Computational Task Offloading in Mobile Edge Computing using Learning Automata

Zahir Abbas, Jun Li, Nagendra Yadav, Irfan Tariq
School of Electronic and Optical Engineering,
Nanjing University of Science and Technology, Nanjing, 210094, P. R. CHINA
Email: {zahir.abbas,jun.li,irfantariq}@njust.edu.cn
nagendra1nagendra@gmail.com

***Abstract*—The widespread diffusion of mobile devices is triggering an exponential growth of mobile data causing an offloading problem. In this paper, a model has been proposed to investigate computing task offloading problem of small base stations (SBSs). We formulate an optimization problem to minimize utility function for energy and execution time consumptions, where mobile devices (MDs) can select local computation or offloading to SBSs. We use discrete generalist pursuit algorithm (DGPA) based on learning automata (LA) to optimize the computation allocation. First, we examine a faster-converging of discrete generalist pursuit algorithm (DGPA) for LA is based on the concept of conditional inaction discrete generalist pursuit algorithm (CI-DGPA). Then, we design a reward function for the selection of optimal allocation of SBSs with minimum offloading latency. After convergence to optimal allocation, the offloading computing task of MDs has been performed to SBSs with minimum weighted sum of time and energy consumption. Finally, numerical results demonstrate the best improvement of DGPA algorithm.**

I. INTRODUCTION

The introduction of smart mobile devices in recent years has led to the wide usage of mobile applications such as video calling, online gaming, speech recognition, industrial application and social media applications. It has become challenging for mobile devices to fulfill user demands due to their limited computational and battery capacity. However, with the introduction of MEC to traditional cloud computing (CC), there has been an improvement in system performance as well as a reduction in energy consumption [1–3].

Due to the long distance transmission in wide area networks, there is an increase in latency of CC system. However, MEC which provides the computation, storage and networking services for mobile devices at the edge of the network, reduces the network latency in such scenarios. By offloading computation task to proximate MEC server, the efficient computation task will be improved greatly [4].

MEC has many advantages e.g., energy saving, fast offloading, low latency, and optimal resource allocation, so it is a significant candidate for the future generation of wireless communication. The recent research is based on efficient offloading and resource allocation schemes for MEC systems. The resource allocation for partial offloading schemes frequently used in recent research based on time division multiple access (TDMA) and frequency division multiple access (FDMA). We need to focus on best approximation and user association algorithm for mobile devices in small cell network to optimize our objectives [5, 6]. Similarly, we can improve the performance of system to manage allocation of resource for computational offloading [7, 8]. When the offloading task can be executed closer to concern mobile users, then user can directly get computational offloading results from the MEC servers of SBSs rather than the macro base stations.

We propose a distributive model for offloading the computing task to the MEC servers. MEC system consists of multiple MDs and SBSs. Mobile devices can link with one or more SBSs. MEC servers of SBSs with some finite computing abilities links with multiple MDs. Mobile devices access to small base stations for offloading data on the basis of highest signal to noise ratio (SNR) and the reward function. The most challenging part of this work is the design of reward function for the action taken by LA. DGPA converge to the optimal action is based on best rewarded function and positive environment feedback. We develop a framework for a game of independent LA on the basis of (CI-DGPA) to select the optimum allocation [9–11]. The information exchange between mobile devices and small base stations with best adjacent neighbors based on marginal distribution function or probability mass function (PMF) as mentioned in [12]. Simulation results demonstrate that the proposed decentralized distributed algorithm can notably reduce the weighted sum of time and energy consumption.

II. SYSTEM MODEL

We develop the MEC task offloading system model in which I mobile devices and J small base stations. The mobile devices denoted $m = \{M_1, M_2, M_3, ..., M_I\}$ and the set of SBSs $b = \{B_1, B_2, B_3, ..., B_J\}$. Let T_i be the bit data needed to be executed and D_i be the computational complexities of all mobile devices. The fractional computing amount of bits data to the total data bits of M_i mobile device denoted by $\mu_i^j \big(0 \leqslant \mu_i^j \leqslant 1\big)$. It should be considered that the computational data size of SBSs cannot exceed the maximum ability denoted by T_j. The computing speed of MEC servers denoted by f_j^S . We assume that all executed data of mobile devices should be serial or the data cannot be executed parallel.

A. Local Computing Model

The speed of computation capabilities of the MDs is denoted by s_i^I . The energy and power consumption of mobile device

will be as $e = k\big(s_i^I\big)^2$ and $P = k\big(s_i^I\big)^3$ where, k is the coefficient of MDs chip. The local computation execution time and energy consumption given;

$$\tau_i^I = \frac{\Big(1 - \sum_{j=1}^{J} \mu_i^j\Big) T_i D_i}{s_i^I} \tag{1}$$

$$E_i^I = \Big(1 - \sum_{j=1}^{J} \mu_i^j\Big) T_i D_i k\big(s_i^I\big)^2 \tag{2}$$

where equation (1) represents the local executing time for mobile devices and equation (2) is for energy consumption in local computing.

B. Computational Offloading Model

We consider that the system has sufficient number of channels available for all the MDs. Let the channel gain between mobile devices and SBSs be $g_{i,j}$ and $R_{i,j}$ be the offloading rate from MDs to SBSs. N_O And W denotes the noise power and channel bandwidth. So the uplink rate as:

$$R_{i,j} = Wlog\Big(1 + \frac{P_i g_{i,j}}{N_o}\Big) \tag{3}$$

The uploading time for mobile devices can be determined as:

$$\tau_i^{off} = \sum_{j=1}^{J} \frac{\mu_i^j T_i}{R_{i,j}} \tag{4}$$

The offloading computing time of MDs at the MEC server of the SBSs can be written as:

$$\tau_i^C = \sum_{j=1}^{J} \frac{\mu_i^j T_i D_i}{f_i^S} \tag{5}$$

Therefore, the total offloading computing execution time and energy consumption while offloading data by MDs can be written as:

$$\tau_i^S = \sum_{j=1}^{J} \Big(\frac{\mu_i^j T_i}{R_{i,j}} + \frac{\mu_i^j T_i D_i}{f_j^S}\Big) \tag{6}$$

$$E_i^S = \sum_{j=1}^{J} \frac{\mu_i^j T_i P_i}{R_{i,j}} \tag{7}$$

C. Problem Formulation

The resource allocation for multiple mobile device computing offloading is formulated as an optimization problem. Our aim is to minimize weighted sum of mobile energy and time consumptions: $\sum_{i=1}^{I} \alpha E_i^T + \beta \tau_i^T$, where α and β are the weighted parameters of energy and time. The expanded form of objective function is given as:

$$\begin{aligned} &\min_{\mu} \sum_{i=1}^{I} \alpha \Bigg(\Big(1 - \sum_{j=1}^{J} \mu_i^j\Big) T_i D_i k(s_i^I)^2 + \sum_{j=1}^{J} \frac{\mu_i^j T_i}{R_{i.j}} \Bigg) \\ &+ \beta \Bigg(\frac{\Big(1 - \sum_{j=1}^{J} \mu_i^j\Big) T_i D_i}{s_i^I} + \sum_{j=1}^{J} \Big(\frac{\mu_i^j T_i}{R_{i.j}} \\ &+ \frac{\mu_i^j T_i D_i}{f_i^S} \Big) \Bigg) \end{aligned} \tag{8}$$

$$S.T \rightarrow \Big\{ \sum_{i=1}^{I} T_i D_i \mu_i^j \leq T_j \quad \text{(8a)}$$

The constraint (8a) denotes that a mobile device can only associate with one SBS and the size offloading task should not exceed the size of SBS. Transforming above object function, we can get maximum local utility function as:

$$\max_{\mu} \sum_{i=1}^{I} -\alpha E_i^T - \beta \tau_i^T \tag{9}$$

D. Computational Offloading Arrangements

We denote the vector $\mu_j = \{\mu_1^j, \mu_2^j, \mu_3^j, ..., \mu_I^j\}$ all mobile devices the ratio of offloading computing amount of bits out of total bits to SBSs B_j. We decompose the objective function into J local utility functions for SBSs as $B_J = \{B_1, B_2, B_3, ...B_J\}$,where

$$\begin{aligned} B_j = \sum_{i=1}^{I} -\alpha \Big(&\frac{T_i D_i k\big(s_i^I\big)^2}{J} - \mu_i^j T_i D_i k(s_i^m)^2 \\ &+ \frac{\mu_i^j T_i P_i}{R_{i,j}} \Big) - \beta \Big(\frac{T_i D_I}{s_i^I J} - \frac{\mu_i^j T_i D_i}{s_i^I} + \frac{\mu_i^j T_i}{R_{i,j}} \\ &+ \frac{\mu_i^j T_i D_i}{f_i^S} \Big). \end{aligned} \tag{10}$$

III. LEARNING AUTOMATA (LA)

LA helps MDs to select best optimal action for data offloading on the basis of reward function. Here we focus on discrete generalized pursuit algorithm (DGPA) which is faster convergence most accurate algorithm of the learning automaton [8–10]. LA generalizes concepts of the pursuit algorithm by "pursuing" all actions that have higher reward estimated than the current chosen action.

A. Discrete Generalized Pursuit Algorithm (DGPA)

The goal of LA is to determine an optimal action out of all available actions. The MDs link with the most rewarded actions of SBSs to minimize the weighted sum of time and energy consumption. Let the actions in the searching space be $A = \{a_1, a_2, a_3, ...a_s\}$, where s is length of search space. Let $\sigma = [0, 1]$ be the environment feedback and if $\sigma = 1$ feedback is positive, otherwise it become negative. The probability vector of actions in searching space are denoted by $P(t) = \{p_1(t), p_2(t), p_3(t), ...p_s(t)\}$, where $p_i(t)$ is the probability that automaton select a_i action in time slot t . The sum all probability of actions $\sum_{i=1}^{S} p_i(t) = 1$. The reward estimator vector is denoted by $d(t) = \{d_1(t), d_2(t), d_3(t), ..., d_s(t)\}$. The number of actions have higher reward estimate then the current action are consider by $E(t)$. By using DGPA, the probabilities of all actions with higher reward estimates $d_j(t)$ then the current action $d_i(t)$ increment by factor $\frac{\Delta}{E(t)}$, where $\Delta = \frac{1}{r.N}$ is the smallest step size, r is number of total actions in search space and N is resolution parameter. The probability vector $P(t)$ uses the direction vector $v(t)$ to update the probability for next action and it becomes one when higher reward estimates vector $d_j(t)$ greater than the current chosen

action $d_i(t)$, otherwise it becomes zero. The update probability vector for next action can be obtained by following equation:

$$P(t+1) = P(t) + \frac{\Delta}{E(t)}v(t) - \frac{\Delta}{r - E(t)}[U - v(t)], \quad (11)$$

where U is the unit vector and $v(t)$ the direction vectors for both higher reward estimate action $d_j(t)$ and current chosen action $d_i(t)$ can be define in the following equations:

$$v_j(t) = \begin{cases} 1, & \text{if } d_j(t) > \ d_i(t); \\ 0. & \text{if } d_j(t) <= \ d_i(t). \end{cases} \quad (12)$$

Similarly, we can also define the direction vector $v(t)$ for current action as:

$$v_i(t) = \begin{cases} 1, & \text{if } d_i(t) = \max d_j(t); \\ 0. & \text{if } d_i(t) < \max d_i(t). \end{cases} \quad (13)$$

In equation (11), Δ is the step size of the algorithm which shows the trade-off between convergence and accuracy. The action probability updates for the next action in (11) on the basis of reward probability estimate $d(t)$. An action may be rewarded if the mass probability $p_i(t)$ is very small or approaches to zero causes number of penalties. It consumes more time to avoid this delay, we use the concept of CI-DGPA.

B. Conditional Inaction DGPA (CI-DGPA)

We need to update the direction vector $v(t)$ on the basis of probabilities. If the probability of any action is greater than zero then the direction vector $v(t)$ will increases otherwise it decreases. An action may be rewarded even its probability $p_i(t)$ approaches to zero then multiple penalties being assigned it. To minimize the convergence time, we introduce the concept of conditional inaction. The probability updates policy increases the direction vector $v(t)$ only when the probability $p_i(t)$ is not equal to zero, otherwise it decreases the direction vector as:

$$v_j^I(t) = \begin{cases} 1, & \text{if } d_j(t) > \ d_i(t)\text{U } p_j(t) \neq 0; \\ 0, & \text{if } d_j(t) <= \ d_i(t)\text{U } p_j(t) \neq 0; \\ 0, & \text{if } p_j(t) = 0. \end{cases} \quad (14)$$

$$v_i^I(t) = \begin{cases} 1, & \text{if } d_i(t) = \max d(t); \\ 0. & \text{otherwise.} \end{cases} \quad (15)$$

We can also define the decrement of direction vector $v(t)$ as:

$$v_j^D(t) = \begin{cases} 0, & \text{if } d_j(t) > \ d_i(t)\text{U } p_j(t) \neq 0; \\ 1, & \text{if } d_j(t) <= \ d_i(t)\text{U } p_j(t) \neq 0; \\ 0, & \text{if } p_j(t) = 0. \end{cases} \quad (16)$$

$$v_i^D(t) = \begin{cases} 1, & \text{if } d_i(t) \neq maxd(t); \\ 0. & \text{otherwise.} \end{cases} \quad (17)$$

Now we can easily update probability vector for the next action based on direction vector as;

$$P(t+1) = P(t) + \frac{\Delta}{E(t)}v^I - \frac{\Delta}{r - E(t)}v^D \quad (18)$$

IV. DISTRIBUTED COMPUTATIONAL OFFLOADING BASED ON LA

We propose a decentralized approach to solve offloading problem through independent Learning Automata (LA). The set of independent learners of LA are denoted by k, C is the environment in which LA learns from, σ is a binary vector for reward function or feedback. Initially the fractional offloading computing task μ_i^j of mobile devices represents in the form of Zip-f distribution with discount rate γ. We denote the latency w_i^j that MDs experience while offloading data μ_i^j to the SBSs B_j. With the selection of the optimal actions, MDs reduces the offloading latency w_i^j equivalent to maximize the uplink rate as:

$$D_{off} = \sum_{j=1}^{J} \mu_i^j w_i^j \quad (19)$$

and the average network offloading latency be;

$$D_{ad} = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\mu_i^j w_i^j}{N} \quad (20)$$

where N is the number of MDs. The role of DGPA is to minimize average offloading latency by selection of best actions as:

$$D_d = \arg \ min(D_{ad}). \quad (21)$$

Mobile devices estimate the offloading latency w_i^j with the connected SBSs. LA process occurs in three consecutive time slot, first LA select the best allocations and broadcast their choices to the MDs, then MDs check the SNR of selected actions of SBSs and send feedbacks to the LA. Finally, LA check the environment feedback $\sigma = [0, 1]$ for taking action. In LA process, the feedback of MDs send to the learner of LA be r_{ik}, where more popular data amount for higher share and feedback should be distance dependent, so that more weight is given to feedbacks from the closer MDs. We design mobile devices feedbacks as:

$$r_{ik} = +\frac{u_i^j}{w_i^j}; \quad (22)$$

$$r_{ik} = -\frac{u_i^j}{w_i^j}. \quad (23)$$

Equation (22) denotes the rewards for positive feedback and equation (23) denotes the penalties. The accumulative MDs feedbacks as:

$$C_{ik} = \sum_{i=1}^{I} r_{ik}. \quad (24)$$

An action is rewarded if and only if $C_{ik} > 0$. We can minimize the offloading weighted delay using equation (21) to minimize offloading weighted delay of each mobile device so, we can minimize the weighted sum of time and energy consumptions. The environment feedback $\sigma(t)$ is obtained as follows:

$$\sigma(t) = \begin{cases} 1, & \text{if } C_{ik} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (25)$$

The estimated reward probability $d_i(t)$ is set to zero initially, referred as a game initialization. At every single process, $d_i(t)$ updates with maximum likelihood estimation (MLE). The automata keeps two counters for each mobile device to select the best action for task offloading. One counter counts the number of times, when an action is being selected for offloading denoted by $Z_i(t)$ and the second counter counts the number of time, when an action has been rewarded denoted by $W_i(t)$. The reward function is the ratio of most rewarded action to the number of time the action being selected $d_i(t) = W_i(t)/Z_i(t)$. If the feedback from the environment is positive then updating equation of estimate reward $d_i(t)$ for the best chosen action as:

$$Z_i(t+1) = Z_i(t) + 1, \tag{26}$$

$$W_i(t+1) = W_i(t) + \sigma(t), \tag{27}$$

$$d_i(t+1) = \frac{W_i(t+1)}{Z_i(t+1)}. \tag{28}$$

if $\sigma = 1$ the feedback is positive and action is rewarded one. At the same time, $P(t)$ and $d(t)$ are updated according to (18) and (28). The game initialization will be successful when $d_i(t) \neq 0$.

Once LA maximize the uplink rate or minimize uplink latency by selection of an action on the basis of reward estimate function, then the information exchange between mobile devices and SBSs through marginal distribution. The probability mass function (PMF) of the variables μ_i of MDs can be written as:

$$p(\mu) = \frac{1}{G}\exp(qB_j) \tag{29}$$

where q is any positive number and G is normalize parameter. The message update between MDs μ_i and SBSs B_j, we can obtain the probability mass function for μ_i as:

$$p(\mu_i^j) = \{p(\mu_i^{[j,1]}), p(\mu_i^{[j,2]}), p(\mu_i^{[j,3]}), ...p(\mu_i^{[j,Y]})\} \tag{30}$$

where y^{th} probability estimated of the SBSs B_j . $\mu_i^{j,y}$ represents the yth valve of μ_i^j . We denote variable node of mobile devices m and factor node of SBS b . The message exchange between mobile devices and SBSs based on marginal distribution are described in following steps:

1) Initialization: At each mobile variable m, set message as $p_{m\to b}^{t=1}(\mu_m^b)$, where $t = 1$ iteration index of initial distribution of μ_m^b . We set the initial PMF of all mobile variables as a uniform distribution.

2) MDs update: At the t^{th} iteration, the mobile device m updates the message $p_{m\to b}^t(\mu_i^j)$ for the b is based on the messages obtained from all neighboring SBSs of m mobile devices other than b as:

$$mg_{m\to b}^t(\mu_m^b) = \frac{1}{G_i^t}\prod_{b^-\in H(m)} p_{b^-\to m}^{t-1}(\mu_m^b) \tag{31}$$

$H(m)$ denote the neighboring SBSs of mobile devices m.

3) SBSs update: At the t^{th} iteration, the SBSs b updates the message $p_{b\to m}^t(\mu_m^b)$ for the mobile devices m is based on

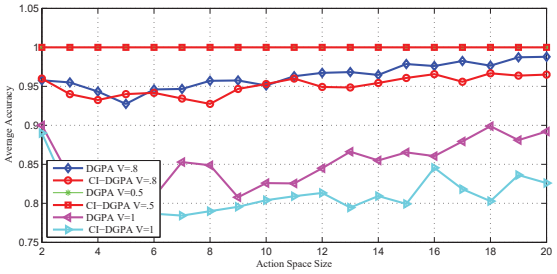

Fig. 1: The accuracy of DGPA and CI-DGPA over the normal environment variance V.

messages collected from all neighboring MDs of b SBSs other than m on basis given below equation:

$$mg_{b\to m}^t(t, \mu_m^b) = \sum(qb(H(t, \mu_m^b)) \prod_{m^-\in H(b)} p_{m^-\to b}^{t-1}(t, \mu_m^b)) \tag{32}$$

where $H(b)$ denote the neignboring MDs of the SBSs b.

4) Final decision: when the total T iteration finished MDs choose the specific offloading decision to SBSs b having maximum probability function by the following formula:

$$p_i(t+1, \mu_i^{j[y]}) = \frac{1}{G^T}\prod_{b\in H(m)} p_m^b(t, \mu_i^{j[y]}) \tag{33}$$

based on above formula, the MDs variable μ_m could choose specific offloading decision to SBSs B_b.

V. PERFORMANCE ANALYSIS

A. Accuracy and Convergence

As a_o be the optimal action and action taken by LA is a_t, if the expected value of action taken by LA is equal to the optimal action then action taken by LA is optimal one. As shown in figure 1, the accuracy is comparable with $V < 1$ a small gap can be observed for $V = 1$ and $V = .8$ with minimum accuracy. When $V = .5$ or less the both algorithms get maximum accuracy that is 100%. In figure 2, the average convergence time of CI-DGPA and DGPA increase linearly with the search space size. When the $V = 1$ both algorithms show large convergence time, $V = .8$ show that convergence time is reduce and at $V = .5$ CI-DGPA minimize the convergence time. Therefore, DGPA select optimal actions for offloading computing task to small base stations from the mobile devices.

B. Numerical Results

In figure 3, the fractional computing task of MDs were offloaded to most rewarded allocation of SBSs, where the network offloading delay minimize with the increase number of discount rate γ, where $0 <= \gamma <= 1$ of Zip-f distribution. Our proposed algorithm minimize the offloading weighted delay or equivalent to maximize offloading rate as compare to greedy algorithm and random access.

Figure 4 show the average weighted sum time and energy consumptions with the increase number of CPU maximum time cycles. First, we compute the possible optimal values

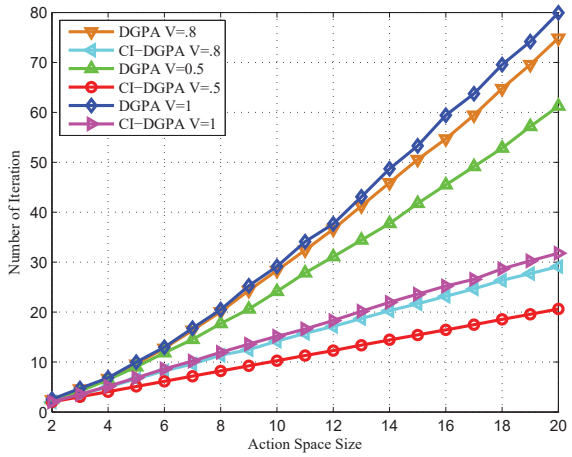

Fig. 2: The convergence of DGPA and CI-DGPA over the normal environment variance V.

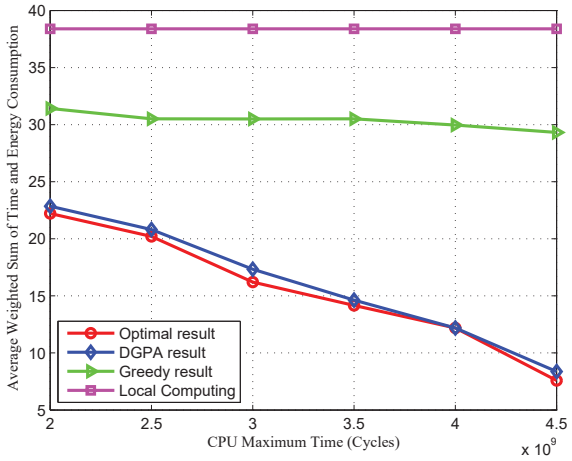

Fig. 4: The average weighted sum of time and energy consumptions while offloading computing from MDs to the SBSs with CPU maximum time cycles.

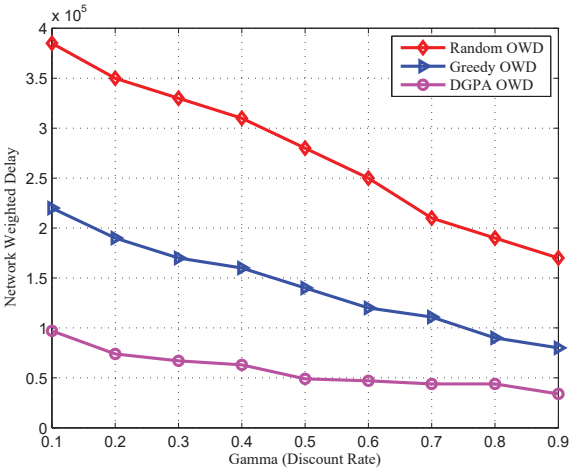

Fig. 3: The average offloading weighted delay while offloading computing data from MDs to the SBSs with increase discount rates γ.

of average weighted sum of time and energy consumption in all SBSs. Then, we compute the values of average weighted sum of time energy consumption for local computing, greedy algorithm and DGPA. The DGPA approaches to optimal values as compare to greedy and local computing with the increase number of CPU maximum time cycles. At CPU maximum time cycle $4 * 10^9$ the DGPA touch to optimal value shows the significance our work.

VI. CONCLUSION

In this paper, a model of distributed algorithm has been developed to solve the computational offloading problem in MEC based on the Learning Automata (LA). Due to the convergence speed and higher learning accuracy, we have proposed a faster converging DGPA algorithm for Learning Automata based on the concept of condition inaction (CI-DGPA). We designed reward functions to select optimal allocations for offloading computing data with minimum offloading weighted latency while offloading computing task from MDs to the MEC servers of SBSs. The average weighted sum of time and energy consumption have shown that the proposed distributed DGPA algorithm approaches to the optimal result as maintaining low computation complexity.

REFERENCES

[1] P. Zhao, H. Tian, C. Qin, and G. Nie, Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing, IEEE Access., vol. 5, pp. 11 255-11 268, Jun. 2017.
[2] X. Chen, L. Jiao, W. Li, and X. Fu, Efficient multi-user computation offloading for mobile edge cloud computing, IEEE/ACM Transactions on Networking, vol.24, no.5, pp.2795-2808, oct.2016
[3] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta and A. Neal, Mobile-edge computing, Mobile-Edge Computing-Introductory Technical White Paper, Sep. 2014
[4] Y. Chen, J. Li, W. Chen, Z. Lin, and B. Vucetic, Joint user association and resource allocation in the downlink of heterogeneous networks, IEEE Transactions on Vehicular Technology, vol. 65, no. 7, pp. 5701-5706, Jul. 2016.
[5] L. Song, D. Niyato, Z. Han, and E. Hossain, Game-theoretic resource allocation methods for device-to-device communication, IEEE Wireless Communications, vol. 21, no. 3, pp. 136-144, 2014.
[6] M. Ji, G. Caire, and A. F. Molisch, Wireless device-to-device caching networks: Basic principles and system performance,CoRR, vol. abs/1305.5216, 2013.
[7] J. Li, Y. Chen, Z. Lin, W. Chen, B. Vucetic and L. Hanzo, Distrusted Caching for data dissemination in the heterogeneous network, IEEE Transactions on communications, vol. 63, no. 10, pp. 3553-3568, Jul. 2015.
[8] L. Marini, J. Li, Y. Li and B. Vucetic, Distributed caching based on decentralized learning automata, in IEEE international Conference on communications (ICC). IEEE, 2015.
[9] Chuan. Ma, Zihuai. Lin, Loris. Marini, Jun. Li, and Branka. Vucetic,Learning automata based distributed caching for mobile social networks in IEEE wireless communications and network conference (WCNC), April. 2016.
[10] M. Agache and B. Oommen, Generalized pursuit learning schemes new families of continuous and discrete learning automata, Systems,Man, and Cybernetics, part B: Cybernetics, IEEE Transactions vol. 32, no. 6, pp. 738-749,Dec 2002.
[11] S. Borst, V. Gupta, and A. Walid, Distributed caching algorithms for content distribution networks, in Proc. IEEE INFOCOM, 2010, pp. 19.
[12] D. L. Donoho, A. Maleki, and A. Montanari, Message passing algorithms for compressed sensing, arXiv:0907.3574v1 [cs.IT], Jul. 2009.