Internet of Things, Ad Hoc and Sensor Networks Technical Committee Newsletter

(IoT-AHSN TCN)

Volume 1, No. 15

December 2021

CONTENTS

PREFACE

The IEEE ComSoc Ad Hoc and Sensor Networks Technical Committee (IoT-AHSN TC) sponsors papers, discussions, and standards on all aspects of IoT, ad hoc and sensor networks. It provides a forum for members to exchange ideas, techniques, and applications, and share experience among researchers. Its areas of interest include systems and algorithmic aspects of sensor and ad hoc networks, networking protocols and architecture, embedded systems, middle-ware and information management, novel applications, flow control and admission control algorithms, network security, reliability, and management. In an attempt to make all the TC members as well as the IoT-AHSN worldwide community aware of what is going on within our main areas of concerns, this newsletter had been set up. The newsletter aims at inviting the authors of successful research projects and experts from all around the world with large vision about IoT-AHSN-related research activities to share their experience and knowledge by contributing in short news.

The fifteenth issue of the IoT-AHSN TC Newsletter focuses on the theme “Cloud/Edge Computing for Internet of Things”. Specifically, this issue includes 2 news articles: i) Correlation Aware Scheduling for Mobile Edge Computing in Industrial Internet of Things; ii) Deep Reinforcement Learning Assisted Energy Efficient Computation Offloading with DVFS for Time-Critical IoT Applications in Edge Computing. We thank the contributors for their efforts to help make the IoT-AHSN TC Newsletter a success. We hope that the methods/approaches presented in this issue could significantly benefit researchers and application developers who are interested in IoT and ad hoc/sensor networks.

Newsletter Co-Editors
Qiang Ye (Dalhousie University, Canada)
Moez Esseghir (University of Technology of Troyes, France)
Lu Lv (Xidian University, China)

TC OFFICERS AND NEWSLETTER EDITORS

TC Officers

Name	Affiliation	Email
Jiajia Liu (Chair)	Xidian University, China	liujiajia@xidian.edu.cn
Sharief Oteafy (Vice Chair)	DePaul University, USA	soteafy@depaul.edu
Shuai Han (Secretary)	Harbin Institute of Technology, China	hanshuai@hit.edu.cn

Newsletter Editors

Name	Affiliation	Email
Qiang Ye (Editor in Chief)	Dalhousie University, Canada	qye@cs.dal.ca
Moez Esseghir (Technical Editor)	University of Technology of Troyes, France	moez.esseghir@utt.fr
Lu Lv (Technical Editor)	Xidian University, China	lulv@xidian.edu.cn

Correlation Aware Scheduling for Mobile Edge Computing in Industrial Internet of Things

Tongxin Zhu*, Zhipeng Cai†, Xiaolin Fang*, Junzhou Luo* and Ming Yang*
Southeast University*, Georgia State University†
{zhutongxin,xiaolin,jluo,yangming2002}@seu.edu.cn
, zcai@gsu.edu

***Abstract*—Mobile edge computing is a promising technology to support the computation-intensive and latency-sensitive applications in Industrial Internet of Things networks. One major problem for Mobile Edge Computing in Industrial Internet of Things (MEC-IIoT) networks is to schedule limited computation and communication resources with the purpose of improving computation efficiency. One feature of IIoT networks is that all IIoT devices monitor the same industrial site collaboratively. Therefore, the computation tasks originated from IIoT devices are correlated accordingly. To reduce the computation redundancy and further improve the computation efficiency in MEC-IIoT networks, a correlation aware scheduling algorithm is proposed in this paper, whose performance is verified by theoretical analysis and simulation results.**

***Index Terms*—Industrial Internet of Things, Mobile Edge Computing, correlation aware scheduling.**

I. Introduction

Industrial Internet of Things (IIoT) is widely applied in intelligent manufacturing, where IIoT devices in an industrial site are connected with each other through internet to monitor the industrial site collaboratively [1]. With the rapid development of IIoT, more and more computation-intensive and latency-sensitive tasks are required by IIoT devices. However, the computation, communication and storage capacities of IIoT devices are limited. To alleviate the conflict between limited capacity of IIoT devices and high requirements of computation tasks, Mobile Edge Computing (MEC) technique is employed in IIoT networks. With the help of edge devices located at the edge of IIoT networks, computation tasks originated from IIoT devices can be offloaded to near edge devices.

One major problem in MEC-IIoT networks is the scheduling for processing computation tasks. Plenty of works have been devoted to investigating the computation scheduling problem for MEC-IIoT networks [2]–[4]. However, these works ignored the correlations among computation tasks originated from IIoT devices in the same industrial site. It is well known that all IIoT devices in an industrial site sense the same industrial environment and their sensory data are highly correlated accordingly. Therefore, the computation of these sensory data are correlated as well, i.e., computation tasks in an industrial site are correlated. For example, in the industrial scenario where workers wearing Virtual Reality (VR) devices participating in industrial activities [5]. The main computation task of VR devices is the viewpoint rending. Workers in similar places may share a similar virtual environment, and the VR content of these VR devices may be highly correlated. Therefore, the computation tasks of VR devices in similar places are highly correlated. It inspires us to schedule highly correlated computation tasks to the same IIoT device or edge device. Therefore, correlation aware computation scheduling is necessary for reducing computation redundancy and improving computation efficiency in MEC-IIoT networks.

There are some challenges for correlation aware computation scheduling in MEC-IIoT networks. The first challenge is that whether a computation task being computed locally by its IIoT device or offloaded to an edge device is no longer independent. The computation offloading for computation tasks are related to their correlations. The second challenge is that different processing orders of computation tasks in an IIoT device or an edge device have different performances. It is necessary to make processing order decision for each device elaborately based on the correlations among their computation tasks. We solve the above challenges by proposing a correlation aware scheduling algorithm to approximately minimize computation latency in MEC-IIoT networks.

II. Problem Definition

We consider a MEC-IIoT network as shown in Fig.1(a). The set of edge devices is $\mathcal{E} = \{E_1, \cdots, E_M\}$ and the set of IIoT devices is $\mathcal{I} = \{I_1, \cdots, I_N\}$. The set of tasks is $\mathcal{T} = \{T_1, \cdots, T_K\}$ and the set of tasks originated from IIoT device I_i is $\mathcal{T}_i \subset \mathcal{T}$, where $\mathcal{T} = \bigcup_{i=1}^{N} \mathcal{T}_i$. Each task is represented by $T_k = (C_k, D_k)$, where D_k is the data size and C_k is the CPU cycles required by T_k. The definition of correlation coefficient of two tasks T_k and T_l is as follows.

Definition 1 (Correlation Coefficient): The correlation coefficient of two tasks $T_k, T_l \in \mathcal{T}$ is $\varepsilon_{k,l}$, indicating that if tasks T_k and T_l are successively computed in a single device, the number of CPU cycles required by the device computing them successively is $(C_k + C_l) \times (1 - \varepsilon_{k,l})$.

Based on definition 1, we have that $(C_k + C_l) \times (1 - \varepsilon_{k,l}) > C_k$ and $(C_k + C_l) \times (1 - \varepsilon_{k,l}) > C_l$. The reason is that the number of CPU cycles required by a device computing T_k and T_l must no less than the number of CPU cycles required by the device computing either T_k or T_l. Therefore, the correlation coefficient satisfies that $0 \leq \varepsilon_{k,l} < \frac{1}{2}$.

Each IIoT device can either compute one task locally or offload it to a selected edge device, i.e., choose local

Identify applicable funding agency here. If none, delete this.

computation model or edge computation model. Indicators $x_{k,j}$ $(j = 0, 1, \cdots, M)$ are used to indicate the schedule of task T_k, where $x_{k,0} = 0$ indicates T_k being computed by its original IIoT device and $x_{k,j}$ $(j = 1, \cdots, M)$ indicates T_k being offloaded to edge device E_j. We have that $\sum_{j=0}^{M} x_{k,j} = 1$.

Local Computation Model: The processing order of tasks computed at IIoT device I_i is denoted as a sorted set $\vec{\mathcal{N}}_i$, where $n_i = |\vec{\mathcal{N}}_i|$. Then, the CPU cycles required by I_i for processing the first n tasks in $\vec{\mathcal{N}}_i$ is

$$Z^I_{i,n} = \begin{cases} C_k, & n = 1, \\ Z^I_{i,n-1} + C_k - \varepsilon_{k,l}(C_k + C_l), & 2 \le n \le n_i. \end{cases} \quad (1)$$

Therefore, the latency for I_i processing all tasks in $\vec{\mathcal{N}}_i$ is $t^I_{i,n_i} = Z^I_{i,n_i}/f^I_i$, where f^I_i is the CPU frequency of I_i. The maximum computation latency for all IIoT devices processing tasks locally is $t^I_{sum} = \max\{t^I_{i,n_i} | \forall I_i \in \mathcal{I}\}$.

Edge Computation Model: The processing order of tasks computed at edge device E_j is denoted as a sorted set $\vec{\mathcal{M}}_j$, where $m_j = |\vec{\mathcal{M}}_j|$. Then, the CPU cycles required by E_j for processing the first n tasks in $\vec{\mathcal{M}}_j$ is

$$Z^E_{j,n} = \begin{cases} C_k, & n = 1, \\ Z^E_{j,n-1} + C_k - \varepsilon_{k,l}(C_k + C_l), & 2 \le n \le m_j. \end{cases} \quad (2)$$

Therefore, the latency for E_j computing all tasks in $\vec{\mathcal{M}}_j$ is $t^E_{j,m_j} = Z^E_{j,m_j}/f^E_j$, where f^E_j is the CPU frequency of E_j. Besides, the total latency for offloading all T_k originated from $I_i \in \mathcal{I}$ to E_j is $t^C_j = \sum_{i=1}^{N} \sum_{T_k \in \mathcal{T}_i} \frac{x_{k,j} D_k}{r_{i,j}}$, where $r_{i,j}$ is the uplink communication rate between I_i and E_j. The maximum latency for all edge devices receiving and computing offloaded tasks is $t^E_{sum} = \max\{t^C_j + t^E_{j,m_j} | \forall E_j \in \mathcal{E}\}$.

In summary, the latency for a MEC-IIoT network processing all tasks is $t_{sum} = \max\{t^E_{sum}, t^I_{sum}\}$. And the Correlation Aware Latency Minimization Scheduling (CALMS) problem in MEC-IIoT networks is

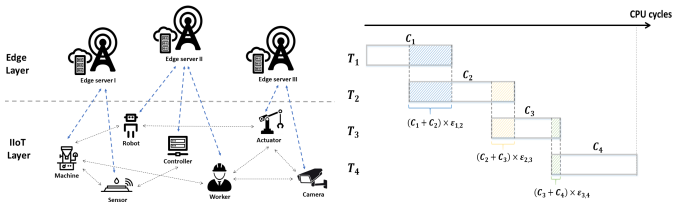
$$\begin{aligned} \textbf{LMS}: \quad & \min_{\boldsymbol{x}, \vec{\boldsymbol{\mathcal{N}}}, \vec{\boldsymbol{\mathcal{M}}}} \quad t_{sum} \\ & \text{s.t.} \quad \sum_{j=0}^{M} x_{k,j} = 1, \qquad T_k \in \mathcal{T}, \\ & \quad \sum_{T_k \in \mathcal{T}} x_{k,j} = |\vec{\mathcal{M}}_j|, \; E_j \in \mathcal{M}, \\ & \quad \sum_{T_k \in \mathcal{T}_i} x_{k,0} = |\vec{\mathcal{N}}_i|, \; I_i \in \mathcal{I}, \\ & \quad x_{k,j} = \{0, 1\}, \qquad T_k \in \mathcal{T}, 0 \le j \le M, \end{aligned} \quad (3)$$

where $\boldsymbol{x} = \{x_{k,j} | T_k \in \mathcal{T}, 0 \le j \le M\}$, $\vec{\boldsymbol{\mathcal{N}}} = \{\vec{\mathcal{N}}_i | I_i \in \mathcal{I}\}$, and $\vec{\boldsymbol{\mathcal{M}}} = \{\vec{\mathcal{M}}_j | E_j \in \mathcal{E}\}$.

Theorem 1: The Correlation Aware Latency Minimization Scheduling problem in MEC-IIoT networks is NP-hard.

III. Skeleton of the CAS Algorithm

We proposed an approximation algorithm for the Correlation Aware Latency Minimization Scheduling problem in MEC-IIoT networks, named as the Correlation Aware Scheduling (CAS) algorithm. The CAS algorithm consists of two parts, i.e., Computation Model Decision and Processing Order Decision.



(a) The network infrastructure (b) The correlation coefficients

Fig. 1. The illustration of a MEC-IIoT network.

Computation Model Decision: We first make the computation model decision $\boldsymbol{x}$ for all tasks without considering their correlations. In this case, the problem can be modified to a classic makespan minimization scheduling problem for unrelated parallel machines. According to this modification and an existing algorithm for the classic makespan minimization scheduling problem [6] [7], we propose the computation model decision policy with approximation ratio of 2.

Processing Order Decision: We make the processing order decision $\vec{\boldsymbol{\mathcal{N}}}$ and $\vec{\boldsymbol{\mathcal{M}}}$ for tasks in all IIoT devices and edge devices according to correlations among tasks. A task queue for a device is maintained in a greedy strategy as follows. The task queue is initialized as two tasks with the maximum correlation coefficient. Then, the task having maximum correlation coefficient with the head or tail of the task queue is added to the head or tail ot the task queue. In this way, the computation redundancy is reduced as far as possible.

Theorem 2: The approximation ratio of the CAS algorithm is $\frac{2}{(1-2\varepsilon_{max})}$, where $\varepsilon_{max} = \max\{\varepsilon_{k,l} | T_k, T_l \in \mathcal{T}\}$.

Theorem 2 is proved to verify the performance of the CAS algorithm. And simulation results show that the proposed CAS algorithm can significantly reduce latency of MEC-IIoT networks by considering correlations among tasks.

References

[1] G. S. S. Chalapathi, V. Chamola, A. Vaish, and R. Buyya, "Industrial internet of things (iiot) applications of edge and fog computing: A review and future directions," *CoRR*, vol. abs/1912.00595, 2019.

[2] M. Li, C. Chen, H. Wu, X. Guan, and X. Shen, "Age-of-information aware scheduling for edge-assisted industrial wireless networks," *IEEE Trans. Ind. Informatics*, vol. 17, no. 8, pp. 5562–5571, 2021.

[3] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge," *IEEE Trans. Ind. Informatics*, vol. 16, no. 4, pp. 2737–2746, 2020.

[4] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A novel framework of three-hierarchical offloading optimization for MEC in industrial iot networks," *IEEE Trans. Ind. Informatics*, vol. 16, no. 8, pp. 5424–5434, 2020.

[5] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. C. M. Leung, "Resource management for pervasive-edge-computing-assisted wireless VR streaming in industrial internet of things," *IEEE Trans. Ind. Informatics*, vol. 17, no. 11, pp. 7607–7617, 2021.

[6] V. V. Vazirani, *Approximation algorithms*. Springer, 2001.

[7] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," in *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*. IEEE Computer Society, 1987, pp. 217–224.

Deep Reinforcement Learning Assisted Energy Efficient Computation Offloading with DVFS for Time-Critical IoT Applications in Edge Computing

Saroj Kumar Panda
Department of Computer Science
St. Francis Xavier University
Canada
x2019fpn@stfx.ca

Man Lin
Department of Computer Science
St. Francis Xavier University
Canada
mlin@stfx.ca

***Abstract*—As more and more battery-powered IoT devices are deployed in the field to support the growing number of IoT applications requiring real-time response, reducing the energy consumption of these IoT devices while meeting the computational goals have become the most important challenge. In this article, we propose a deep reinforcement learning and Dynamic Voltage and Frequency Scaling (DVFS) based application-deadline-aware data offloading scheme in an edge computing environment to reduce the energy consumption of IoT devices. This scheme learns the DVFS frequency scaling for local computation and the optimal data distribution policies by interacting with the system environment and learning the device, network, and edge server's behaviours. Experimental results using various devices show that the proposed scheme can achieve the IoT application's timing and computational goals while minimizing energy consumption. The proposed scheme also achieves substantial energy gain compared to the native Linux governors.**

***Index Terms*—IoT, Edge Computing (EC), DVFS, Deep Reinforcement Learning (DRL).**

I. Introduction

Edge Computing (EC) in IoT applications perform the computation close to the data source by placing high computing power edge servers near the IoT devices. IoT devices manufactured in recent years have computational power and memory, allowing EC to perform IoT application computation directly on the IoT device. However, compared with edge servers, IoT devices have limited computation capability and battery-power, making it infeasible to achieve delay-sensitive IoT applications' computational goals by computing only on the IoT device. Computation Offloading is one of the best solutions to this problem, which can transfer some of the computation tasks and data to nearby edge servers.

The core challenge of computation offloading is to jointly optimize energy consumption, computation, communication, and make a controlled computation offloading decision in a dynamic EC environment. Various dynamically changing factors can affect the offloading decision, such as communication medium, and edge server utilization states. If the offloading decision does not consider these dynamic variations, it will perform poorly.

Deep reinforcement learning (DRL) is a promising solution for optimization problems [1]–[3] including computation offloading in dynamic environments. IoT device learns the best offloading policy by interacting with the environment in a trial-and-error manner. Dynamic Voltage and Frequency Scaling is another promising solution to optimize processor energy consumption where the processor frequency and voltage are adjusted to lower levels to reduce energy consumption.

In recent years, Deep Learning (DL) based IoT applications have become popular. Training the DL models requires high computational power and can be trained offline on cloud or edge servers using high computational resources, whereas performing inference on the trained model requires significantly less computational power and can be performed on IoT devices. These DL models are highly data-parallel, i.e. multiple instances of the same DL model deployed on multiple edge devices can perform inference on different data sets in parallel. We take advantage of this data parallelism and propose a deep reinforcement learning (DRL) based simpler computation offload approach that learns to jointly optimize energy consumption by selecting optimal DVFS frequency for the local computation, and the data offload distribution to offload part of the data from the IoT device to the edge servers while considering the dynamically varying factors of communication network state and edge servers resource utilization state. Both the IoT device and the edge server execute the same computation task on separate data sets.

II. Proposed DRL Offloading Scheme and Experiment Results

The System Architecture of the proposed DRL Offloading Scheme is schematically illustrated in Fig. 1. IoT device energy consumption varies for different application deadlines. The edge server computation results availability and energy consumption varies based on the communication medium and the edge server's computational resources availability states. We capture the impact of both by computing the edge server response time per data. We use a combination of the response time and the application deadline as the environment state.

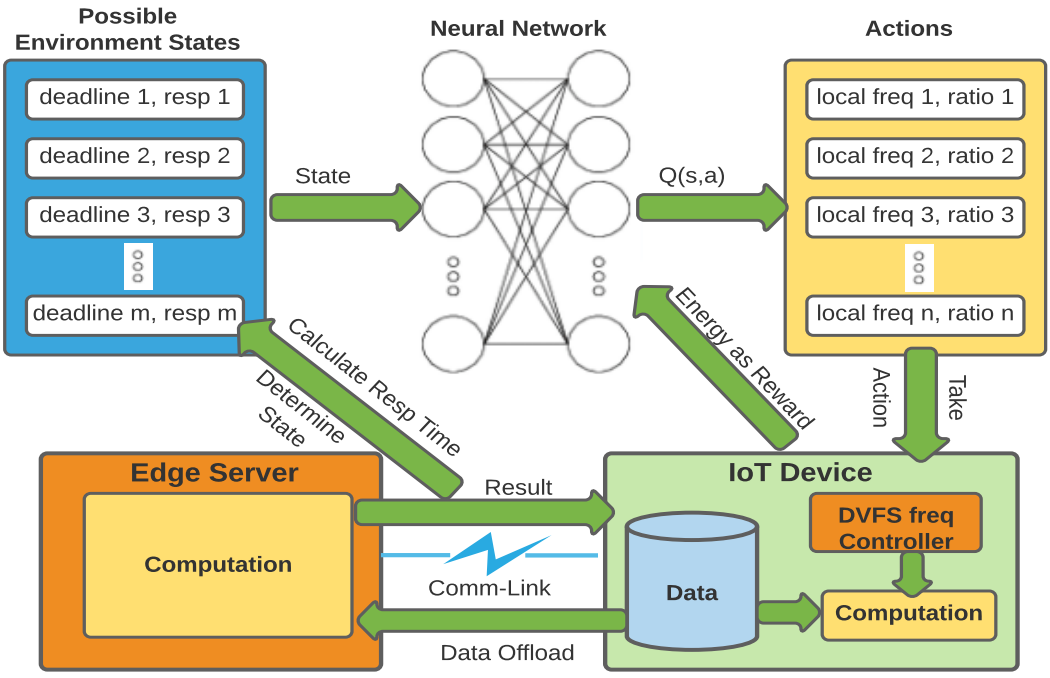


Fig. 1. System Architecture of DRL Offloading Scheme

We use the local computation CPU frequency, and the data offload distribution among IoT device and edge servers as the actions.For simplicity of implementation, we use the same set of actions for all the states.

We use energy consumption as the reward if the action is completed within the application deadline and a penalty of a high number if the action misses the deadline. The reinforcement learning agent initially sends some data to the edge server to determine the initial system state. After the initial state is determined, it takes random action, observes the reward or penalty, and determines the new state by observing the response time for the action taken. It repeats the steps of determining the state, taking action in that state, observing the reward/penalty, and trains a neural network from the data collected to estimate the action values *Q(s,a)* by feeding the state as the input and reward/penalty as the expected action values for the set of actions defined for that state.

A. Experiment Results

We used one IoT device and one edge server with MNIST classification as the IoT application. We conducted experiment using Raspberry Pi [4], Jetson Nano [5], and a Linux laptop [6] as the IoT device. The trained DRL model was able to predict the right action for the varying system states that minimized the energy consumption when compared with rest of the actions. The proposed scheme also saved *3 - 7%* energy against *ondemand* and *7 - 10%* against *conservative* governor when Jetson Nano [5] and Linux Laptop [6] were used as IoT device. Raspberry Pi [4] had negligible energy savings as it does not support voltage scaling.

III. Conclusion

This article proposes a DRL based offloading scheme that learns the optimal data distribution and local computation DVFS frequency scaling by interacting with the system environment and learning the device, the network, and edge servers' behavior. Experiment results from various IoT devices show that this scheme always selects the best action that achieves the application deadline while minimizing the energy consumption. This scheme also saves energy in comparison to the native DVFS scaling Linux governors.

References

[1] H. Huang, Q. Ye, and H. Du, "Reinforcement learning based offloading for realtime applications in mobile edge computing," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[2] T. Zhou and M. Lin, "Deadline-aware deep-recurrent-q-network governor for smart energy saving," *IEEE Transactions on Network Science and Engineering*, 2021.

[3] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[4] R. P. Foundation, "Raspberry pi 4 model b specifications," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[5] N. Corporation, "Jetson nano 2gb developer kit," https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit.

[6] Intel, "Intel® core™ i3-6006u processor specification," https://ark.intel.com/content/www/us/en/ark/products/91157/intel-core-i3-6006u-processor-3m-cache-2-00-ghz.html.