Mobility-Aware Joint Task Scheduling and Resource Allocation for Cooperative Mobile Edge Computing

Umber Saleem, Yu Liu, Sobia Jangsher, *Member, IEEE*, Yong Li, *Senior Member, IEEE*, and Tao Jiang, *Fellow, IEEE*

***Abstract*— Mobile edge computing (MEC) has emerged as a new paradigm to assist low latency services by enabling computation offloading at the network edge. Nevertheless, human mobility can significantly impact the offloading decision and performance in MEC networks. In this context, we propose device-to-device (D2D) cooperation based MEC to expedite the task execution of mobile user by leveraging proximity-aware task offloading. However, user mobility in such distributed architecture results in dynamic offloading decision that instigates mobility-aware task scheduling in our proposed framework. We jointly formulate task assignment and power allocation to minimize the total task execution latency by taking account of user mobility, distributed resources, tasks properties, and energy constraint of the user device. We first propose Genetic Algorithm (GA)-based evolutionary scheme to solve our formulated mixed-integer non-linear programming (MINLP) problem. Then we propose a heuristic named mobility-aware task scheduling (MATS) to obtain effective task assignment with low complexity. The extensive evaluation under realistic human mobility trajectories provides useful insights into the performance of our schemes and demonstrates that, both GA and MATS achieve better latency than other baseline schemes while satisfying the energy constraint of mobile device.**

***Index Terms*— Mobile edge computing, device-to-device cooperation, human mobility, task scheduling, latency minimization.**

I. Introduction

WITH the convergence of Internet of Things (IoT), 5G, and artificial intelligence, resource hungry and mission-critical smartphone applications are evolving rapidly [1]. However, the mobile devices are facing the dilemma of limited computing resources, which makes it challenging for them to embrace such applications. Although cloud computing can cope with large amounts of data and processing, the Quality-of-Service (QoS) would be compromised due to the centralized architecture which does not provide location awareness and mobility support.

In this regard, mobile edge computing (MEC) has emerged as a promising approach which provides powerful computing and storage capabilities closer to the end users [2]. Especially, MEC allows computation offloading that has been proven to improve the application response time and energy efficiency [3]. Meanwhile, the device-to-device (D2D) cooperation based MEC architecture that leverages the nearby idle resources for task execution has attracted a lot of attention [4]. D2D computation offloading not only tackles the resource contention in case of stand-alone MEC server, but also benefits from lower latency and higher data rates of D2D communications [5]. In this way, the task offloading performance is scaled up without any additional cost. However, a key design challenge is to determine the destination for offloading due to the distributed architecture. Therefore, task scheduling is the fundamental problem while designing offloading mechanism for users in cooperative MEC network.

Besides task scheduling and resource management problems, human mobility can largely impact the computation offloading experience. With the frequent location changes of offloading user, the feasible allocation of communication and computation resources vary over the time. Specifically, time varying channel gains degrade the QoS between mobile user and MEC server, and thus it becomes critical to uphold the task offloading latency. In this context, D2D cooperation can enable proximity-aware task offloading with reduced delay by leveraging frequent contacts with nearby devices. However, human mobility in presence of heterogeneous computing resources and tasks makes the offloading decision e.g., what, when, and where to offload highly dynamic. Therefore, more sophisticated task scheduling and resource allocation policies should be designed by considering mobility-driven uncertainty to enhance the task offloading experience of a mobile user in the distributed system.

This motivates us to propose mobility-aware task offloading framework in the D2D-enabled cooperative MEC system. By considering the aforementioned challenges, we design task scheduling schemes to reduce the total latency under the energy consumption constraint of the mobile user with

Manuscript received October 11, 2019; revised March 28, 2020 and July 21, 2020; accepted August 30, 2020. Date of publication September 25, 2020; date of current version January 8, 2021. This work was supported in part by The National Key Research and Development Program of China under Grant 2019YFB180003400, the National Nature Science Foundation of China under U1936217, 61971267, 61972223, 61941117, 61861136003, Beijing Natural Science Foundation under L182038, Beijing National Research Center for Information Science and Technology under 20031887521, and Research Fund of Tsinghua University-Tencent Joint Laboratory for Internet Innovation Technology. The associate editor coordinating the review of this article and approving it for publication was J. Lee. *(Corresponding author: Yong Li.)*

Umber Saleem, Yu Liu, and Yong Li are with the Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: anb17@mails.tsinghua.edu.cn; liuyu2419@126.com; liyong07@tsinghua.edu.cn).

Sobia Jangsher is with the Wireless and Signal Processing (WiSP) Lab, Department of Electrical Engineering, Institute of Space Technology (IST), Islamabad 44000, Pakistan (e-mail: sobia.jangsher@ist.edu.pk).

Tao Jiang is with the Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: tao.jiang@ieee.org).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.3024538

multiple tasks along known trajectory [6]. In particular, our proposed schemes utilize user location information while deciding the offloading location and time, and meanwhile deal with the underlying power-delay tradeoff.

A. Contribution and Organization

The main contributions of our work can be summarized as follows.

1) We consider human mobility and propose mobility-aware task scheduling in a D2D-enabled cooperative MEC framework to minimize task offloading latency. Accordingly, we jointly formulate task assignment and power allocation as a mixed-integer non-linear programming (MINLP) problem by taking account of user mobility, distributed computation capacities and task properties in presence of user energy constraint.
2) To solve the formulated MINLP problem, we propose two task scheduling approaches. First approach is the Genetic Algorithm (GA)-based evolutionary scheme which obtains the solution directly. Second approach is a low complexity heuristic scheme named Mobility-Aware Task Scheduling (MATS), which yields further insight into designing effective offloading mechanism by considering network dynamics.
3) We use real-world dataset of human mobility trajectories [7] to evaluate the performance of our proposed schemes under various key parameters. Comparison with other baselines depicts that, both GA and MATS achieve lower latency while ensuring the desired energy consumption of the mobile user.

The rest of the paper is organized as follows. Section II provides an overview of relevant works. Section III presents the system model and discuses the mobility, task and computation offloading models in detail. Section IV discusses the latency minimization problem formulation, while Section V describes our proposed approaches to obtain effective task scheduling strategies. Section VI presents the results and performance analysis. Section VII finally concludes this work and highlights future direction.

II. Related Work

Mobile edge computation offloading has been widely considered to improve task execution latency or energy in various scenarios [4]. For the simple case of single-user single-server MEC system, the authors investigated offloading decision (local/remote execution) together with computation resource allocation to design effective offloading strategies [8]–[11]. However, several new challenges are raised when multiple users share the stand-alone MEC server. Keeping in view the finite radio and computing resources in multi-user single-server MEC scenario, the authors studied joint communication and computation resource allocation besides the computation offloading decision [12]–[15]. Although the preceding works demonstrated efficacy of MEC, the rapidly increasing users and wide range of applications demand for ubiquitous computing paradigm.

In this regard, cooperative MEC systems were proposed consisting of heterogeneous MEC servers. Specially, D2D collaboration for task offloading has received a lot of attention [16]–[19]. The underlying motivation is two-fold. First, plenty of underutilized resources available in the network bring in diversity without any additional cost. Second, the proximity gain, low latency, energy efficiency and better coverage of D2D communication makes it ideal for ubiquitous MEC [5]. However, a fundamental design challenge in cooperative MEC systems is task scheduling i.e. where to offload, which is unique due to distributed computing resources and varying tasks characteristic.

The works in [16]–[19] studied task scheduling while considering computation offloading problem in D2D-enabled MEC network specifically. Pu *et al.* [16] investigated energy efficient task execution by taking into account the incentive constraints, while Zhao *et al.* [17] investigated similar problem by jointly considering user association and power allocation. Online task offloading algorithms based on Lyapunov optimization were proposed in both works to minimize the energy consumption of task execution. In contrast, Feng *et al.* [18] jointly formulated task scheduling, subcarrier assignment, and power allocation as a stochastic optimization problem with objective to minimize the average expenses of collaborating devices in terms of wireless communication and computation. The authors proposed a suboptimal algorithm by decoupling the optimization variables. To minimize the latency for a single user multi-task cooperative MEC network, Xing *et al.* [19] jointly formulated task assignment and wireless resource allocation as MINLP problem. The authors proposed a suboptimal solution based on the optimal solution to the relaxed convex problem.

It is important to note that, task assignment and resource allocation strategies proposed in the aforementioned studies were designed while considering static users in distributed MEC network. However, user mobility-driven uncertainty makes task scheduling decision more complex in presence of heterogeneous radio and computing resources [4]. Hence, user location information needs to be incorporated while designing task scheduling schemes for mobile users in distributed MEC scenario.

Considering mobile nature of users, Wang *et al.* [20] and Hong *et al.* [21] modeled user mobility in D2D computation offloading scenario. Wang *et al.* [20] maximized the probability of successful task computing by exploiting statistical property of contact rates together with the computation capability while making the offloading decision. In particular, the authors used convex optimization to determine the amounts of computation to be offloaded to other devices. However, the task scheduling strategy proposed in [20] cannot be adopted for atomic tasks which cannot be partitioned and need to be executed as a whole either locally or remotely. Moreover, the task diversity need to be considered together with the resource diversity while making the task offloading decision for a multi-task multi-server scenario. On the other hand, the authors in [21] proposed a probabilistic model for connectivity based on a continuous-time Markov chain with aim to minimize the cooperative task execution time

in D2D network. A heuristic algorithm based on linear programming was devised to solve the connectivity-aware computation assignment. Although the authors incorporated the connectivity-awareness while minimizing task execution time, under the consideration of user mobility, the QoS guarantee is critical [3]. Meanwhile, the accuracy of intuitive models cannot be guaranteed in diverse real life scenarios due to which they may yield ineffective task scheduling strategies.

Unlike the prior works, the authors in [22]–[24] exploited user mobility prediction motivated by the fact that human trajectory can be predicted with 90% accuracy using trajectory data mining [6]. Plachy *et al.* [22] and Wang *et al.* [23] studied load balancing among the heterogeneous MEC servers using different approaches. In particular, Plachy *et al.* [22] proposed VM migration to solve user mobility problem. The authors considered MDP-based dynamic resource allocation which jointly performed VM migration and communication path selection to minimize the task execution delay of users. On the contrary, the authors in [23] focused on reducing the delay within the mobile edge network by formulating task assignment as a constraint satisfaction problem. The authors proposed a heuristic approach by taking into account the task properties and resource distribution among MEC servers. However, both [22] and [23] overlooked the energy constraint while minimizing the computation offloading delay. Meanwhile energy availability at the user directly impacts the transmission latency, and thus degrades the overall task offloading latency. Keeping in view that mobile users are resource-poor as compared to static ones, Zhu *et al.* [24] exploited user mobility prediction to formulate energy minimization problem while considering latency as a constraint. Nevertheless, while minimizing task execution latency for a resource-constrained mobile user the underlying power-delay tradeoff is the fundamental challenge to be addressed.

Although the prior works have studied mobility-aware computation offloading, the problem of task execution latency minimization subject to energy constraint of mobile user in cooperative MEC scenario has not been addressed so far. In our work, we novelly present the joint task assignment and power allocation schemes to minimize latency while preserving energy of mobile user in D2D-enabled MEC network.

III. System Model

We consider a D2D-enabled cooperative MEC network, where the idle and resource rich devices such as phone, tablet, laptop and desktop computer can serve as fog/edge nodes to facilitate computation offloading by establishing direct D2D links [25] with a resource limited task device (TD) indexed by 0. We assume that TD is carried by a walking user moving along a trajectory. Meanwhile, TD has K independent computation intensive tasks to execute denoted by the set $\mathcal{K} = \{1, 2, \ldots, K\}$. Along the path of TD, M resource devices (RDs) including cellphones, smart wearable devices, tablets, and laptops are distributed. The set of RDs is denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. We consider a network assisted architecture, where the base station (BS) has global network information including details about user's mobility, computational resources, and tasks. We assume that BS can discover the updated network topology and location of users by using bootstrapping program similar to [26].

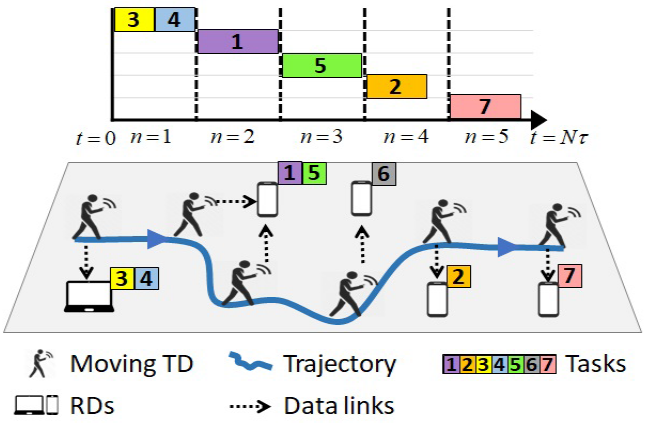


Fig. 1. Mobility-aware task offloading in D2D-enabled cooperative MEC network.

For task offloading, TD can connect to any of the RDs by establishing a direct D2D link (using technologies such as WiFi-Direct or Bluetooth) with the assistance of BS. BS anticipates TD's mobility in order to schedule its tasks to RDs such that the computation offloading latency is minimized. Fig. 1 illustrates the considered scenario where the tasks of mobile user are scheduled on RDs along user's trajectory. The notations that will be used in the rest of this paper are summarized in Table I.

A. User Mobility Model

As human trajectory can be predicted with 90% accuracy using trajectory data mining [6], [23], we build our mobility model based on obtained trajectory of a pedestrian. Particularly, we assume that trajectory of TD is available for time $[0, T]$, where $t = 0$ is regarded as the time when task buffer of TD is full with K tasks. Assuming that the space/region of cell is partitioned into 2-D space, the location of TD at time instant $t \in [0, T]$ is represented as $l(t) = \{x(t), y(t)\}$. Hence, we have trajectory of TD with initial and final location as $l(t = 0)$ and $l(t = T)$, respectively.

Similar to the existing works in [27] and [28], we assume the system to operate in a slotted structure to precisely capture the user mobility. Hence, we divide the mobility trajectory over time T in N time slots of equal duration τ such that $T = N\tau$. The set of time slots is represented as $\mathcal{N} = \{1, 2, \ldots, N\}$. We choose the value of N such that the location of TD is approximately unchanged in each slot. Here, we emphasize that the velocity of walking user is relatively slow and the distance covered over short time does not vary at large [24]. Now, the horizontal location of TD in slot $n \in \mathcal{N}$ is denoted as $\hat{l}(n) = l(n\tau)$. Based on the location of TD in slot $n \in \mathcal{N}$, the distance $d^m(n)$ between TD and RD $m \in \mathcal{M}$ can be obtained as

$$d^m(n) = \| \hat{l}(n) - l^m \|, \tag{1}$$

TABLE I
NOTATIONS

Notation	Definition	Notation	Definition
$\mathcal{K}$	Set of tasks	$a_k^m(n)$	Task assignment
$\mathcal{M}$	Set of RDs	$\boldsymbol{a}$	Task assignment profile
$\mathcal{N}$	Set of slots	$p^m(n)$	Power allocated to RD m in slot n
m	Index of RD	$\boldsymbol{p}$	Power allocation profile
n	Index of slot	$p^{\max}$	Maximum transmit power of TD
τ	Slot duration	t_k^{loc}	Local execution latency of task k
D_k	Data size of task k	e_k^{loc}	Local execution energy of task k
C_k	Computation complexity of task k	t_k^{off}	Transmission latency of task k
f^m	Computation capacity of RD m	e_k^{off}	Transmission energy of task k
f^0	Computation capacity of TD 0	t_k^{exe}	Remote execution latency of task k
$d^m(n)$	Distance between TD and RD m in slot n	ϵ^0	Expected energy consumption of TD
$R^m(n)$	Data rate of RD m in slot n	T^{tot}	Total latency of task execution

where l^m denotes the location of RD $m \in \mathcal{M}$, and $\| \cdot \|$ is the Euclidean norm.

B. Computation Offloading Model

Each computation task can be described as $\mathrm{I}_k \triangleq (D_k, C_k)$, $k \in \mathcal{K}$, where D_k and C_k denote the task input data size (in bits) and computation intensity (in CPU cycles per bit), respectively. Considering that the tasks cannot be partitioned [4], [18] each task can either be offloaded to one of RDs for remote execution in a slot $n \in \mathcal{N}$, or executed locally on TD.

We introduce the task assignment variable as $a_k^m(n) \in \{0, 1\}$, where $a_k^m(n) = 1$ means that a task $k \in \mathcal{K}$ is scheduled for remote execution on device $m \in \mathcal{M}$ during slot $n \in \mathcal{N}$. The respective task assignment profile is provided as $\mathbf{a} = \{a_k^m(n) |\ k \in \mathcal{K},\ m \in \mathcal{M},\ n \in \mathcal{N}\}$. In each slot, an RD can be assigned at most one task to avoid the server side queueing delay [29]. However, multiple tasks can be scheduled in the same slot on different devices. The execution decision of a task $k \in \mathcal{K}$ can be determined from the task assignment variable as

$$\begin{cases} \sum_{n=1}^{N} \sum_{m=1}^{M} a_k^m(n) = 0, & \text{local execution;} \\ \sum_{n=1}^{N} \sum_{m=1}^{M} a_k^m(n) = 1, & \text{remote execution;} \end{cases} \tag{2}$$

It is important to note that we build our model based on pedestrian's trajectory, where slot duration can be sufficiently large to execute a task [9]. Therefore, it is empirically established that for each task the required computation is always completed within the same slot. Hence, it can be inferred that each task is executed by only one device (either TD or RD $m \in \mathcal{M}$) along the path.

Based on the task assignment, the computation offloading latency of a task is determined by the local execution or offloading delays, where the latter involves transmission as well as remote execution time. We discuss the model for each of these processes in the following sections.

1) Local Execution: In case of higher computation offloading cost, local execution is preferred. We assume that TD has a fixed CPU frequency denoted as f^0 (in CPU cycles per second). Then the time consumed by TD to execute a task locally is given as

$$t_k^{\text{loc}} = \sum_{n=1}^{N} \sum_{m=1}^{M} \frac{(1 - a_k^m(n)) C_k D_k}{f^0}. \tag{3}$$

The energy consumed by TD for computing task k locally is given as [4]

$$e_k^{\text{loc}} = \left(1 - \sum_{n=1}^{N} \sum_{m=1}^{M} a_k^m(n)\right) \kappa^0 C_k D_k {f^0}^2, \tag{4}$$

where κ^0 denotes the effective capacitance coefficient with a constant value which depends on the chip architecture of TD [10]. Keeping in view the limited energy of TD, we assume that the total energy consumed by TD for local computing and communication over N slots is limited by the expected energy consumption ϵ^0 [19].

2) Communication: Tasks which are scheduled for remote execution need to be transmitted via D2D links. The transmission delay of D2D communication is affected by the wireless channel states, and the size of the task input size. Therefore, offloading decision should take into account the communication cost as well. We use TDMA scheme to model the communication between TD and RDs. Hence, TD transmits each task to its corresponding RD via TDMA by establishing a D2D link. We assume independent block fading channels for D2D links, such that channel state remains invariant during a slot and varies across different slots. We model the channel between TD and RD $m \in \mathcal{M}$ during slot $n \in \mathcal{N}$ by considering distance dependent path loss, multipath fading and shadowing. Thus the channel gain $g^m(n)$ can be given as [30]

$$g^m(n) = \theta d^m(n)^{-\alpha} \|h^m\|^2 \zeta, \tag{5}$$

where θ and α are the path loss coefficient and exponent, respectively. $d^m(n)$ is the distance between TD and RD m during slot n, and its value can be obtained using (1).

h^m is the Rayleigh fading coefficient, and ζ is the log-normal distributed shadowing. It is important to note that, we consider pedestrian mobility which has relatively slow velocity. Meanwhile, the slot duration τ is chosen such that there is insufficient change in user's location over the same slot. Hence, it is practical to assume that the little variation in distance will not affect the transmission rate of D2D link during a slot [24].

As TD has time varying channel gain due to mobility, we assume that it can adopt its transmission power to each resource in a particular slot. Let $p^m(n)$ denote the transmission power from TD to resource $m \in \mathcal{M}$ in slot $n \in \mathcal{N}$. Then the power allocation profile can be defined as $\mathbf{p} = \{p^m(n) | m \in \mathcal{M}, n \in \mathcal{N}\}$. Let B denote the total channel bandwidth in Hz. Using Shannon formula, the maximum achievable data rate from TD to RD m in slot n is given as

$$R^m(n) = B\log_2\left(1 + \frac{p^m(n)h^m(n)}{N_0}\right), \tag{6}$$

where N_0 denotes the noise power. Based on the task assignment decision and the corresponding data rate, the time required to offload task k is given as

$$t_k^{\text{off}} = \sum_{n=1}^{N}\sum_{m=1}^{M}\frac{a_k^m(n)D_k}{R^m(n)}. \tag{7}$$

Assuming fixed transmission rate D_k/t_k^{off} [19], the energy consumed by TD for offloading a task k can be expressed as

$$e_k^{\text{off}} = \sum_{n=1}^{N}\sum_{m=1}^{M}\frac{a_k^m(n)p^m(n)D_k}{R^m(n)}. \tag{8}$$

3) Remote Execution: Once a task is offloaded to RD it is executed remotely and the output results are downloaded at TD. Here we assume that computation results are relatively much smaller in size and the results downloading takes negligible time [10], [31]. The time consumed by resource $m \in \mathcal{M}$ to compute a task $k \in \mathcal{K}$ is given as

$$t_k^{\text{exe}} = \sum_{n=1}^{N}\sum_{m=1}^{M}\frac{a_k^m(n)D_kC_k}{f^m}, \tag{9}$$

where f^m denotes the computation capacity of RD $m \in \mathcal{M}$, while the above equation holds only if $a_k^m(n) = 1$. Keeping in view the diversity among the devices, we assume that the computation capacity of RDs vary, and thus the selection of RD effects the remote execution delay.

IV. Problem Formulation

In this work, we aim to minimize the total computation offloading latency of a mobile TD including the local execution, offloading and remote execution delays. We define our objective as

$$T^{\text{tot}} = \sum_{k=1}^{K}\left(t_k^{\text{loc}} + t_k^{\text{off}} + t_k^{\text{exe}}\right). \tag{10}$$

In order to reduce the offloading delay, we make the task offloading decision by utilizing the location information at different time slots along user's trajectory. We formulate joint task assignment and resource allocation problem to minimize the total latency of task execution as follows

$$\textbf{P1:} \min_{\mathbf{a},\mathbf{p}} T^{\text{tot}} \tag{11a}$$

$$\text{s.t.} \sum_{k=1}^{K}\sum_{m=1}^{M}\frac{a_k^m(n)D_k}{R^m(n)} + \max_{\substack{k\in K,\\ m\in M}}\left(\frac{a_k^m(n)D_kC_k}{f^m}\right) \le \tau, \quad \forall n \in \mathcal{N} \tag{11b}$$

$$\sum_{k=1}^{K}(e_k^{\text{loc}} + e_k^{\text{off}}) \le \epsilon^0, \tag{11c}$$

$$\sum_{n=1}^{N}\sum_{m=1}^{M}a_k^m(n) \le 1, \quad \forall k \in \mathcal{K} \tag{11d}$$

$$\sum_{k=1}^{K}a_k^m(n) \le 1, \quad \forall m \in \mathcal{M},\ \forall n \in \mathcal{N} \tag{11e}$$

$$0 \le p^m(n) \le p^{\max}, \quad \forall m \in \mathcal{M}\ \forall n \in \mathcal{N} \tag{11f}$$

$$a_k^m(n) \in \{0,1\}, \quad \forall k \in \mathcal{K},\ \forall m \in \mathcal{M}, \forall n \in \mathcal{N} \tag{11g}$$

Here, (11a) shows our objective function with the task assignment and power allocation as optimization variables. Constraint in (11b) bounds the offloading latency of the tasks assigned within a slot. It shows that the tasks scheduled in each slot should finish execution within the same slot. Constraint in (11c) depicts that the total energy consumed by TD for local computing and transmission over N slots should not exceed the expected energy consumption ϵ^0. Constraint in (11d) presents that a task can be assigned to at most one resource in a particular slot as a task cannot be partitioned. On the other hand, constraint in (11e) presents that each resource can be assigned at most one task in a particular slot, which ensures the fairness among RDs. It is important to note that constraints in (11d) and (11e) together ensure that each task would be scheduled in at most one slot; otherwise would be executed locally. Constraint in (11f) shows that the transmit power from TD to any resource in a slot is bounded by maximum power $p^{\max}$ in uplink. Constraint in (11g) shows that task assignment is a binary decision variable which implies that a task cannot be partitioned.

The formulated optimization problem has a mixed integer objective function with binary and continuous variables $a_k^m(n)$ and $p^m(n)$, respectively. In addition, the constraints in (11b) and (11c) are non-linear due to relationship between task assignment $a_k^m(n)$ and power allocation $p^m(n)$. In terms of complexity, the formulated problem is NP-hard [32], and thus the exhaustive search approach is unacceptable.

Theorem 1: The joint task assignment and power allocation problem in **P1** is NP-hard.

Proof: In order to prove the NP-hardness of problem **P1**, we first consider special case of task assignment and power allocation over single slot. As user location does not vary at large during a slot, we assume that power allocation is fixed such that the local energy consumption constraint of user is satisfied for the slot duration. As a consequence, the problem **P1** is transformed to latency minimization by optimal task assignment. We then reduce the multiple knapsack problem

which is a well-known NP-hard problem [32] to this special case.

Multiple knapsack problem [33]: Given a set of items and a set of knapsacks, there is a weight and profit of each item and a capacity of each knapsack. Then multiple knapsack problem is to select and assign a disjoint subset of items to a unique knapsack such that the total profit of selected items is maximized, while the capacity of knapsack is no less than the weight of the selected items. In this context, we can regard the set of K tasks and M RDs as the items and knapsacks, respectively. The computing resources required by each task is the weight, while the computation offloading latency is the profit. The computation resource of RD is the capacity. Then, filling the items to knapsacks is equivalent to assigning the tasks to the RDs such that the computation offloading latency is minimized without exceeding the available resources of RDs. Since this special case of problem **P1** is NP-hard, it can be inferred that the problem of task assignment and power allocation over multiple slots is also NP-hard.

V. Joint Task Assignment and Power Allocation

To solve our formulated MINLP latency minimization problem, we propose two different approaches: 1) GA which is a metaheuristic belonging to larger class of evolutionary algorithms, 2) MATS which is a low complexity search heuristic.

A. GA-Based Task Scheduling

We adopt GA to obtain a baseline solution motivated by the fact that GAs have been widely adopted by the researchers to solve the job-shop scheduling problem (JSSP) [34]. Our task scheduling problem is similar to JSSP where a set of jobs has to be processed on a set of machines such that a specific optimization criterion is satisfied. It is hard to obtain solution for such problems in polynomial time. In this regard, unlike heuristic search methods, GA is a meta-heuristic that operates on a population of solutions rather than a single solution and has been reported to produce better solutions. In particular, GA inspires from the process of natural evolution and survival of the fittest to evolve solution [35]. Starting with an initial population (set of solutions), GA selects individual solutions (chromosomes) as parents to produce children for the next generation using genetic operations such as crossover, mutation, and fitness evaluation. Over successive iterations, the population converges toward the near-optimal solution. We describe the respective terms, and provide work flow of GA-based scheduling scheme in Algorithm 1 in the following discussion.

1) Chromosomes and Population: In GA, the term chromosome refers to a potential solution of the optimization problem, and population is a set of candidate solutions/chromosemes. We provide a chromosome representation as $v_j = \{a_1^1(1), \ldots, a_k^m(n), \ldots, a_K^M(N), p^1(1), \ldots, p^m(n), \ldots, p^M(N)\}$, where j is the chromosome sequence number. Each variable on the chromosome is referred as a gene. The chromosomes are encoded by GA in a way that the constraints in (11f) and (11g) are always satisfied. Then a population is defined as $\mathcal{P} = \{v_1, \ldots, v_j, \ldots, v_P\}$, where P is the population size or number of chromosomes.

Algorithm 1 GA-Based Task Scheduling

Input: $\mathcal{K}$, $\mathcal{M}$, $\mathcal{N}$, I_k, $\forall k \in \mathcal{K}$, f^m $\forall m \in \mathcal{M}$, f^0, ϵ^0, $p^{\max}$, τ, G, δ, γ
Output: $\boldsymbol{a}$, $\boldsymbol{p}$
Encode $\boldsymbol{a}$ and $\boldsymbol{p}$ into a chromosome v;
Create a random initial population $\mathcal{P}$;
Score each chromosome in $\mathcal{P}$ by computing fitness values;
$f_{\min}(t) =$ minimum fitness value in $\mathcal{P}$;
while $t \leq G$ **do**
  $f_{\min}(t-1) = f_{\min}(t)$;
  Select parents using stochastic tournament method;
  Perform crossover with probability ρ;
  Perform mutation with probability ϕ;
  Score each chromosome in $\mathcal{P}$ by computing fitness values using (12);
  $f_{\min}(t) =$ minimum fitness value in $\mathcal{P}$;
  if $|f_{\min}(t-1) - f_{\min}(t)| \leq \delta$ **then**
    break;
  end if
  $t = t + 1$;
end while

2) Fitness Function: A fitness function is used to check quality of each solution in the population, while a lower fitness value means higher quality. In most of the optimization problems, the fitness function is in accord with the objective function [36]. In this paper, we aim to minimize the total task offloading latency and meanwhile satisfy the local energy consumption requirement and task assignment constraints. Consequently, we define the fitness function in terms of our objective and add penalty to punish the chromosome violating the constraints in (11b)-(11e), similar to the works in [37] and [38]. The fitness function is given as

$$\begin{aligned} f = T^{\text{tot}} + \gamma\Big(&\max\{0, \sum_{k=1}^{K}\sum_{m=1}^{M} \frac{D_k}{a_k^m(n)R^m(n)} - \tau\} \\ &+ \max\{0, \sum_{k=1}^{K}(e_k^{loc} + e_k^{off}) - \varepsilon^0\} \\ &+ \max\{0, \sum_{m=1}^{M} a_k^m(n) - 1\} + \max\{0, \sum_{k=1}^{K} a_k^m(n) - 1\}\Big), \end{aligned} \tag{12}$$

where γ is a very large positive number. It is important to note that, the penalty function in (12) speeds up the convergence by avoiding the infeasible regions.

3) Selection: In each iteration, i.e. generation, the parents are selected from existing population such that better offsprings (solutions) are produced in the new generation. We select the parents using tournament method, which randomly selects N_t chromosomes from the population and runs tournaments among them. The chromosome with best fitness is selected as a parent for the crossover and mutation. In comparison to other selection procedure, this method has lower computational cost and maintains diversity in solutions.

4) Crossover and Mutation: GA iteratively performs selection, crossover and mutation to evolve the population from one generation to the next. Specifically, crossover and mutation are the two techniques which are adopted to avoid the slow and premature convergence to suboptimal solutions. The crossover operation is performed by choosing a crossover point on a pair of parent chromosomes and then exchanging the genes. On the other hand, mutation produces new offsprings by randomly altering the genes on parent chromosomes.

We provide a brief summary of Algorithm 1. As a first step of GA, population of size P is chosen randomly from the search space of encoded solutions in Line 4. In Line 5, fitness value of each chromosome in $\mathcal{P}$ is computed using the fitness function provided in (12). The solution in current population with highest fitness/lowest fitness value is stored in Line 6, and it is updated in each iteration. To produce next generation in each iteration, first the parents are selected in Line 9, and then crossover and mutation operations are performed to produce new solutions in Line 10 and 11, respectively. Our stopping criteria for GA is based on maximum number of generations denoted as G or the minimum tolerance δ. Thus, the algorithm runs until the maximum number of generations is reached or the relative change in fitness value is less than the tolerance.

Finally, we discuss the complexity of proposed scheme. The complexity of initializing the population of size P with K tasks, M resources and N slots is given by $O(P(KMN + MN)) \approx O(PKMN)$. The genetic operations of selection, crossover and mutation are performed iteratively for each generation. The complexity of these genetic operations for maximum G generations is given by $O(GPN_t) + O(GP) + O(GPKMN)$ and is dominated by $O(GPKMN)$. The time complexity of fitness evaluation for G generations each with population size P is also $O(GPKMN)$. Hence, the overall time complexity of GA-based scheme can be approximated by $O(GPKMN)$.

B. Mobility-Aware Task Scheduling Heuristic

Keeping in view that the complexity of GA does not scale well with the problem size, we devise a low complexity task scheduling heuristic algorithm. The key idea is that, using location information select RD that yields minimum delay for each task. As channel quality and computation capacity varies across different RDs, first a selection criteria needs to be defined. Besides, the feasible set of offloading RDs changes across different slots as respective channel gains vary with mobility. Therefore, we take into account both time varying data rates and computation capacities of RDs to define a weighted sum utility matrix $\boldsymbol{U} = [u_{mn}]_{M\times N}$. In particular, u_{mn} corresponds to the utility of RD $m \in \mathcal{M}$ in slot $n \in \mathcal{N}$, and computed as

$$u_{mn} = w_r \bar{R}^m(n) + w_f \bar{f}^m, \tag{13}$$

where $w_r > 0$ and $w_f > 0$ denote the relative weights of data rate and computation capacity, respectively, such that $w_r + w_f = 1$. The $\bar{}$ shows the scaling operation performed on original values due to different units. The relative value of the weights reflects the relative importance of the two different criteria in utility. It implies that, RD with higher data rate would be preferred when communication capacity becomes worst than computation capacity, and vice versa.

Algorithm 2 MATS Algorithm

1: **Input:** $\mathcal{K}$, $\mathcal{M}$, $\mathcal{N}$, I_k, $\forall k \in \mathcal{K}$, f^m $\forall m \in \mathcal{M}$, f^0, ϵ^0, $p^{\max}$, τ
2: **Output:** $\boldsymbol{a}$, $\boldsymbol{p}$
3: Fix transmission power in $\boldsymbol{p}$ to $p^{\max}$;
4: **for** each $m \in \mathcal{M}$ **do**
5: **for** each $n \in \mathcal{N}$ **do**
6: Assign $\boldsymbol{U}(m,n) = u_{mn}$ for RD m in slot n using equation 13;
7: **end for**
8: **end for**
9: Sort RDs in $\boldsymbol{U}$ in descending order of utilities;
10: Generate a vector $|\boldsymbol{K_c}| = K$ with tasks in descending order of required computation $C_k D_k$;
11: **for** each $u_{mn} \in \boldsymbol{U}$ **do**
12: **for** each task $k \in \boldsymbol{K_c}$ **do**
13: **if** $\boldsymbol{A}(k) = 0$ **then**
14: Compute $t_k^{\text{off}}, t_k^{\text{exe}}$, and e_k^{off} using (7), (9), and (8), respectively;
15: **if** $e_k^{\text{off}} \leq e^{\text{rem}}$ **then**
16: Find already assigned tasks in slot n;
17: **if** latency constraint in (11b) is satisfied **then**
18: Assign task k to RD m in slot n as $a_k^m(n) = 1$, and set $\boldsymbol{A}(k) = 1$;
19: Update $e^{\text{rem}} = e^{\text{rem}} - e_k^{\text{off}}$;
20: break;
21: **else**
22: continue;
23: **end if**
24: **else**
25: Task k will be executed locally, set $\boldsymbol{A}(k) = 1$;
26: break;
27: **end if**
28: **else**
29: continue;
30: **end if**
31: **end for**
32: **end for**

We summarize our proposed heuristic in Algorithm 2, and discuss the details here. In order to determine that execution decision is made for each task, we define a vector $\boldsymbol{A}_{K\times 1}$ in our algorithm and initialize it to 0 in Line. 2. Once the execution of task k is decided, (to be either local or remote) $\boldsymbol{A}(k)$ is set to 1. We first compute the location-based utility of each RD in each slot in terms of data rate and computation capacity using equation (13) in Line 6. Intuitively, the RD with highest utility provides lowest communication and computation overhead.

In order to assign the tasks, we sort the RDs in descending order of utilities in Line 9. Instead of assigning the tasks sequentially, we first sort them in descending order of required computation in Line 10. As a consequence, the tasks with

higher computation are scheduled for offloading to RDs with better computing capacity and channel quality along the TD's trajectory. Otherwise, it may happen that some computation intensive tasks are assigned to RDs with relatively less computing power and lower channel quality, while lightweight tasks are assigned to better RDs.

The first for loop selects RDs arranged in descending order of utility, whereas the second for loop determines the feasible task assignment based on the constraints of our problem. The constraint in (11d) is always satisfied as the utility matrix has unique entries, which ensures that a particular RD would not be assigned more than one task. Line 13 checks that a task is either assigned for remote or local execution, and meanwhile ensures the constraint in (11e). If the task is not already assigned then algorithm proceeds to check the constraints, and otherwise proceeds to corresponding else statement in Line 29 to choose another task. In case of unassigned task, Line 15 checks that offloading it does not utilize the TD's power beyond the expected value in (11c). In case the energy constraint is satisfied, Line 17 checks that the selected task can be offloaded and completed with in the same slot according to (11b). When both the energy and latency constraints are met, the selected task is assigned to RD in respective slot, and the value of decision vector for k is set to 1 in Line 18. The remaining energy of TD is updated in Line 19. After successful assignment, the inner for loop breaks in Line 20, and the algorithm then proceeds to find feasible task assignment for next RD in $\boldsymbol{U}$.

At first assignment, the corresponding slot is not occupied by any other task. However, the feasibility of successful task completion in the same slot decreases with the subsequent assignments, which eventually results in violation of latency constraint. In that case, the algorithm continues to the next iteration of inner loop in Line 22 to find a new task in $\boldsymbol{K_c}$. With the successive assignments, fewer resources are left in the system due to which the offloading energy consumption bottlenecks the task offloading and offloading is rejected for remaining tasks in Line 25. It may happen that, both local and remote execution become infeasible for a task due to insufficient energy at TD. Therefore, we explicitly determine and state the operating thresholds of different schemes in Section VI by evaluating the energy consumption of TD for varying system load. The algorithms stops when the execution decision for all the task has been made.

Now we discuss the complexity of MATS that mainly lies in the nested for loops of utility computation and task assignment. The complexity of finding the utility of M RDs in N slots is given by $O(MN)$. For task assignment, the outer for loop iterates MN times for all the entries of utility matrix $\boldsymbol{U}$, while the inner for loop iterates for all the tasks K. Hence, the complexity of task assignment phase becomes $O(KMN)$. The overall complexity of MATS is given as $O(MN) + O(KMN)$ and can be approximated by $O(KMN)$. It can be observed that, MATS provide lower complexity as compared to GA whose complexity increases manyfold as the size of problem increases.

TABLE II
SIMULATION PARAMETERS

Parameter	Symbol	Value
Bandwidth	B	1 MHz
Noise power	N_0	-100 dBm
Maximum transmit power of TD	$p^{\max}$	23 dBm
Number of slots	N	5
Slot duration	τ	30 sec
Maximum number of generations	G	200
Tolerance for GA	δ	10^{-3}
Penalty factor for GA	γ	10^{6}

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our proposed scheme for realistic human mobility traces. Particularly, we use well-known pedestrian mobility datasets used in [7]: 1) Orlando dataset collected by 4 participants who usually walked and occasionally rode trolleys in Disney world, Orlando, Florida, United States. 2) KAIST dataset collected by 20 students within the Korea Advanced Institute of Science and Technology (KAIST) campus, Daejeon, Korea.

Garmin GPS 60CSx handheld receivers were used for data collection which can provide position accuracy of better than three meters 95 percent of the time. We extract the {Time stamp, Longitude, Latitude} information of 150 traces from KAIST and Orlando datasets, and map the original data to a square area of $150m \times 150m$. For each (x, y) coordinate of the original data, we perform the resize operation as $\bar{x} = 0 + ({}^{x-x_{\min}}/_{x_{\max}-x_{\min}}) \times 150$, and similarly $\bar{y} = 0 + ({}^{y-y_{\min}}/_{y_{\max}-y_{\min}}) \times 150$ to obtain the resized values $\bar{x}$ and $\bar{y}$ of x and y coordinates respectively. Here, $x_{\max}, x_{\min}$ and $y_{\max}, y_{\min}$ are the maximum and minimum values of x and y coordinates, respectively. For each dataset, we first collect 150 consecutive traces. Then we group 5 positions and regard it as one trajectory, which means that there are 5 slots in one trajectory with slot duration $\tau = 30$ sec as the position data was sampled at every 30 seconds by default. In this way, we obtain 30 trajectories of 150 sec each according to the relation $T = N\tau$ for each dataset.

We set $K = 8$ and $M = 6$, unless otherwise stated. The parameters specific to TD are set as, $f_{\text{TD}} = 1$ GHz, $\epsilon_{\text{TD}} = 7.4$ dB, and $\kappa_{\text{TD}} = 10^{-28}$ [4]. For each task, the data size is uniformly distributed as $D_k \in [5, 10]$ Mb, where CPU cycles are fixed for each task as $C_k = 1500$ cycles per bit. The computing capacity of each RD is selected from the uniform distribution as $f^m \in [2, 3]$ GHz. The path loss coefficient and exponent are set as $\theta = -40$ dB, and $\alpha = 4$, respectively. The other simulation parameters are listed in Table II.

We first analyse the convergence of GA as it works in an iterative fashion by improving the fitness over successive generations. In Fig. 2, we plot the total latency versus number of iterations for $K = 8$, $K = 10$ and $K = 12$ while keeping the number of RDs fixed as $M = 6$. Moreover, KAIST dataset

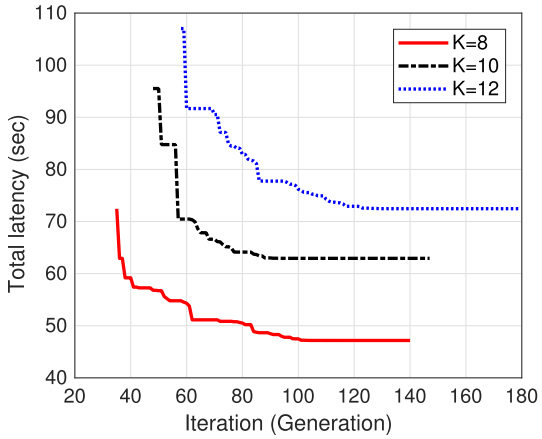


Fig. 2. Convergence of GA for different number of tasks.

is used for this plot. For initial generations, the solutions obtained by the algorithm are infeasible due to which constraint violation penalty is applied to the fitness function [39]. Hence, latency values are not plotted in the figure for infeasible points. It is evident from the figure that, for different number of tasks, GA converges to the best solution before reaching the maximum number of generations. On the other hand, the convergence performance degrades with increase in number of tasks K. It is obvious, as the search space increases with increase in K, and thus GA needs more iterations to converge.

Beside MATS and GA-based schemes, we consider two other heuristic schemes for performance comparison: 1) 'shortest distance first' (SDF) and 2) 'shortest execution first' (SEF). SDF sequentially assigns tasks to the closest RDs with higher channel gains/lower transmission delays. On the other hand, SEF sequentially assigns tasks to the RDs with higher computation capacities/shortest execution time in each slot. SDF, SEF, and MATS perform fix power allocation such that $p^m(n) = p^{\max}$.

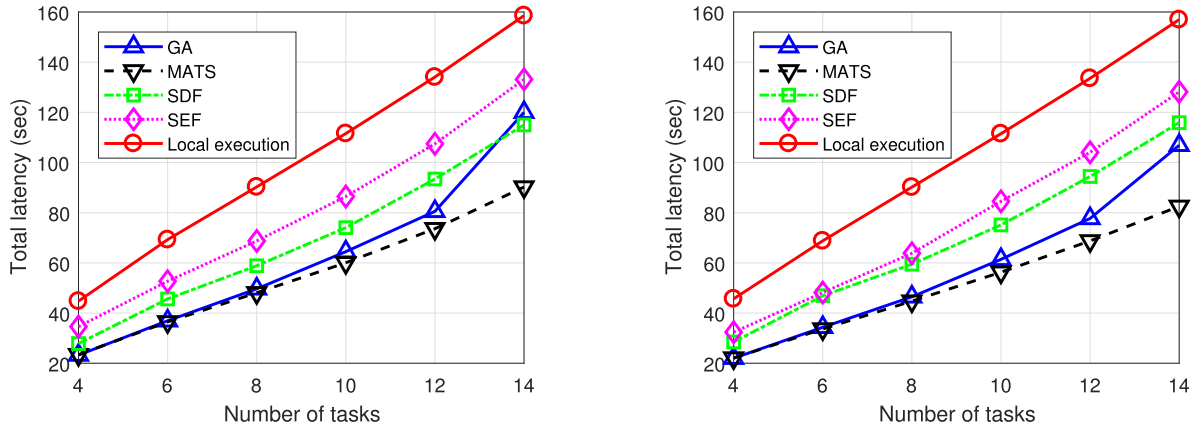
Practically, the computation and communication load can vary at large with the increase in number of service requests. Therefore, we first analyse the scalability of the proposed scheme for increasing number of tasks while keeping the RDs fixed as $M = 6$. In particular, we first analyse the impact on latency performance in Fig. 3, where Fig. 3(a) and Fig. 3(b) correspond to Orlando and KAIST datsets, respectively. The figures depict the increasing latency trend for all the schemes, which is obvious as limited computing and radio resources cannot uphold increasing service requests within the fixed time. Nevertheless, other baseline schemes can always achieve lower latency as compared to local computing due to collaborative task execution. As compared to other schemes, MATS provides lowest latency mainly due to two factors: first is the offloading ordering, and second is joint consideration of computation and communication cost. Meanwhile, it can be observed from Figs. 3(a) and 3(b) that GA provides comparable performance up to $K = 10$ tasks. Beyond that, the exponentially increasing complexity of GA with the increasing number of tasks leads to the delay-energy tradeoff. It is worth mentioning that, the offloading order significantly impacts the performance of heuristics. As a matter of fact, both SDF and SEF fail to benefit from user mobility-driven time varying performance gain due to sequential task assignment. Eventually, SDF and SEF have to assign tasks for local execution as they reach the computation and communication bottlenecks, respectively. In contrast, MATS reduces the latency by jointly considering the location based computing and communication gains along with the computational complexity of tasks.

In real world, the number of executable tasks depends on the available power of mobile device. Therefore, we next analyse the total energy consumed by TD for increasing number of tasks in Fig. 4, and determine the upper bound on number of successfully executable tasks for different schemes based on the energy consumption constraint of TD. Specifically, Fig. 4(a) and Fig. 4(b) show the energy performance in case of Orlando and KAIST datasets, respectively. The figures depict increasing trend for all the schemes as more power is consumed in transmission and local execution with increasing number of tasks. Local execution performs worst as it does not utilizes the edge resources. Energy performance of MATS and GA is better than other baselines, mainly due to the task assignment strategy in the former scheme and optimal power allocation in the latter scheme. In contrast to Fig. 4(b), MATS fails to satisfy the energy constraint for $K = 14$ tasks in Fig. 4(a). To know the underlying reason, we analyse our mobility data which reveals that the trajectories in Orlando dataset are relatively much concentrated as compared to the dispersed trajectories in KAIST dataset. Hence, it can be deduced that the mobility-aware scheduling in MATS looses its essence when mobility of user is low. Meanwhile, it can be noticed that GA compromises the latency to achieve lower energy by controlling the transmission power. On the other hand, computation and communication bottlenecks in SDF and SEF, respectively, lead to local execution which consumes higher energy than offloading. The worst performance of SEF indicates that communication resources in the network are more scarce as compared to computing resources. Therefore, SEF fails to meet the energy constraint of TD beyond $K = 6$ tasks. SDF can successfully execute $K = 8$ tasks.

It can be deduced from Figs. 3 and 4 that, MATS can achieve best latency performance without exhausting the TD up to 12 tasks and in some cases up to 14 tasks. On the other hand, as GA cares more for energy it can always execute up to 14 tasks without violating the energy constraint at the cost of higher latency. Besides, the complexity of GA is much higher than MATS. Therefore, we quantify the reduction in latency by limiting our discussion to $K = 12$ tasks. MATS reduces the total latency by approximately 9% and 12% for Orlando and KAIST datasets, respectively, while satisfying the energy consumption constraint of TD.

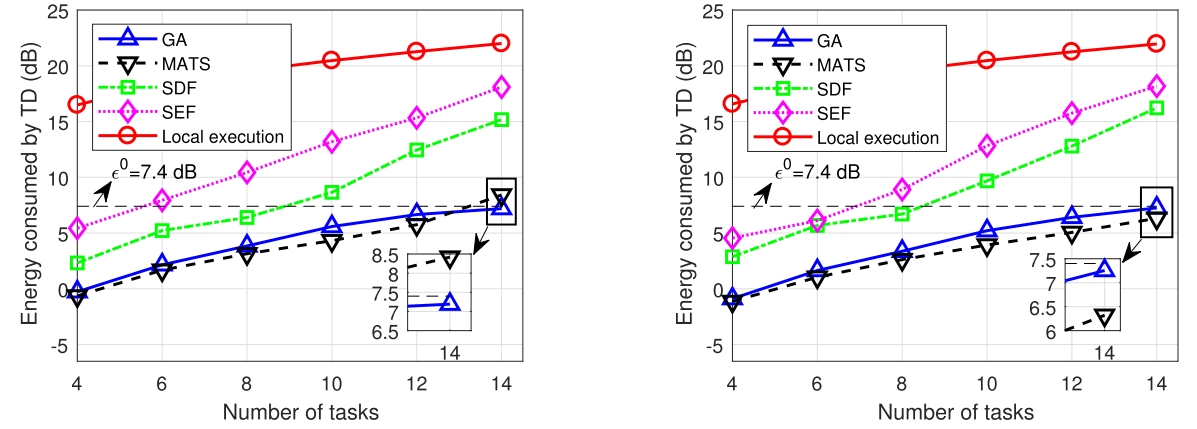
In Fig. 5, we plot efficiency which is measured in terms of percentage reduction in latency and energy similar to [40] as

$$\text{Efficiency} = \beta \left(\frac{T^{\text{loc}} - T^{\text{tot}}}{T^{\text{loc}}} \right) + (1 - \beta) \left(\frac{E^{\text{loc}} - E^{\text{tot}}}{E^{\text{loc}}} \right), \tag{14}$$



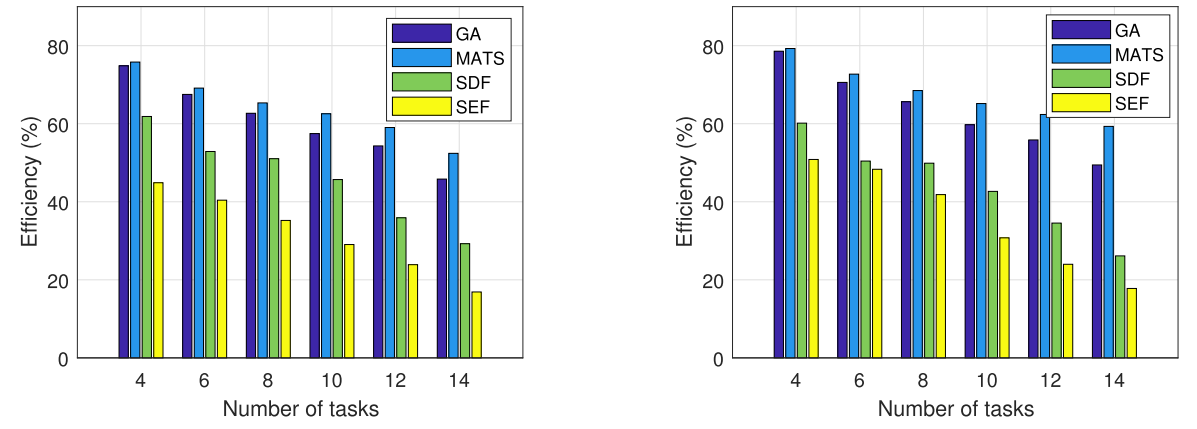
(a) Orlando dataset (b) KAIST dataset

Fig. 3. Effect of increase in number of tasks on total latency.



(a) Orlando dataset (b) KAIST dataset

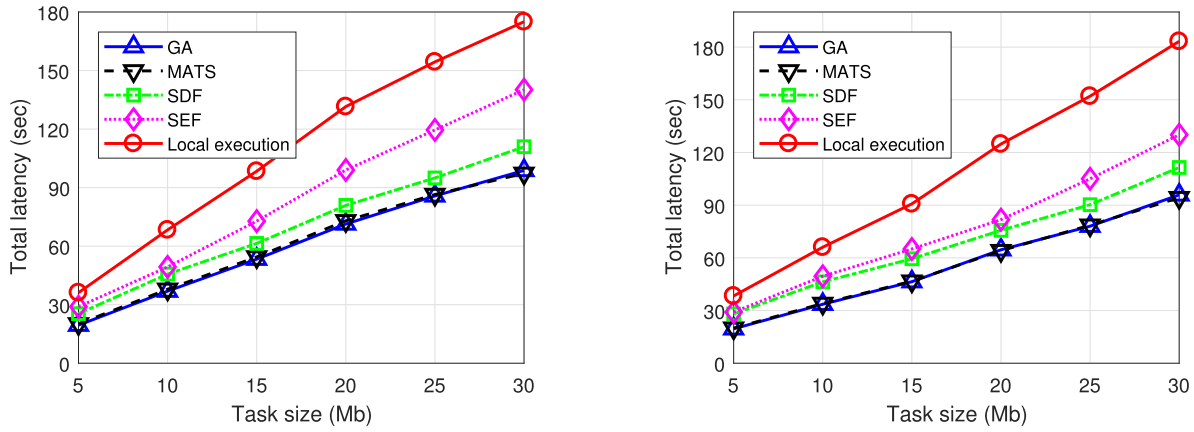
Fig. 4. Effect of increase in number of tasks on total energy consumed by TD.



(a) Orlando dataset (b) KAIST dataset

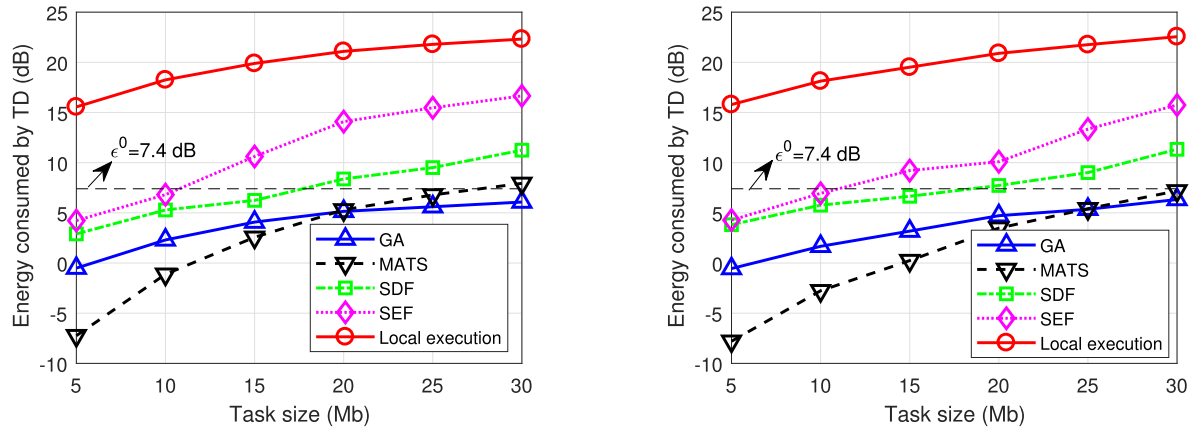
Fig. 5. Effect of increase in number of tasks on efficiency.

where T^{loc} and E^{loc} are the total latency and total energy consumption of TD, respectively, assuming that all the tasks are executed locally. E^{tot} denotes the energy consumed by TD in offloading and local execution, and its value can be obtained according to (11c) as $E^{\text{tot}} = \sum_{k=1}^{K} (e_k^{\text{loc}} + e_k^{\text{off}})$. The first part in (14) represents the fraction reduction in total execution time by offloading the tasks, while the second part indicates the energy reduction. Here, β is the weighting factor in the range $[0, 1]$. We set $\beta = 0.5$ in our simulations, which implies that both latency and energy criteria have equal importance in determining the composite efficiency.



(a) Orlando dataset (b) KAIST dataset

Fig. 6. Effect of increase in task data size on total total latency.



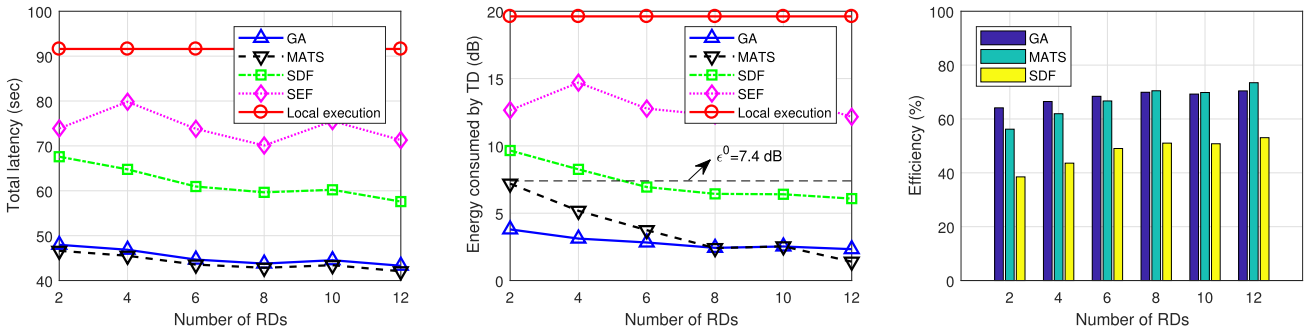
(a) Orlando dataset (b) KAIST dataset

Fig. 7. Effect of increase in task data size on total energy consumption of TD.

Figs. 5(a) and 5(b) show decreasing efficiency with increase in number of tasks for Orlando and KAIST datasets, respectively. The gradually decreasing trends agree with the increasing trends of latency and energy in Fig. 3 and Fig. 4, respectively. It is evident form both the figures that MATS can always achieve higher efficiency than other baselines. In addition, its performance degradation is relatively lower with the increase in number of tasks as compared to other schemes. The efficiency of GA is comparable to MATS initially, while it gradually degrades as the energy constraint becomes hard to satisfy with the increasing number of tasks and underlying complexity. For $K = 12$ tasks, the efficiency of MATS is around 60% which is approximately 10% higher than GA.

The real world user applications are multipurpose, which instigates scalability analysis for varying computation loads in terms of task size. In Fig. 6, we plot latency by increasing the task size from 5 to 30 Mb, while keeping the number of tasks and RDs fixed as $K = 8$, and $M = 6$, respectively. It is important to note that the task size at each point is uniformly distributed. For instance, $D_k \in [1, 5]$ Mb for first x-axis value and likewise for rest of the values. The increasing trend of latency for all the schemes is clear from Fig. 6(a) and Fig. 6(b) for Orlando and KAIST datasets, respectively. It is obvious as a larger task would aggravate transmission and remote execution latencies for fixed bandwidth and computing resources. Comparison shows that MATS and GA achieve lowest latency with almost same performance due to location based communication and computation aware scheduling for each task. In contrast, latency provided by SDF and SEF is higher as they perform sequential assignment and ignore the complexity of task which increases with the task size. Specifically, SEF ignores the communication cost which leads to higher transmission latency, and eventually local execution as the communication capacity approaches bottleneck.

In order to determine the operating threshold of different schemes for increasing task size, we plot energy consumption of TD in Fig. 7. In accordance with the increasing latency trends in Fig. 6, the energy consumed by TD gradually increases for all the schemes in Fig. 7(a) and Fig. 7(b) for Orlando and KAIST datasets, respectively. MATS outperforms other baselines up to a certain threshold for both datasets.



(a) Total latency vs RDs (b) Total energy consumed by TD vs RDs (c) Efficiency vs RDs

Fig. 8. Effect of increase in number of RDs on (a) total latency, (b) energy consumption of TD, and (c) efficiency.

It is due to the fact that, the complex tasks are assigned best resources, while the light weight tasks are decided to be executed locally as the task size increases. Beyond the threshold value, GA provides slightly better energy at the expense of minor latency performance degradation. Moreover, GA can uphold the maximum task size while satisfying the energy constraint of TD. Keeping in view the discrepancy between Figs. 7(a) and 7(b) due to the varying mobility patterns, we conclude that MATS can uphold task size of 25 Mb within the expected energy consumption. Meanwhile, SDF and SEF fail to ensure the energy constraint of TD beyond 15 Mb and 10 Mb, respectively.

From Figs. 6 and 7, it is clear that GA can scale up to maximum task size with almost same latency as MATS. However, MATS offer scalability up to 25 Mb with low computational complexity which makes it more suitable for practical scenarios.

In Fig. 8, we analyse the performance in terms of latency, energy consumption of TD, and efficiency by increasing number of RDs. The number of tasks is fixed as $K = 8$. Fig. 8(a) plots latency for different schemes. The performance of local execution remains invariant as it does not utilize the RDs. With increase in RDs, latency gradually decreases for MATS, GA, and SDF, as more feasible offloading options are available for each of these schemes. MATS and GA show similar latency trends with low decreasing rate, where MATS always achieves the lowest latency. The latency for both the schemes reduces by approximately 9% as the number of RDs increase from 2 to 12. For SDF, latency decreases at a higher rate than other schemes as it only selects the RD with the best channel condition, which is more probable to happen as the number of RDs increase. However, it cannot achieve similar performance to MATS and GA even with 12 RDs. It is important to note that, with increase in RDs, the variation in communication capacity is relatively larger than computation capacity, due to mobility-driven uncertainty in channel gains. In this background, we try to explain the inconsistent latency trend exhibited by SEF. SEF selects RDs with best computing resources while ignoring the transmission cost which eventually creates performance bottleneck. It may happen that the selected RD provide lower communication cost leading to better total latency. Nevertheless, the uncertainty in location based data rates may result in higher transmission cost which aggravates the total latency.

Fig. 8(b) plots the corresponding energy consumption of TD for different schemes versus the number of RDs. The energy performance improves for MATS, GA, and SDF, as the probability to find better RDs for offloading increases with the increase in total number of RDs. Comparison among these three baselines shows that, both MATS and GA manage to satisfy the energy constraint even with 2 RDs, while SDF fails to meet the expected energy consumption when number of RDs is less than 6. Once the energy constraint is satisfied for SDF, the performance does not vary at large with increase in RDs beyond 6. GA achieves the lowest energy consumption within the expected value by employing optimal power allocation. On the other hand, MATS performs slightly worse than GA when number of RDs is small. However, its energy performance improves at a higher rate, and eventually becomes comparable to GA as number of RDs exceed beyond 6. It is due to the fact that, MATS exploits the diversity among the RDs to assign the complex tasks first which results in lower communication overhead at the TD. Finally, for SEF the irregular energy trend is consistent with the latency trend exhibited in Fig. 8(a), and the underlying reason is already explained before.

According to the performance on latency and energy consumption of TD, the efficiency of the different schemes is shown in Fig. 8(c). Both MATS and GA provide comparable efficiency. When number of RDs is small, GA provides best efficiency. As the number of RDs exceed beyond 6, the efficiency of MATS approaches GA and eventually becomes 5% higher.

Finally, we provide a comparison of the computational complexity of each scheme running on a computer with Intel Core i5-4200M 2.5 GHz CPU and 16 GB RAM. Table III provides average running time per iteration of each scheme for different number of tasks. The number of RDs is fixed as $M = 10$, while two different trajectory lengths are considered with $N = 5$ and $N = 10$ respectively. As expected the running time increases gradually for all the schemes with increase in number of tasks. Moreover, the computation complexity increases when trajectory length increases from $N = 5$ to $N = 10$ slots since more options need to be explored

TABLE III
RUNNING TIME COMPLEXITY COMPARISON (MILLISECONDS)

Schemes	$N = 5$			$N = 10$		
	$K = 6$	$K = 10$	$K = 14$	$K = 6$	$K = 10$	$K = 14$
GA (x10^3)	130	192	210	367	700	803
MATS	0.47	0.48	2.50	0.79	0.86	1.02
SDF	0.85	1.14	4.60	1.08	1.53	2.37
SEF	0.82	0.96	1.47	1.19	1.46	2.14

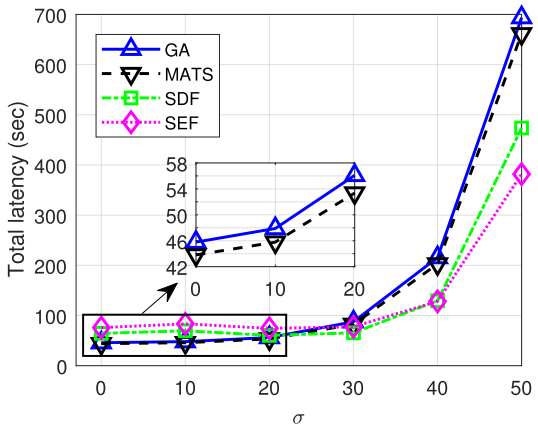


Fig. 9. The total latency of four schemes for different values of prediction error.

along the trajectory. MATS achieves the lowest complexity, while GA has the highest run time complexity among all the schemes. Especially, when the number of slots are doubled, the running time of GA increases manifold which agrees with the time complexity $O(GPKMN)$ expressed in Section V-A. Although GA has much higher computation complexity, it is still suitable and practical to implement for offline task scheduling.

We evaluate the performance of different schemes for several significant aspects, which demonstrates that MATS and GA outperform other baselines. Our analysis provides the operating thresholds for these schemes, for instance, maximum load in terms of number of tasks and task size, and effective number of edge resources. Although GA-based scheme manifests better scalability than MATS for different mobility scenarios, its computational complexity increases directly with the size of the problem. On the other hand, MATS cannot always scale to the maximum loads. However, it provides comparable scalability with relatively lower complexity, which makes it preferable for practical scenarios.

Impact of Trajectory on Performance

Since our proposed scheme relies on predicted trajectories, we analyse the impact of mobility prediction error on total latency in Fig. 9 similar to the work in [41]. For this plot we use Orlando dataset, and fix the number of tasks and RDs as $K = 8$ and $M = 6$, respectively. We introduce the position prediction error $(\Delta x, \Delta y)$ using two dimensional normal distribution with mean 0 and standard deviation σ, where Δx and Δy is the prediction error in X-coordinate and Y-coordinate, respectively. As expected, the latency increases for all the schemes with increase in σ. For $\sigma = 0$ i.e., the original trajectory, it can be observed that both MATS and GA provide comparable latency. Up to $\sigma = 30$, all the schemes show robustness by upholding almost same performance. As the prediction error increases beyond 30, the latency of all the schemes increases exponentially. However, the higher increasing rate for MATS and GA is obvious as their mobility-aware scheduling makes them more sensitive to error. In contrast, SDF and SEF have relatively lower increasing rate. Specifically, SEF relies the least on user mobility, and thus shows more robustness to prediction error.

VII. CONCLUSION

In this work, we investigated mobility-aware task scheduling problem to minimize the latency of an energy-constrained mobile user in a D2D-enabled MEC network. We formulated joint task assignment and power allocation as MINLP problem by assuming predicted trajectory of mobile user. We first solved our problem using GA-based evolutionary scheme. Then we proposed MATS scheme to show that how to assign tasks by taking into account the location based communication and computation capacity of RDs together with task properties. We used real life human walk trajectories in our simulations, which provided some interesting and useful insights relevant to the performance of our proposed schemes. Based on our performance analysis, we conclude that both GA and MATS can perform well in diverse network scenarios. However, MATS provide lower complexity than GA.

In this work, we considered a single-user multi-task scenario and exploited user mobility to enhance the offloading performance. However, it would be challenging to handle multiple users with wide range of varying applications striving for shared resources. In this context, it would be interesting to consider the social behavior/activities of users together with mobility behavior. By jointly exploiting these two aspects for individual users, resource reservation based offloading strategies can be designed to effectively share the communication and computing resources.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[2] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.
[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
[5] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in LTE-advanced networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1923–1940, 4th Quart., 2015.
[6] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.
[7] I. Rhee, M. Shin, S. Hong, K. Lee, S. J. Kim, and S. Chong, "On the levy-walk nature of human mobility," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 630–643, Jun. 2011.
[8] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.
[9] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
[10] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[11] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.
[12] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[13] H. Q. Le, H. Al-Shatri, and A. Klein, "Efficient resource allocation in mobile-edge computation offloading: Completion time minimization," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2513–2517.
[14] J. Ren, G. Yu, Y. Cai, Y. He, and F. Qu, "Partial offloading for latency minimization in mobile-edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.
[15] U. Saleem, Y. Liu, S. Jangsher, and Y. Li, "Performance guaranteed partial offloading for mobile edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[16] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.
[17] S. Zhao, Y. Yang, X. Yang, W. Zhang, X. Luo, and H. Qian, "Online user association and computation offloading for fog-enabled D2D network," in *Proc. IEEE Fog World Congr. (FWC)*, Oct. 2017, pp. 1–6.
[18] J. Feng, L. Zhao, J. Du, X. Chu, and F. R. Yu, "Computation offloading and resource allocation in D2D-enabled mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[19] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[20] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014.
[21] Z. Hong, Z. Wang, W. Cai, and V. C. M. Leung, "Connectivity-aware task outsourcing and scheduling in D2D networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9.
[22] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. IEEE 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Sep. 2016, pp. 1–6.
[23] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 1–8, Aug. 2018.
[24] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.
[25] P. Bellavista, S. Chessa, L. Foschini, L. Gioia, and M. Girolami, "Human-enabled edge computing: Exploiting the crowd as a dynamic extension of mobile edge computing," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 145–155, Jan. 2018.
[26] Y. Niu, Y. Liu, Y. Li, X. Chen, Z. Zhong, and Z. Han, "Device-to-device communications enabled energy efficient multicast scheduling in mmWave small cells," *IEEE Trans. Commun.*, vol. 66, no. 3, pp. 1093–1109, Mar. 2018.
[27] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[28] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 193–200.
[29] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[30] Y. Wu, W. Liu, S. Wang, W. Guo, and X. Chu, "Network coding in device-to-device (D2D) communications underlaying cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 2072–2077.
[31] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.
[32] D. Pisinger, "Where are the hard knapsack problems?" *Comput. Oper. Res.*, vol. 32, no. 9, pp. 2271–2284, Sep. 2005.
[33] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM J. Comput.*, vol. 35, no. 3, pp. 713–728, 2005.
[34] H.-L. Fang, P. Ross, and D. Corne, "A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems," Dept. Artif. Intell., Univ. Edinburgh, Edinburgh, U.K., 1993.
[35] L. Davis, *Handbook of Genetic Algorithms*. New York, NY, USA: Van Nostrand Reinhold, 1991.
[36] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 164–174, Feb. 2018.
[37] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.
[38] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang, "Exploiting joint computation offloading and data caching to enhance mobile terminal performance," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2016, pp. 1–6.
[39] MathWorks. *Mixed Integer Ga Optimization*. [Online]. Available: https://www.mathworks.com/help/gads/mixed-integer-optimization.html.
[40] Y.-D. Lin, E. T.-H. Chu, Y.-C. Lai, and T.-J. Huang, "Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds," *IEEE Syst. J.*, vol. 9, no. 2, pp. 393–405, Jun. 2013.
[41] Y. Liu, X. Chen, Y. Niu, B. Ai, Y. Li, and D. Jin, "Mobility-aware transmission scheduling scheme for millimeter-wave cells," *IEEE Trans. Wireless Commun.*, vol. 17, no. 9, pp. 5991–6004, Sep. 2018.

Umber Saleem received the B.E. degree in information and communication systems engineering and the M.S. degree in electrical engineering (telecommunications) from the National University of Sciences and Technology (NUST), Pakistan, in 2013 and 2016, respectively. She is currently pursuing the Ph.D. degree in electronic engineering with Tsinghua University, China. Her research interest includes resource allocation for mobile edge mobile edge computing.

Yu Liu received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree in electronic engineering. His research interests include wireless networks, edge computing, and optimization.

Sobia Jangsher (Member, IEEE) received the B.E. degree in electronics engineering and the M.S. degree in communication system engineering from the National University of Science and Technology (NUST), Pakistan, and the Ph.D. degree in wireless communication from The University of Hong Kong, Hong Kong. She did her M.S. thesis on "Adaptive transmission of video over MIMO channels" under the supervision of Dr. Syed Ali Khayam and the Ph.D. thesis on "Resource allocation in moving small cell network" under the supervision of Prof. Victor O. K. Li. She is currently working as an Assistant Professor with the Institute of Space Technology, Islamabad, Pakistan. Her research interest includes resource allocation in future wireless communication systems.

Yong Li (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from Tsinghua University in 2012. He is currently a tenured Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include machine learning and big data mining, particularly in, automatic machine learning and spatial–temporal data mining for urban computing, recommender systems, and knowledge graphs. He has published over 100 papers on first-tier international conferences and journals, including KDD, WWW, UbiComp, SIGIR, AAAI, TKDE, and TMC, and his papers have total citations of more than 8300. Among them, ten are ESI Highly Cited Papers in Computer Science, and received five Conference Best Paper (run-up) Awards. He received the 2016 IEEE ComSoc Asia-Pacific Outstanding Young Researchers, the Young Talent Program of China Association for Science and Technology, and the National Youth Talent Support Program. He has served as the general chair, TPC chair, SPC/TPC member for several international workshops and conferences, and he is on the editorial board of two IEEE journals.

Tao Jiang (Fellow, IEEE) received the Ph.D. degree in information and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in April 2004. He is currently a Distinguished Professor with the Wuhan National Laboratory for Optoelectronics and with the School of Electronics Information and Communications, Huazhong University of Science and Technology. From August 2004 to December 2007, he worked with some universities, such as Brunel University and the University of Michigan-Dearborn. He has authored or coauthored over 300 technical papers in major journals and conferences and nine books/chapters in the areas of communications and networks. He served or is serving as the Symposium Technical Program Committee Member of some major IEEE conferences, including INFOCOM, GLOBECOM, and ICC. He was invited to serve as the TPC Symposium Chair for the IEEE GLOBECOM 2013, IEEE WCNC 2013, and ICCC 2013. He is served or serving as an associate editor of some technical journals in communications, including *IEEE Network*, the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE INTERNET OF THINGS JOURNAL, and he is the Associate Editor-in-Chief of *China Communications*.