpha_m \times O_m \geq Q_{goal}^{off}. \quad (18)$$

- *Tasks have no overlapping executions in the same processor.* Let S_p and S_q denote the start time of task τ_p and task τ_q when the two tasks are executed on processor $\Theta_{i,l}$, respectively, and the variable $Smin_{(p,q)}$ denote the minimum of the S_p and S_q. Thus, the equation $Smin_{(p,q)}$ = $\min(S_p, S_q)$ holds. The $b_{p,q}$ is an auxiliary binary decision variable indicating the relationship of $Smin_{(p,q)}$, S_p, and S_q. If $S_p < S_q$ holds, that is, $Smin_{(p,q)} = S_p$, then $b_{p,q}$ = 1; else $b_{p,q}$ = 0. $\mathcal{H}$ is a large enough constant number and is set to 10000 in the experiment. Similar to $Smin_{(p,q)}$, the variable $Smax_{(p,q)}$ is introduced to indicate the maximum of the S_p and S_q. That is, $Smax_{(p,q)}$ = $\max(S_p, S_q)$ holds. Two auxiliary variables $h_{p,q}$ and $g_{p,q}$ are also introduced as pseudo-linear constraints to facilitate the formulation. The following constraints must be satisfied to avoid overlapping executions.

$$Smin_{(p,q)} \leq S_p \quad (19)$$

$$Smin_{(p,q)} \leq S_q \quad (20)$$

$$Smin_{(p,q)} \geq S_p - \mathcal{H} \times (1 - b_{p,q}) \quad (21)$$

$$Smin_{(p,q)} \geq S_q - \mathcal{H} \times b_{p,q} \quad (22)$$

$$b_{p,q} = 1, 0 \quad (23)$$

$$Smax_{(p,q)} = S_p + S_q - Smin_{(p,q)} \quad (24)$$

$$h_{p,q} = \mathcal{H} \times (S_p - Smax_{(p,q)}) + S_q \quad (25)$$

$$S_p - h_{p,q} \geq \frac{M_q + \alpha_q O_q}{f_{i,l}} + \frac{R_q}{V_{q,i}} \quad (26)$$

$$g_{p,q} = \mathcal{H} \times (S_q - Smax_{(p,q)}) + S_p \quad (27)$$

$$S_q - g_{p,q} \geq \frac{M_p + \alpha_p O_p}{f_{i,l}} + \frac{R_p}{V_{p,i}} \quad (28)$$

B. Algorithm of MILP Based Offline Approach

Algorithm 1: Mobility-Aware MILP Offline Scheme

```
Input: 1) Task set $\Gamma$, 2) Device set $\psi$,
       3) Offline QoS requirement $Q_{goal}^{off}$;
Output: Offline schedule table $\Omega^{off}$;
1 Generate schedule table $\Omega^{off}$ for $N$ IoT devices by
  solving the MILP formulated in Section IV-A;
2 Return Schedule table $\Omega^{off}$.
```

Algorithm 1, performed by the gateway, demonstrates the pseudo-code of our proposed MILP based offline lifetime optimization scheme. Inputs to the algorithm are the task set, device set, and offline QoS requirement. The optimal offline task scheduling that can maximize network lifetime under the network QoS and real-time constraints is derived by solving the MILP formulated in Section IV-A.

V. Cross-Entropy Method Based QoS-Adaptive Online Lifetime Optimization

The proposed MILP based offline lifetime optimization scheme generates an optimal task schedule that maximizes network lifetime while satisfying all design constraints. However, due to the fact that the network QoS requirement may fluctuate at runtime, the offline task schedule needs to be adjusted at runtime. Therefore, we provide a low-cost yet high-performance QoS-adaptive online task scheduling algorithm based on cross-entropy method. Our online scheme is designed to redetermine the optimal processors for overlapping task execution such that not only is the network lifetime optimized but also task execution can quickly adapts to the fluctuating QoS requirement. In this section, we first briefly introduce the theoretic basis of cross-entropy method, and then describe our proposed online scheme based on cross-entropy method.

A. Theoretical Basis of Cross-Entropy Method

The basic idea of cross-entropy method is to transform the deterministic optimization problem into its corresponding stochastic optimization problem [17]. By using an iterative sampling algorithm, the stochastic optimization problem is then addressed. In each iteration of the sampling algorithm, multiple random samples representing solutions to the deterministic optimization problem are generated, and these random samples converge to the optimal or near-optimal solution in a probabilistic way. Compared to conventional optimization techniques such as genetic algorithm [18] or particle swarm optimization algorithm [19], cross-entropy method has a complete theoretical basis, and has been proven to be an effective solution to optimization problems [17]. For more details on the theoretical basis of cross-entropy method, readers are suggested to refer to literature [17].

Now let us consider a discrete or combinatorial optimization problem that the goal is to find the optimal mapping $\beta^* : \mathbb{R}^n \to \mathbb{R}^{n^*}$ such that the value of function $F(x)$ about variable x in the state space $\mho$ is maximized when $x = \beta^*$, that is,

$$F(\beta^*) = \gamma^* = \max_{x \in \mho} F(x). \tag{29}$$

This optimization problem is associated with the following estimation of the probability [17]

$$\lambda(\gamma) = P_u(F(X) \geq \gamma) = E_u(\Phi_{\{F(X) \geq \gamma\}}). \tag{30}$$

$X = (X_1, \cdots, X_z, \cdots, X_Z)$ is a sample vector consisting of Z random samples produced by the corresponding probability $P(x, u)$. $P(x, u)$ is in parametric class of probabilities $\{P(x, \iota), \iota \in \ell\}$ with parameter ι being set to u. γ is a threshold or level parameter, and $P_u(F(X) \geq \gamma)$ represents the probability of $F(X) \geq \gamma$. $E_u(\Phi_{\{F(X) \geq \gamma\}})$ denotes the expected value of $\Phi_{\{F(X) \geq \gamma\}}$, and $\Phi_{\{F(X) \geq \gamma\}}$ is the indicator function, that is,

$$\Phi_{\{F(x) \geq \gamma\}} = \begin{cases} 1 & F(x) \geq \gamma \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

The cross-entropy method aims to find the minimal γ such that $\lambda(\gamma)$ (i.e., the probability of $F(X) \geq \gamma$) approaches 0. It is then that the probability of $F(X) < \gamma$ approaches 1, indicating γ is the minimal upper bound on $F(x)$ for $\forall x \in \mho$ and thus the optimal solution to Eq. (29). The main steps of cross-entropy method are summarized and described as follows.

1) Set iteration counter $t \leftarrow 1$ and initialize probability vector P_0.
2) Generate Z samples $\{X_1, \cdots, X_z, \cdots, X_Z\}$ based on probability vector P_{t-1}, and calculate sample performance $\{F(X_1), \cdots, F(X_z), \cdots, F(X_Z)\}$.
3) Select B^{elite} samples with best performance, and let ϑ denote the set of indices of best samples. Derive threshold γ_t using the average performance of B^{elite} best samples:
$$\gamma_t \leftarrow \frac{1}{B^{elite}} \sum_{\epsilon \in \vartheta} F(X_\epsilon). \tag{32}$$
4) Obtain probability vector P_t using
$$P_{t,a,b} = \frac{\sum_{z=1}^{Z} \Phi_{\{F(X_z) \geq \gamma_t\}} \Phi_{\{x_{z,a} = b\}}}{\sum_{z=1}^{Z} \Phi_{\{F(X_z) \geq \gamma_t\}}}, \quad a = 1, 2, \cdots, n;\ b = 1, 2, \cdots, n^*, \tag{33}$$
where $x_{z,a}$ is the ath element in sample X_z, and $P_{t,a,b}$ indicates the probability of $x_{z,a}$ being mapped to b (i.e., $x_{z,a} = b$) at tth iteration.
5) If predefined stop criterion is met, exit; otherwise, set $t \leftarrow t + 1$, and return to 2).

B. Our Online Scheme Based on Cross-Entropy Method

The objective of online scheduling includes not only generating a high quality task schedule, but also minimizing the runtime scheduling overheads. To this end, the main idea of our QoS-adaptive online scheme is to reallocate overlapping tasks to optimal processors by using cross-entropy method such that the network lifetime is optimized while task execution can be quickly adapted to the fluctuating QoS requirement.

Algorithm 2, executed by the gateway, shows the pseudo-code of our proposed cross-entropy method based online scheme. Line 1 of the algorithm initializes the online task schedule Ω^{on} to the offline task schedule Ω^{off}. If no change in network QoS requirement at runtime is detected, the optimal task schedule produced at offline stage will be directly adopted without any modification (lines 2-3); else an online heuristic based on cross-entropy method will be invoked (lines 4-24). Lines 5-6 of the algorithm construct a task set and a processor set. For each IoT device, lines 7-9 first delete all overlapping tasks from its schedule table and then updates the start times of all exclusive tasks in the schedule table. The available energy supply that can be used for overlapping task execution for each IoT device is calculated in line 10. Lines 11-12 first derive network QoS difference and then equally assign the network QoS difference to overlapping tasks. Lines 13-23 perform the reallocation process for overlapping tasks by using cross-entropy method. Lines 13-14 initialize the iteration counter, maximal iteration number, and probability vector. Line 15 determines whether the algorithm continues the iteration process or exits the optimization. In each iteration, lines 16-17 first generate multiple samples according to current probability vector and then select feasible samples from the generated samples. If no feasible sample is found and the current iteration counter is no less than 2, lines 18-19 terminate the iterative process. Lines 20-21 derive the performance of feasible samples and accordingly update the threshold. Lines

Algorithm 2: QoS-Adaptive Online Scheme

```
Input: 1) Task set $\Gamma$, 2) Device set $\psi$, 3) Online QoS
       requirement $Q_{goal}^{on}$, 4) Offline schedule table $\Omega^{off}$
Output: Online schedule table $\Omega^{on}$;
1  Initialize $\Omega^{on} \leftarrow \Omega^{off}$;
2  if $Q_{goal}^{on} == Q_{goal}^{off}$ then
3      Return $\Omega^{on}$.
4  else
5      Construct task set $\Gamma'$ that stores all overlapping tasks;
6      Create processor set $\Psi$ where each element $\Psi_m \in \Psi$
       indicates the set of these processors that overlapping
       task $\tau_m \in \Gamma'$ can be assigned to;
7      for each IoT device $\psi_i$ do
8          Delete all overlapping tasks from $\Omega_i^{on}$;
9          Update start times of all exclusive tasks in $\Omega_i^{on}$;
10         Derive its available energy supply using Eq. (7);
11     Calculate QoS difference $\triangle Q = Q_{goal}^{on} - Q_{goal}^{off}$;
12     Assign $\triangle Q$ equally to overlapping tasks in task set
       $\Gamma'$ while satisfying Eq. (15);
13     Set $t = 1$, and initialize maximal iteration number
       $t^{max}$;
14     Initialize probability vector $P_0$: for each overlapping
       task $\tau_m \in \Gamma'$, set the probability of allocating $\tau_m$ to
       the processors in $\Psi_m$ to $1/sizeof(\Psi_m)$ and the
       processors not in $\Psi_m$ to 0;
15     while $t \leq t^{max}$ do
16         Generate $J$ samples according to $P_{t-1}$ using
           Latin hypercube importance sampling [20];
17         Select $Z$ feasible samples meeting Eq. (16) and
           Eq. (17) from $J$ samples using
           acceptance-rejection method [21];
18         if find no feasible sample and $t \geq 2$ then
19             break;
20         Calculate network lifetime for each feasible
           sample using Eq. (8);
21         Obtain threshold $\gamma_t$ using Eq. (32);
22         Derive probability vector $P_t$ using Eq. (33);
23         $t \leftarrow t + 1$;
24     Return $\Omega^{on}$ that is the feasible sample with optimal
       network lifetime at $(t-1)$th iteration.
```

22-23 update probability vector and iteration counter used for next iteration. Line 24 returns the online schedule table generated based on the sample with best performance at the $(t-1)$th iteration.

VI. Evaluation

A. Experimental Settings

Two sets of simulations have been implemented to validate the effectiveness of our proposed network lifetime optimization solution. The first set of simulations is based on synthetic applications while the second set of simulations is based on real-life benchmarks. In each set of simulations, two local networks that consist of, respectively, 5 (N=5) and 10 (N=10) IoT devices are adopted. In addition to comparing the performance achieved by our proposed offline method with that of three benchmarking algorithms RAN, CTF [22], and HWG [23], we also compare the performance achieved by the proposed online approach with that of three benchmarking algorithms GEN, PSO, and GCS [12]. The mentioned benchmarking algorithms are described below.

- **RAN** is an algorithm that randomly selects tasks whose optional parts are to be completed for the purpose of meeting the network QoS requirement.
- **CTF** [22] is a method that assigns QoS-critical jobs higher priorities such that their optional cycles can be completed first. The QoS-critical jobs are defined as tasks with larger maximum optional cycles.
- **HWG** [23] is a state-of-the-art approach that integrates a worst-fit based partitioning heuristic with genetic algorithm to generate a task allocation that reduces energy consumption while satisfying all design constraints.
- **GEN** is a method that replaces cross-entropy method (lines 13-24 of Algorithm 2) with genetic algorithm [18] to perform network lifetime optimization.
- **PSO**, similar to GEN, is an approach that uses the particle swarm optimization algorithm [19] to replace the cross-entropy method adopted in Algorithm 2.
- **GCS** [12] is a dynamic scheduling method that determines the best allocation of slack cycles for maximizing QoS. For fair comparison, we stop the running of GCS when the network QoS requirement is satisfied.

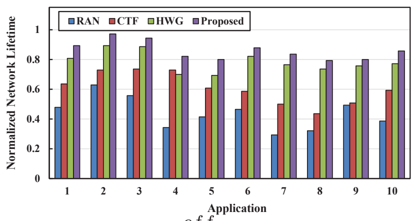
All the algorithms above are implemented in C++, and the simulations are performed on a machine with Intel i7 Dual-Core 3.5GHz processor and 16GB memory. We perform 1000 experiments to obtain the average of the simulation data.

TABLE I: Parameters of the simulated processor model [24].

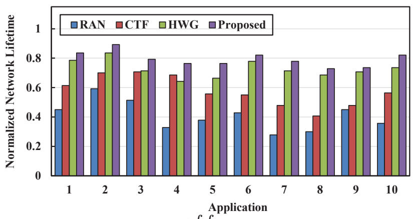
No.	v (V)	f (GHz)	C^{eff} (nF)
1	0.85	0.8010	13.0
2	0.90	0.8291	12.0
3	0.95	0.8553	14.0
4	1.00	0.8797	15.0
5	1.05	0.9027	17.0
6	1.10	1.0000	16.0
7	1.15	1.0527	18.0
8	1.20	1.1236	16.0
9	1.25	1.1867	19.0
10	1.30	1.2500	18.0

B. Simulation for Synthetic Applications

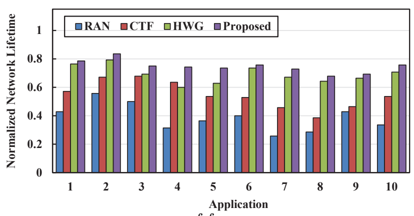
In this set of simulations, a set of synthetic MPSoC systems is adopted to execute synthetic applications. The supply voltage v, operating frequency f, and effective switching capacitance C^{eff} of ten processors built on 65nm technology are listed in TABLE I. The number of processors of an MPSoC system is randomly selected in the range $[2, 10]$. Task activity factors are uniformly distributed in the interval $[0.4, 1]$ [15]. The worst case execution cycles (WCEC) of tasks are assumed to be in the range of $[4 \times 10^9, 6 \times 10^{10}]$ [25]–[27]. Each task $\tau_{m,i}$ is instantiated by haphazardly picking two WCECs



(a) $Q_{goal}^{off} = 0.5$

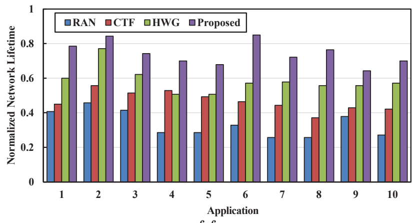


(b) $Q_{goal}^{off} = 0.6$

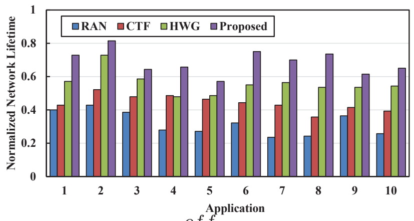


(c) $Q_{goal}^{off} = 0.7$

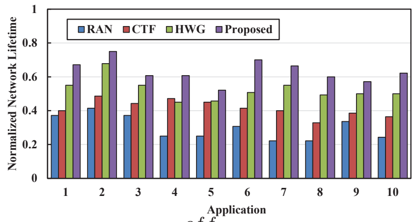
Fig. 6: The network lifetime when running 10 synthetic applications under $N = 5$ and varying offline QoS requirements.



(a) $Q_{goal}^{off} = 0.5$



(b) $Q_{goal}^{off} = 0.6$



(c) $Q_{goal}^{off} = 0.7$

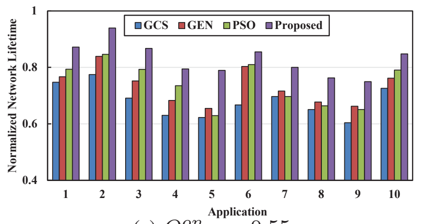
Fig. 7: The network lifetime when running 10 synthetic applications under $N = 10$ and varying offline QoS requirements.

from the range. One is for its mandatory part $M_{m,i}$ and the other is for its maximum optional part $O_{m,i}$. The number of tasks on individual IoT device is randomly selected from the range $[10, 30]$. The amount of input data needed for each task is randomly picked from the interval $[1, 20]$Mb [28]. The information matrix C_i is also generated in a random way. The common deadline D of tasks is assumed to be $1.5 \times \max\{\sum_{\tau_{m,i}\in\Gamma_i} M_{m,i}/f_{i,max}|\ i = 1, 2, \cdots, N\}$, where $f_{i,max}$ is the maximum frequency supported by IoT device ψ_i [24], [29]. The application period T is set to be the same as the common deadline D. A set of Li-lion battery banks equipped with $[4, 10]$Ah normal capacity and $[15, 20]$V terminal voltage [30] is served as energy supply modules. For the battery bank on each IoT device, its normal capacity and terminal voltage are generated at random from corresponding intervals. The amount of data per sampling cycle sensed by sensors on IoT device ψ_i to acquire the input data required for task $\tau_{m,i}$ is arbitrarily chosen from the interval $[1, 5]$Mb/s [31]. The power consumption of sensors and that of the signal transceiver on each IoT device are indiscriminately picked from the ranges $[0.5, 2]$W and $[1, 5]$W, respectively [32]. The maximum iteration number t^{max} is set to 10.

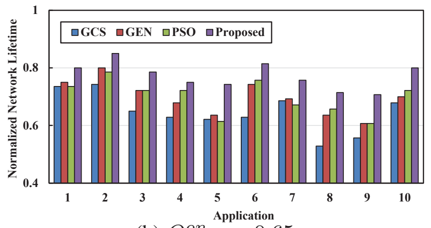
1) Performance Comparison for Offline Approach: We utilize a normalized network QoS requirement in the comparative study, the maximum of which is set to 1. To be specific, the network QoS requirement is normalized against the sum of the maximum optional cycles of all tasks on N IoT devices. Fig. 6 presents the normalized network lifetime under varying offline QoS requirements when the local network consists of 5 IoT devices executing synthetic applications. The results given in the figure clearly show that our proposed offline scheme achieves better performance in terms of network lifetime compared to the three benchmarking algorithms. For example, in the case of $Q_{goal}^{off} = 0.5$, the network lifetime achieved by the proposed approach is 96.25%, 41.86%, and 9.76% higher on average than that of RAN, CTF, and HWG, respectively. In the case of $Q_{goal}^{off} = 0.6$, the network lifetime achieved by the proposed approach is 94.57%, 38.18%, and 9.24% higher on average than that of RAN, CTF, and HWG, respectively. In the case of $Q_{goal}^{off} = 0.7$, the network lifetime achieved by the proposed approach is 92.8%, 36.6%, and 8.18% higher on average than that of RAN, CTF, and HWG, respectively.

Fig. 7 plots the normalized network lifetime under varying offline QoS requirements when the network consists of 10 IoT devices executing synthetic applications. Similar to the results shown in Fig. 6, our proposed offline algorithm significantly improves the network lifetime. For instance, in the scenario of $Q_{goal}^{off} = 0.5$, the network lifetime achieved by the proposed approach is 122.22%, 59.02%, and 27.14% higher on average than that of RAN, CTF, and HWG, respectively. In the scenario of $Q_{goal}^{off} = 0.6$, the network lifetime achieved by the proposed approach is 115.47%, 55.5%, and 23.05% higher on average than that of RAN, CTF, and HWG, respectively. In the scenario of $Q_{goal}^{off} = 0.7$, the network lifetime achieved by the proposed approach is 111.08%, 50.21%, and 20.60% higher on average than that of RAN, CTF, and HWG, respectively.

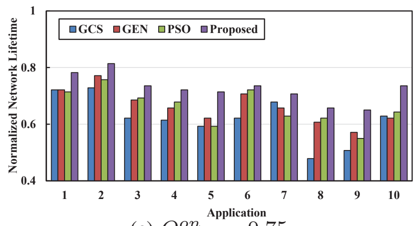
2) Performance Comparison for Online Approach: Fig. 8 depicts the normalized network lifetime under varying online QoS requirements when the network consists of 5 IoT devices. As demonstrated in the figure, our proposed online scheme achieves remarkable network lifetime improvement compared to the three benchmarking algorithms. To be specific, in the case of $Q_{goal}^{on} = 0.55$, the network lifetime achieved by the proposed approach is 21.54%, 13.14%, and 11.74% higher on average than that of GCS, GEN, and PSO, respectively. In the case of $Q_{goal}^{on} = 0.65$, the network lifetime achieved by the proposed approach is 19.58%, 10.87%, and 10.42% higher on average than that of GCS, GEN, and PSO, respectively. In the



(a) $Q_{goal}^{on} = 0.55$

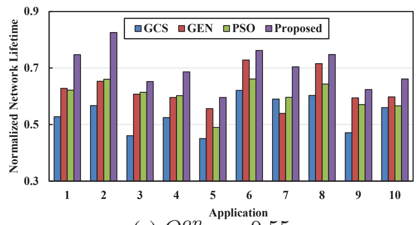


(b) $Q_{goal}^{on} = 0.65$

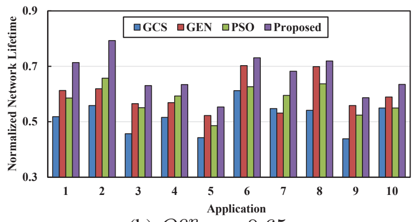


(c) $Q_{goal}^{on} = 0.75$

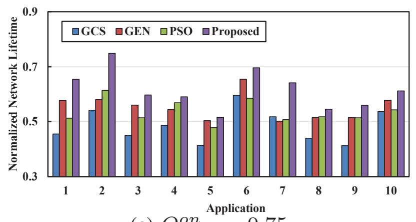
Fig. 8: The network lifetime when running 10 synthetic applications under $N = 5$ and varying online QoS requirements.



(a) $Q_{goal}^{on} = 0.55$



(b) $Q_{goal}^{on} = 0.65$



(c) $Q_{goal}^{on} = 0.75$

Fig. 9: The network lifetime when running 10 synthetic applications under $N = 10$ and varying online QoS requirements.

case of $Q_{goal}^{on} = 0.75$, the network lifetime achieved by the proposed approach is 17.13%, 9.55%, and 9.90% higher on average than that of GCS, GEN, and PSO, respectively.

TABLE II: The runtime (unit: s) on average for deriving the scheduling of 10 synthetic applications under $N = 5$.

Q_{goal}^{on}	The number of IoT device: $N = 5$					
	Proposed	GEN	PSO	GCS	Min. Sp	Max. Sp
0.55	2.06	4.18	3.91	27.69	1.90x	13.44x
0.65	2.12	4.72	4.10	38.60	1.93x	14.76x
0.75	2.22	5.91	4.39	49.28	1.97x	22.15x

Fig. 9 plots the normalized network lifetime under varying online QoS requirements when the network consists of 10 IoT devices. As shown in the figure, our proposed online algorithm achieves better lifetime improvement compared to the three benchmarking algorithms. Specifically, in the scenario of $Q_{goal}^{on} = 0.55$, the network lifetime achieved by the proposed approach is 30.39%, 12.72%, and 16.28% higher on average than that of GCS, GEN, and PSO, respectively. In the scenario of $Q_{goal}^{on} = 0.65$, the network lifetime achieved by the proposed approach is 28.89%, 11.88%, and 15.06% higher on average than that of GCS, GEN, and PSO, respectively. In the scenario of $Q_{goal}^{on} = 0.75$, the network lifetime achieved by the proposed approach is 26.99%, 10.49%, and 12.85% higher on average than that of GCS, GEN, and PSO, respectively.

TABLE III: The runtime (unit: s) on average for deriving the scheduling of 10 synthetic applications under $N = 10$.

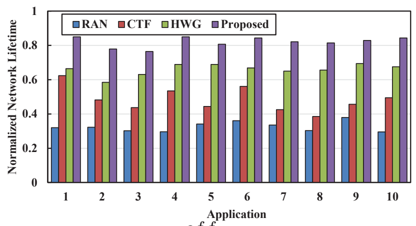
Q_{goal}^{on}	The number of IoT device: $N = 10$					
	Proposed	GEN	PSO	GCS	Min. Sp	Max. Sp
0.55	13.28	27.88	25.49	249.04	1.92x	18.76x
0.65	17.12	33.85	26.02	351.12	1.52x	20.50x
0.75	19.91	37.17	31.06	523.17	1.56x	26.27x

TABLE II demonstrates the runtime on average for deriving the scheduling of synthetic applications using the proposed online scheme and benchmarking schemes when the network consists of 5 IoT devices. The metric Sp denotes the speedup achieved by the proposed method when compared to the baseline approach in terms of average runtime. It can be easily seen from the table that our proposed online scheme greatly reduces the runtime for deriving task scheduling. For example, when online QoS requirement Q_{goal}^{on} is set to 0.55, our proposed online scheme can achieve up to 13.44 times of speedup. TABLE III presents the runtime on average for deriving the scheduling of synthetic applications using the proposed online scheme and benchmarking schemes when the network consists of 10 IoT devices. Similar to the results presented in TABLE II, our proposed online scheme also achieves better performance in terms of runtime compared to the three benchmarking schemes. For instance, when online QoS requirement Q_{goal}^{on} is set to 0.55, our proposed online scheme can achieve up to 18.76 times of speedup.

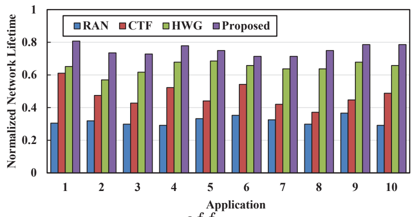
C. Simulation for Real-Life Applications

In this set of simulations, a set of practical MPSoC systems that are constructed on Intel Core Duo processor, Intel Xeon processor, AMD Athlon processor, TI DSP processor, and S-PARC64 processor is adopted for simulation. All the processor parameters of these MPSoC systems can be found in [14]. The tool MEGA [33] that incorporates approximate computation is utilized to generate real-life benchmarks. The settings of task number, sensors, signal transceiver, and Li-lion battery bank on each IoT device are the same as that of simulation for synthetic applications.

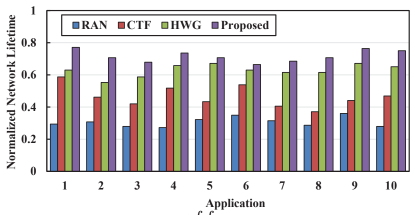
1) Performance Comparison for Offline Approach: Fig. 10 demonstrates the normalized network lifetime under varying offline QoS requirements when the local network consists of 5 IoT devices executing real-life tasks. The results given in



(a) $Q^{off}_{goal} = 0.5$

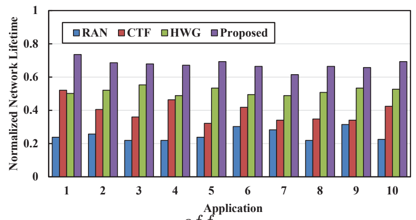


(b) $Q^{off}_{goal} = 0.6$

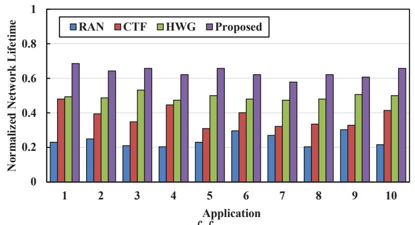


(c) $Q^{off}_{goal} = 0.7$

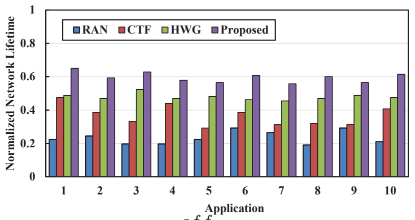
Fig. 10: The network lifetime when running 10 real-life applications under $N = 5$ and varying offline QoS requirements.



(a) $Q^{off}_{goal} = 0.5$

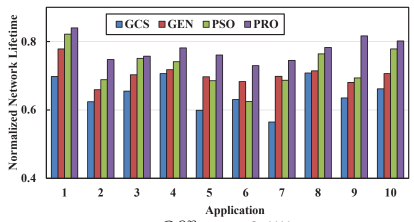


(b) $Q^{off}_{goal} = 0.6$

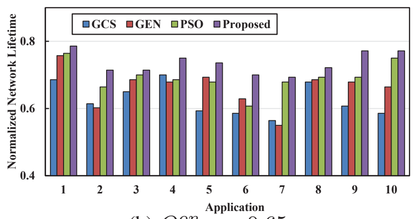


(c) $Q^{off}_{goal} = 0.7$

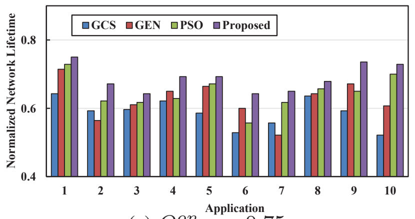
Fig. 11: The network lifetime when running 10 real-life applications under $N = 10$ and varying offline QoS requirements.



(a) $Q^{on}_{goal} = 0.55$

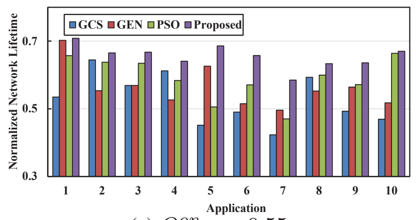


(b) $Q^{on}_{goal} = 0.65$

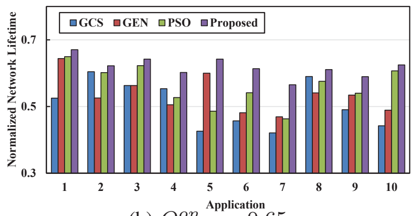


(c) $Q^{on}_{goal} = 0.75$

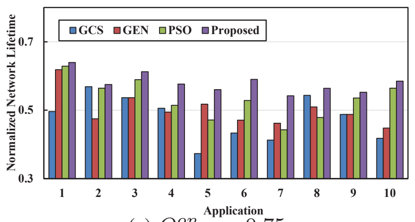
Fig. 12: The network lifetime when running 10 real-life applications under $N = 5$ and varying online QoS requirements.



(a) $Q^{on}_{goal} = 0.55$



(b) $Q^{on}_{goal} = 0.65$



(c) $Q^{on}_{goal} = 0.75$

Fig. 13: The network lifetime when running 10 real-life applications under $N = 10$ and varying online QoS requirements.

the figure clearly present that our proposed scheme achieves higher network lifetime compared to the three benchmarking algorithms. For example, in the case of $Q^{off}_{goal} = 0.5$, the network lifetime achieved by the proposed approach is 155.62%, 70.30%, 25.18% higher on average than that of RAN, CTF, and HWG, respectively. Fig. 11 plots the normalized network lifetime under varying offline QoS requirements when the network consists of 10 IoT devices executing real-life tasks. Similar to the results shown in Fig. 10, our proposed offline algorithm significantly improves the network lifetime.

2) Performance Comparison for Online Approach: Fig. 12 compares the normalized network lifetime under varying online QoS requirements when the network consists of 5 IoT devices executing real-life tasks. As demonstrated in the figure, our proposed online scheme achieves higher network lifetime compared to the three benchmarking algorithms. Take the case of $Q^{on}_{goal} = 0.55$ as an example, the network lifetime achieved by the proposed approach is 19.75%, 10.87%, and 6.94% higher on average than that of GCS, GEN, and PSO, respectively. Fig. 13 plots the normalized network lifetime under varying online QoS requirements when the network consists of 10 IoT devices executing real-life tasks. As shown in the figure, our proposed online algorithm achieves striking lifetime improvement compared to the three benchmarking algorithms. For example, in the scenario of $Q^{on}_{goal} = 0.55$, the network lifetime achieved by the proposed approach is 24.06%, 16.49%, and 11.13% higher on average than that of GCS, GEN, and PSO, respectively.

TABLE IV: The runtime (unit: s) on average for deriving the scheduling of 10 real-life applications under $N = 5$.

Q^{on}_{goal}	The number of IoT device: $N = 5$					
	Proposed	GEN	PSO	GCS	Min. Sp	Max. Sp
0.55	2.46	5.97	4.59	33.18	1.87x	13.51x
0.65	2.58	6.21	4.96	46.05	1.92x	17.85x
0.75	2.73	6.53	5.35	64.32	1.96x	23.59x

TABLE V: The runtime (unit: s) on average for deriving the scheduling of 10 real-life applications under $N = 10$.

Q^{on}_{goal}	The number of IoT device: $N = 10$					
	Proposed	GEN	PSO	GCS	Min. Sp	Max. Sp
0.55	12.86	29.71	24.43	271.22	1.90x	21.09x
0.65	15.30	32.41	25.85	367.15	1.68x	23.99x
0.75	17.62	34.72	27.65	466.43	1.56x	26.47x

TABLE IV lists the runtime on average for deriving the scheduling of real-life tasks when the network consists of 5 IoT devices. It can be easily seen from the table that our proposed online scheme greatly reduces the runtime for producing task scheduling. For example, when online QoS requirement Q^{on}_{goal} is set to 0.55, our proposed online scheme can achieve up to 13.51 times of speedup. TABLE V presents the runtime on average for deriving the scheduling of real-life tasks when the network consists of 10 IoT devices. Similar to the results presented in TABLE IV, our proposed online scheme also achieves better performance in terms of runtime compared to the three benchmarking schemes.

VII. Conclusion

In this paper, we tackle the problem of QoS-constrained network lifetime optimization for approximate computation real-time tasks in battery-powered IoT. Particularly, the mobility of IoT devices is taken into consideration to avoid redundant executions of the same task on different IoT devices for energy reduction. Our proposed solution consists of a mobility-aware offline scheme based on MILP technique and a QoS-adaptive online scheme based on cross-entropy method. Extensive simulations are conducted, and the experimental results reveal that our proposed solution achieves remarkable network lifetime improvement.

References

[1] P. Semasinghe, S. Maghsudi, and E. Hossain, "Game theoretic mechanisms for resource management in massive wireless IoT systems," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 121–127, 2017.

[2] J. Manyika, "The Internet of Things: Mapping the value beyond the hype," *McKinsey Global Institute*, 2015.

[3] J. Henkel, S. Pagani, H. Amrouch, L. Bauer, and F. Samie, "Ultra-low power and dependability for IoT devices (Invited paper for IoT technologies)," *In Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition*, pp. 954–959, 2017.

[4] G. Colistra, V. Pilloni, and L. Atzori, "Objects that agree on task frequency in the IoT: A lifetime-oriented consensus based approach," *IEEE World Forum on Internet of Things*, pp. 383–387, 2014.

[5] J. Luo, D. Wu, C. Pan, and J. Zha, "Optimal energy strategy for node selection and data relay in WSN-based IoT," *Mobile Networks and Applications*, vol. 20, no. 2, pp. 169–180, 2015.

[6] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," *IEEE World Forum on Internet of Things*, pp. 7–12, 2016.

[7] Q. Li, S. Gochhayat, M. Conti, and F. Liu, "Energiot: A solution to improve network lifetime of IoT devices," *Pervasive and Mobile Computing*, vol. 42, pp. 124–133, 2017.

[8] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys*, vol. 48, no. 4, 2016.

[9] J. Liu, K. Lin, W. Shih, A. Yu, J. Chung, and W. Zhao, "Algorithms for scheduling imprecise computations," *Springer US*, 1991.

[10] M. Amirijoo, J. Hansson, and S. Son, "Specification and management of QoS in real-time databases supporting imprecise computations," *IEEE Transactions on Computers*, vol. 55, no. 3, pp. 304–319, 2006.

[11] L. Cortes, P. Eles, and Z. Peng, "Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 10, pp. 1117–1129, 2006.

[12] H. Yu, B. Veeravalli, and Y. Ha, "Dynamic scheduling of imprecise-computation tasks in maximizing QoS under energy constraints for embedded systems," *In Proceedings of the IEEE Asia and South Pacific Design Automation Conference*, pp. 452–455, 2008.

[13] T. Wei, J. Zhou, K. Cao, P. Cong, M. Chen, G. Zhang, X. Hu, and J. Yan, "Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 9, pp. 1733–1746, 2018.

[14] K. Li, X. Tang, and K. Li, "Energy-efficient stochastic task scheduling on heterogeneous computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2867–2876, 2014.

[15] H. Huang, V. Chaturvedi, G. Quan, J. Fan, and M. Qiu, "Throughput maximization for periodic real-time systems under the maximal temperature constraint," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 2s, 2014.

[16] M. Haque, H. Aydin, and D. Zhu, "On reliability management of energy-aware real-time systems through task replication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, 2017.

[17] L. Deng, "The cross-entropy method: A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning," *New York: Springer-Verlag*, 2004.

[18] M. Qiu, K. G. Z. Ming, J. Li, and Z. Zong, "Phase-change memory optimization for green cloud with genetic algorithm," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3528–3540, 2015.

[19] J. Kennedy, "Particle swarm optimization," *Encyclopedia of Machine Learning, Springer US*, pp. 760–766, 2011.

[20] R. Iman, "Latin hypercube sampling," *John Wiley & Sons, Ltd*, 2008.

[21] S. Lucidl and M. Piccioni, "Random tunneling by means of acceptance-rejection sampling for global optimization," *Journal of Optimization Theory and Applications*, vol. 62, no. 2, pp. 255–277, 1989.

[22] H. Kooti, D. Dang, D. Mishra, and E. Bozorgzadeh, "Energy budget management for energy harvesting embedded systems," *In Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 320–329, 2012.

[23] S. Saha, Y. Lu, and J. Deogun, "Thermal-constrained energy-aware partitioning for heterogeneous multi-core multiprocessor real-time systems," *In Proceedings of the IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 41–50, 2012.

[24] J. Zhou, T. Wei, M. Chen, J. Yan, and X. Hu, "Thermal-aware task scheduling for energy minimization in heterogeneous real-time MPSoC systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.

[25] R. Jayaseelan and T. Mitra, "Temperature aware task sequencing and voltage scaling," *In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 618–623, 2008.

[26] K. Cao, J. Zhou, P. Cong, L. Li, T. Wei, M. Chen, S. Hu, and X. Hu, "Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[27] K. Cao, J. Zhou, M. Yin, T. Wei, and M. Chen, "Static thermal-aware task assignment and scheduling for makespan minimization in heterogeneous real-time MPSoCs," *In Proceedings of the IEEE International Symposium on System and Software Reliability*, pp. 111–118, 2016.

[28] Y. Cheng, L. Zhang, Y. Han, and X. Li, "Thermal-constrained task allocation for interconnect energy reduction in 3-d homogeneous MPSoCs,"

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 2, pp. 239–249, 2013.

[29] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, “Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms,” *Journal of Systems and Software*, vol. 133, pp. 1–16, 2017.

[30] Y. Wang, X. Lin, Q. Xie, N. Chang, and M. Pedram, “Minimizing state-of-health degradation in hybrid electrical energy storage systems with arbitrary source and load profiles,” *In Proceedings of the IEEE Design, Automation and Test in Europe Conference and Exhibition*, pp. 1–4, 2014.

[31] B. Martinez, M. Monton, I. Vilajosana, and J. Prades, “The power of models: Modeling power consumption for IoT devices,” *IEEE Sensors Journal*, vol. 15, no. 10, pp. 5777–5789, 2015.

[32] C. Deepu, C. Heng, and Y. Lian, “A hybrid data compression scheme for power reduction in wireless sensors for IoT,” *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, 2017.

[33] Y. Tirat-Gefen and D. Silva, “Incorporating imprecise computation into system-level design of application-specific heterogeneous multiprocessors,” *In Proceedings of the ACM Design Automation Conference*, pp. 58–63, 1997.

Kun Cao is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China.

His current research interests are in the areas of high performance computing, heterogeneous multiprocessor systems, and cyber physical systems. He received the Reviewer Award from Journal of Circuits, Systems, and Computers, in 2016.

Guo Xu is currently pursuing the master degree with the Department of Computer Science and Technology, East China Normal University, Shanghai, China. His current research interest is in the area of power management in mobile devices.

Junlong Zhou (S’15-M’17) received the Ph.D. degree in Computer Science from East China Normal University, Shanghai, China, in 2017. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014-2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include real-time embedded systems, cloud computing, and cyber physical systems. Dr. Zhou has been an Associate Editor for the Journal of Circuits, Systems, and Computers since 2017.

Tongquan Wei (M’11) received his Ph.D. degree in Electrical Engineering from Michigan Technological University in 2009. He is currently an Associate Professor in the Department of Computer Science and Technology at the East China Normal University. His current research interests include Internet of Things, edge computing, cloud computing, and design automation of intelligent and CPS systems. He serves as a Regional Editor for Journal of Circuits, Systems, and Computers since 2012. He is a member of the IEEE.

Mingsong Chen (S’08-M’11) received the B.S. and M.E. degrees from Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2003 and 2006 respectively, and the Ph.D. degree in Computer Engineering from the University of Florida, Gainesville, in 2010. He is currently a full Professor with the Department of Embedded Software and Systems of East China Normal University. His research interests are in the area of design automation of cyber-physical systems, formal verification techniques and mobile cloud computing. He is a member of the IEEE.

Shiyan Hu (SM’10) received his Ph.D. in Computer Engineering from Texas A&M University in 2008. He is an Associate Professor at Michigan Tech. where he is Director of Center for Cyber-Physical Systems and Associate Director of Institute of Computer and Cyber systems. He has been a Visiting Professor at IBM Research (Austin) in 2010, and a Visiting Associate Professor at Stanford University from 2015 to 2016. His research interests include Cyber-Physical Systems, Cybersecurity, Computer-Aided Design of VLSI Circuits, and Embedded Systems, where he has published more than 100 refereed papers. He is a fellow of IET.