

RFID: Towards Low Latency and Reliable DAG Task Scheduling over Dynamic Vehicular Clouds

Zhang Liu, *Student Member, IEEE*, Minghui Liwang, *Member, IEEE*, Seyyedali Hosseinalipour, *Member, IEEE*, Huaiyu Dai, *Fellow, IEEE*, Zhibin Gao *Member, IEEE*, Lianfen Huang

Abstract—Vehicular cloud (VC) platforms integrate heterogeneous and distributed resources of moving vehicles to offer timely and cost-effective computing services. However, the dynamic nature of VCs (i.e., limited contact duration among vehicles), caused by vehicles' mobility, poses unique challenges to the execution of computation-intensive applications/tasks with directed acyclic graph (DAG) structure, where each task consists of multiple interdependent components (subtasks). In this paper, we study scheduling of DAG tasks over dynamic VCs, where multiple subtasks of a DAG task are dispersed across vehicles and then processed by cooperatively utilizing vehicles' resources. We formulate DAG task scheduling as a 0-1 integer programming, aiming to minimize the overall task completion time, while ensuring a high execution success rate, which turns out to be NP-hard. To tackle the problem, we develop a ranking and foresight-integrated dynamic scheduling scheme (RFID). RFID consists of (i) a *dynamic downward ranking* mechanism that sorts the scheduling priority of different subtasks, while explicitly taking into account for the sequential execution nature of DAG; (ii) a *resource scarcity-based priority changing* mechanism that overcomes possible performance degradations caused by the volatility of VC resources; and (iii) a *degree-based weighted earliest finish time* mechanism that assigns the subtask with the highest scheduling priority to the vehicle which offers rapid task execution along with reliable transmission links. Our simulation results reveal the effectiveness of our proposed scheme in comparison to benchmark methods.

Index Terms—Vehicular cloud computing, directed acyclic graph, task scheduling, network dynamics, volatile resources.

I. INTRODUCTION

A. Background and Challenges

Rapid development of Internet of Vehicles (IoV) has led to the emergence of diverse vehicular applications (referred to as *tasks*), e.g., advanced driver assistance system, Netflix streaming, and VTube [1]. Many of these tasks are computation-intensive and resource-hungry, requiring a massive amount of computation resources to meet their execution requirements, provisioning of which is often beyond the capability of onboard computation equipment of a single vehicle. One approach to process these tasks is to exploit the vehicle-to-infrastructure (V2I) communications and offload them to

either remote cloud servers [2] or edge computing servers [3]. Nevertheless, V2I connections are not always accessible, e.g., in suburban areas. Also, continuous transfer of data from the vehicles to the cloud servers may incur high traffic congestion on the backhaul links. Furthermore, edge computing servers may not have enough computation resources to satisfy resource demands of a large number of mobile vehicles.

To address the aforementioned limitations, vehicular cloud (VC) [4]–[6] has emerged as an effective computing paradigm, which exploits the dynamic and distributed computation and communication resources of vehicles to provide responsive and cost-effective computing services. Specifically, VC orchestrates the heterogeneous resources of vehicles in geographic proximity and exploits opportunistic vehicle-to-vehicle (V2V) communications to build flexible and yet scaleable topologies for provisioning of computing services.

One of the key advantages of the scalable architecture of VC is its potential to process computation intensive tasks. These tasks are often represented via a directed acyclic graph (DAG) [11]–[14], where the task is partitioned into interdependent components (referred to as *subtasks*) and the processing intricacies between subtasks is captured via introducing a topological structure among them.

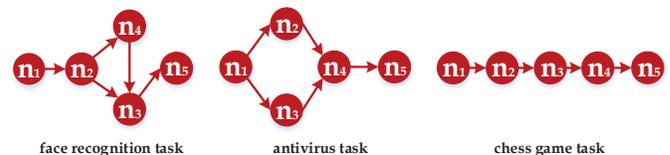


Fig. 1. Examples of DAG-represented tasks, and the corresponding interdependence among subtasks [22].

Fig. 1 illustrates some examples of DAG tasks, and the corresponding interdependencies among their subtasks [22]. Take face recognition task as an example, which can be generated by smart buses to trace the contacts of patients who are tested positive during the COVID-19 epidemic [32]. In a face recognition task, each subtask (vertex) denotes one part of the process of face recognition, while the edges represent the corresponding interdependencies, such as outline and color information. The execution of a DAG task should be conducted in an ordered manner since processing a subtask potentially needs the output data of others (e.g., in the face recognition task in Fig. 1, the processing of subtask n_2 relies on the output

Zhang Liu (zhangliu@stu.xmu.edu.cn), Minghui Liwang (minghuilw@xmu.edu.cn), Zhibin Gao (gaozhibin@xmu.edu.cn) and Lianfen Huang (lfhuang@xmu.edu.cn) are with the Department of Information and Communication Engineering, School of Informatics, Xiamen University, Fujian, China. S. Hosseinalipour (alipour@buffalo.edu) is with the Department of Electrical Engineering, University at Buffalo, SUNY, Buffalo, NY, USA. Huaiyu Dai (hdai@ncsu.edu) is with Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA. (Corresponding author: Minghui Liwang).

data of subtask n_1 and the processing of subtask n_3 relies on the output data of both n_2 and n_4).

Upon execution of a DAG task over a VC, its subtasks can be scheduled and processed by different vehicles in a cooperative manner via V2V communication links. However, there still remains noteworthy issues when scheduling DAG tasks over VCs. First, the heterogeneity of vehicles' processing capabilities (e.g., different computation and communication resources) leads to the non-triviality of determining the scheduling priority of each subtask. Second, dynamic and volatile VC topology leads to time-varying availability of computation resources (e.g., vehicles may arrive at or depart from the VC while a DAG task is being processed), which further adds to the difficulty of subtask ranking and scheduling. Third, given the sequential execution procedure of subtasks within a DAG task, failure in the processing of a single subtask, caused by vehicles' mobility (e.g., upon completion of a subtask, there are no feasible vehicles to execute the subsequent subtasks due to the constrained V2V connections), results in the failure of the entire DAG task. These issues make DAG task scheduling over dynamic VC a challenging problem, which should be carefully investigated while taking into account for the unique characteristics of both the VC and the DAG task structure.

B. Overview and Summary of Contributions

This paper investigates DAG task scheduling over VC aiming to minimize the overall DAG task completion time (i.e., low latency), while ensuring a high execution success rate (i.e., high reliability). We formulate the DAG task scheduling problem, while explicitly taking into account for: *i*) the *heterogeneity* of computation and communication resources of different vehicles (referred to as resource providers), *ii*) *dynamics* and *volatility* of VC's topology, and *iii*) the *sequential execution* nature of a DAG task imposed by the interdependencies among its subtasks. We introduce the unified framework of ranking and foresight-integrated dynamic scheduling scheme (RFID), which aims to minimize the overall completion time of DAG task, while ensuring a commendable probability of successful task execution.

Our major contributions can be summarized as follows:

- To the best of our knowledge, this paper is among the first to address minimizing DAG task completion time under task execution success rate guarantee over dynamic VC. This is achieved via considering the sequential execution order of subtasks within a DAG task, while capturing the dynamics of VC through a time-varying graph.
- We formulate the VC-assisted DAG task scheduling problem as an integer programming, aiming to minimize the overall DAG task completion time while ensuring high execution success rate upon considering dynamics and resource heterogeneity of VC, which turns out to be NP-hard.
- To tackle the problem, we propose a dynamic scheduling algorithm over VC, called RFID. RFID first recursively determines the scheduling priority of different subtasks

according to the assignment of their immediate predecessors. It then selectively changes the scheduling priority of a fraction of subtasks according to the availability of vehicles' resources. Finally, RFID selects the vehicles for subtask assignment based on the vehicles' connectivity and resources.

- We implement RFID over real-world traffic data obtained from the OpenStreetMap [30]. We further leverage SUMO [31] simulation platform to simulate the environment and demonstrate the effectiveness of RFID. The simulation results reveal that RFID outperforms benchmark methods in terms of DAG task completion time and execution success rate, while enjoying a relatively low computation complexity.

The rest of this paper is organized as follows: Section II discusses related works on task scheduling over different network architectures. In Section III, we present the system model and formulate the VC-assisted DAG task scheduling problem. In Section IV, we introduce RFID. Simulation results are presented in Section V and the work is concluded in Section VI.

II. RELATED WORK

Existing works devoted to task scheduling/offloading over cloud-based networks can be roughly divided into three categories with respect to their task model: *i*) tasks represented by indivisible bit streams with no interdependent subtasks, such as [7]–[10]; *ii*) tasks represented via *undirected* graphs considering interdependencies among subtasks, such as [24]–[27], where the subtasks can be offloaded simultaneously and processed on different servers in parallel; *iii*) tasks that are modeled as DAG which further require explicit processing order across their subtasks, e.g., [23], [11]–[14]. In the following, we discuss the contributions of these works and highlight the differences between the scenario considered in this paper and prior works.

A. Scheduling of Bit Stream Tasks

X. Chen *et al.* in [7] studied computation offloading of bit stream tasks in a mobile edge computing (MEC) network, by formulating a multi-user computation offloading game, while achieving Nash equilibrium. In [8], Y. Mao *et al.* addressed computation offloading of bit stream tasks in MEC with energy harvesting devices via proposing a Lyapunov-based algorithm. S. Bi *et al.* in [9] studied the computation rate maximization of bit stream tasks in wireless powered MEC through a bisection search algorithm and a coordinate descent method. In [10], H. Guo *et al.* formulated the MEC offloading problem for bit stream tasks in ultra-dense wireless networks using a two-tiered game-theoretic task offloading scheme. Although the aforementioned works provide useful insights on task scheduling, none of them considers execution of computation intensive tasks, which can be partitioned into multiple subtasks across the computing resources.

B. Scheduling of Undirected Graph (UG) Tasks

For UG tasks, J. Ghaderi *et al.* in [24] proposed a randomized task scheduling algorithm under stochastic task arrival/departure. L. Shi *et al.* in [25] studied the energy-aware scheduling problem for parallel tasks in cloud by designing a time-efficient scheduling algorithm called TaPRA. In [26] and [27], M. Liwang *et al.* focused on the allocation of computation-intensive graph tasks in IoV and proposed subgraph isomorphism extraction-based low complexity mechanisms. However, UG tasks do not require any specific processing order among their subtasks, where all the subtasks of a UG task can be executed in parallel across the computing resources. As a result, this makes their allocation mechanism different than DAG tasks.

C. Scheduling of DAG Tasks over Static Networks

DAG task scheduling has been extensively studied in static MEC networks with fully connected servers. H. Topcuoglu *et al.* in [23] proposed the HEFT algorithm, where each subtask is assigned to the processor that can minimize its corresponding completion time. In [11], L. F. Bittencourt *et al.* utilized forward looking attribution to improve the performance of HEFT. M. Aggarwal *et al.* in [15] developed a genetic algorithm for DAG task scheduling. In [14], H. Kanemitsu *et al.* proposed a clustering-based DAG task scheduling algorithm focusing on assigning the subtasks which are located on the critical path to the same processor. G. C. Sih *et al.* in [12] adopted a dynamic scheduling algorithm, where a global time clock is used to regulate the scheduling process. Recently, in [16], Y. Sahni *et al.* introduced a JDOFH algorithm to schedule multiple DAG tasks in a multi-hop collaborative edge computing environment. However, the aforementioned works mainly focus on static networks, ignoring the dynamics and instability of service provisioning, which are significant features of VCs.

D. Scheduling of DAG Tasks over Dynamic Networks

There exist few recent works dedicated to DAG task scheduling over dynamic networks. F. Sun *et al.* in [18] addressed cooperative DAG task scheduling in VC to reduce the overall task completion time via implementing a modified genetic algorithm. In [19], H. Liu proposed a policy gradient-based offloading scheme for minimizing the overall DAG task completion time in vehicular networks. In our previous work [20], we studied topology-aware DAG task allocation in vehicular networks and proposed a simulated annealing-based task allocation algorithm.

Although the aforementioned works take significant steps toward DAG task scheduling in dynamic networks, they suffer from several limitations, which we aim to address in this work. In particular, in [18], scheduling time slots were defined to include the execution and data transmission process, which, however, created redundancy in the task completion time. Also, short V2V communication path life time was not considered in [19]. Moreover, in our previous work [20], the topology of IoV is assumed to remain unchanged during the

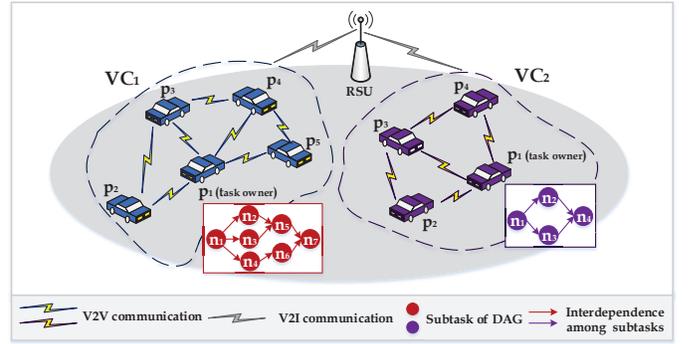


Fig. 2. The framework of the proposed VC-assisted DAG task scheduling.

completion of subtasks. Thus, we are motivated to develop a dynamic scheduling scheme with reasonable computation complexity for DAG task scheduling over VC with volatile V2V links.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We model each DAG task A as a graph $\mathcal{G}^A = (\mathcal{V}^A, \mathcal{E}^A)$, where vertex set $\mathcal{V}^A = \{n_1, n_2, \dots, n_{|\mathcal{V}^A|}\}$ represents the set of subtasks with $|\mathcal{V}^A|$ denoting the number of subtasks in A and \mathcal{E}^A denoting the edge set. Each $e_{n_i, n_j} \in \mathcal{E}^A$ is a directed edge describing the corresponding precedence, indicating subtask n_i has to be completed before the execution of n_j (i.e., the output of n_i is used as the input data for n_j). A subtask without any predecessors represents the *entry* subtask, denoted by n_{entry} , (e.g., subtask n_1 in Fig. 1); while a subtask without any successors indicates the *exit* subtask, denoted by n_{exit} , (e.g., subtask n_5 in Fig. 1). If there is more than one *entry/exit* subtask, they are assumed to be connected to a virtual *entry/exit* subtask with edges that entail zero data exchange requirements, for analytical simplicity. Fig. 2 depicts a schematic of our system model and some examples, where two VCs are managed by a road side unit (RSU) as a centralized controller¹. Major notations used in this paper are summarized by Table I.

We model the dynamic topology of VC as a time-varying undirected graph $\mathcal{G}^{\text{VC}}(t) = (\mathcal{V}^{\text{VC}}(t), \mathcal{E}^{\text{VC}}(t))$. Specifically, the set $\mathcal{V}^{\text{VC}}(t) = \{p_1, p_2, \dots, p_{|\mathcal{V}^{\text{VC}}(t)|}\}$ contains the task owner p_1 (i.e., the vehicle who generates the DAG task) and other service vehicles in the network at time t ; while $\mathcal{E}^{\text{VC}}(t)$ represents the corresponding edge set where each $e_{p_m, p_n}^{\text{VC}}(t) \in \mathcal{E}^{\text{VC}}(t)$ stands for a one-hop two-way V2V link between two vehicles p_m and p_n , where $p_n, p_m \in \mathcal{V}^{\text{VC}}(t)$. The existence of an edge between two vehicles is a result of their corresponding distance as modeled in Section III-B. Vehicles are assumed to have heterogeneous computation capabilities modeled via different local CPU processing speed, denoted by f_{p_m} (cycles/s) for

¹This paper studies the task scheduling problem via considering one DAG task (generated by task owner) and one VC for analytical simplicity. Our proposed algorithm can also be well applied in networks with multiple VCs and DAGs, e.g., two DAGs in one VC can be seen as a virtual big DAG. Cooperation among VCs and competing for limited resources between multiple DAG tasks are left as our future work.

TABLE I
MAJOR NOTATIONS

Notations	Explanation
$\mathcal{G}^{\mathbf{A}} = (\mathcal{V}^{\mathbf{A}}, \mathcal{E}^{\mathbf{A}})$	DAG task model, where $\mathcal{V}^{\mathbf{A}}$ is the set of subtasks and $\mathcal{E}^{\mathbf{A}}$ is the set of directed edges
$\mathcal{G}^{\mathbf{VC}}(t) = (\mathcal{V}^{\mathbf{VC}}(t), \mathcal{E}^{\mathbf{VC}}(t))$	VC model, where $\mathcal{V}^{\mathbf{VC}}(t)$ contains the vehicles in network at time t and $\mathcal{E}^{\mathbf{VC}}(t)$ contains the corresponding one-hop V2V links among vehicles
$\text{TT}_{n_i, n_j}(p_m, p_n)$	The data transmission time from vehicle p_m (assigned to subtask n_i) to vehicle p_n (processing subtask n_j)
$\text{CT}(n_i, p_m)$	The computation time of processing subtask n_i on vehicle p_m
ξ_{n_i, p_m}	A binary variable indicating the assignment of subtask n_i to vehicle p_m
st_{n_i}	The scheduling time of subtask n_i
$\text{pred}(n_i)$	The immediate predecessor set of subtask n_i
$\text{succ}(n_i)$	The immediate successor set of subtask n_i
$\text{RT}(n_i, p_m)$	The ready time of processing subtask n_i on vehicle p_m
$\text{EST}(n_i, p_m)$	The earliest execution start time of subtask n_i on vehicle p_m
$\text{EFT}(n_i, p_m)$	The earliest execution finish time of subtask n_i on vehicle p_m
$\text{AFT}(n_i)$	The actual finish time of subtask n_i when n_i is practically processed on a specific vehicle
$\mathcal{C}^{\mathbf{VC}}(n_i)$	The candidate vehicle set for scheduling subtask n_i
$\mathbf{n}_{\text{ready}}$	The time-varying ready subtask set
$\mathcal{D}^{\mathbf{VC}}(n_i, p_m)$	The degree-based vehicle set when subtask n_i is processed on vehicle p_m
$\text{EFT}^{\mathbf{W}}(n_i, p_m)$	The weighted earliest finish time of subtask n_i on vehicle p_m
n_i, n_j	Indices used to represent subtasks
p_m, p_n	Indices used to represent vehicles

vehicle p_m . They also are assumed to execute one subtask at a time [16], where multiple subtasks assigned to one vehicle may have to wait for resource release.

A. Communication Model

Considering transmitting the processing results of subtask n_i (executed on vehicle p_m) to vehicle p_n which is assigned to execute subtask n_j , where $e_{n_i, n_j}^{\mathbf{A}} \in \mathcal{E}^{\mathbf{A}}$, a dual-slope (power-law) model [33] is leveraged to formulate the propagation loss of the underlying V2V communication link in dB, which is considered to be full-duplex as follows:

$$\text{PL}(d_{p_m, p_n}(t)) = L_b + \begin{cases} 10\eta_1 \log(d_{p_m, p_n}(t) + \text{PL}(d_0)), & \text{if } 1 \leq d_{p_m, p_n}(t) \leq d_{\text{brk}} \\ 10(\eta_1 - \eta_2) \log(d_{\text{brk}}) + & \\ 10\eta_2 \log(d_{p_m, p_n}(t) + \text{PL}(d_0)), & \text{if } d_{p_m, p_n}(t) > d_{\text{brk}} \end{cases} \quad (1)$$

where L_b is a basic transmission-loss parameter that depends on the frequency and the antenna height. $d_{p_m, p_n}(t)$ indicates the Euclidean distance between vehicle p_m and p_n at time t , $d_0 = 1$ (m) is the reference distance, $\eta_1 = 2$ and $\eta_2 \in [2, 7]$ denote the slopes of the best-fit line before and after distance d_{brk} , respectively, and d_{brk} indicates the breakpoint distance given by

$$d_{\text{brk}} = \frac{4h_t h_r}{\delta} - \frac{\lambda}{4}, \quad (2)$$

where h_t and h_r are the height of transmitter (i.e., the antenna on vehicle p_m) and the receiver (i.e., the antenna on vehicle p_n), δ represents the signal power fluctuations due to surrounding objects, and λ denotes the wavelength [36].

Let the binary indicator variable ξ_{n_i, p_m} describe the subtask assignment: $\xi_{n_i, p_m} = 1$ if subtask n_i is scheduled on vehicle

p_m ; otherwise $\xi_{n_i, p_m} = 0$. The transmission time (s) associated with data transmission over the edge $e_{n_i, n_j}^{\mathbf{A}} \in \mathcal{E}^{\mathbf{A}}$, when $\xi_{n_i, p_m} \times \xi_{n_j, p_n} = 1$ at time t is given by

$$\text{TT}_{n_i, n_j}(p_m, p_n) = \begin{cases} c_{n_i, n_j} \times \Gamma(\text{PL}(d_{p_m, p_n}(t))), & p_m \neq p_n \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where c_{n_i, n_j} denotes the size (bit) of output data of subtask n_i , which needs to be transferred to its dependent task n_j , and $\Gamma(\cdot)$ is a monotone increasing function expressed in [34]. Due to vehicles' mobility and their limited contact durations, this paper only considers one-hop data transmission between the vehicles [4]–[6].

B. V2V Contact Model

A contact event between vehicles p_m and p_n can happen when $d_{p_m, p_n}(t) \leq R$ at time t , where R represents the vehicular communication radius. The contact duration between p_m and p_n is assumed to obey an exponential distribution [35] with parameter μ_{p_m, p_n} . Correspondingly, the probability that the residual contact duration denoted by $|e_{p_m, p_n}^{\mathbf{VC}}(t)|$ between vehicle p_m and p_n at time t is larger than T is given by

$$\Pr(|e_{p_m, p_n}^{\mathbf{VC}}(t)| > T) = \exp(-T\mu_{p_m, p_n}). \quad (4)$$

According to (4), a lower value of the required contact duration T for data transmission can be provided with a higher assurance.

C. Computation Model

Let $\text{EST}(n_i, p_m)$ and $\text{EFT}(n_i, p_m)$ denote the earliest execution start time of subtask n_i on p_m , and the earliest execution finish time of subtask n_i on vehicle p_m , respectively.

For the entry subtask, n_{entry} is assumed to be executed on the task owner p_1 , and thus $\text{EST}(n_{\text{entry}}, p_1) = 0$.

Next, we provide key definitions used in developing our methodology.

Definition 1. (Ready time). The ready time $\text{RT}(n_i, p_m)$ is time when all the immediate predecessor subtasks of n_i have been completed, while the required input data for processing n_i has arrived at vehicle p_m , which is given by

$$\text{RT}(n_i, p_m) = \max_{\substack{n_j \in \text{pred}(n_i), \\ \xi_{n_j, p_n} = 1}} \{ \text{AFT}(n_j) + \text{TT}_{n_j, n_i}(p_n, p_m) \}, \quad (5)$$

where $\text{pred}(n_i)$ is the set of immediate predecessor subtasks of n_i and $\text{AFT}(n_j)$ is the actual finish time of n_j when it is practically scheduled on a specific vehicle:

$$\text{AFT}(n_j) = \text{EFT}(n_j, p_m), \text{ where } \xi_{n_j, p_m} = 1. \quad (6)$$

Definition 2. (Scheduling time). The scheduling time of subtask n_i , i.e., st_{n_i} is the earliest time to allocate n_i to a vehicle, which is given by

$$\text{st}_{n_i} = \max_{n_j \in \text{pred}(n_i)} \{ \text{AFT}(n_j) \}, \quad (7)$$

Where, $\text{pred}(n_i)$ is the set of immediate predecessors of subtask n_i . Using Definition 1, for the other subtasks involved in the DAG, the values of EFT and EST can be computed recursively, starting from the entry subtask as follows:

$$\text{EST}(n_i, p_m) = \max \{ \text{Avail}(n_i, p_m), \text{RT}(n_i, p_m) \}, \quad (8)$$

$$\text{EFT}(n_i, p_m) = \text{CT}(n_i, p_m) + \text{EST}(n_i, p_m), \quad (9)$$

where $\text{Avail}(n_i, p_m)$ represents the time, in which vehicle p_m completes its last assigned subtask prior to the execution of n_i and it is ready to process a new subtask. Also, $\text{CT}(n_i, p_m)$ denotes the computation time of processing subtask n_i on vehicle p_m , which is given by

$$\text{CT}(n_i, p_m) = w_{n_i} / f_{p_m}, \quad (10)$$

where w_{n_i} represents the required computing workload (*cpu clock cycles*) of subtask n_i .

Since there can be multiple *exit* subtasks, after all the subtasks in a DAG are scheduled, the actual finish time of the *exit* subtask n_{exit} is defined as the overall DAG task completion time, which is given by

$$\text{OTC} = \text{AFT}(n_{\text{exit}}). \quad (11)$$

The main goal of DAG task scheduling is to determine the assignment of each subtask to the appropriate vehicle such that the overall DAG task completion time can be minimized.

D. Problem Formulation

We formulate DAG task scheduling over dynamic VC as the following optimization problem:

$$\mathcal{P} : \arg \min_{\{ \xi_{n_i, p_m} \}, n_i \in \mathcal{V}^A, p_m \in \mathcal{V}^{\text{VC}}(\text{st}_{n_i})} \text{OTC} \quad (12)$$

$$\text{s.t. C1: } \sum_{p_m \in \mathcal{V}^{\text{VC}}(\text{st}_{n_i})} \xi_{n_i, p_m} = 1,$$

$$\text{C2: } \text{EST}(n_i, p_m) \geq \max_{n_j \in \text{pred}(n_i)} \{ \text{AFT}(n_j) \}, \xi_{n_i, p_m} = 1,$$

$$\text{C3: } \exp(-\text{TT}_{n_j, n_i}(p_n, p_m) \mu_{p_n, p_m}) \geq \theta,$$

$$\forall n_j \in \text{pred}(n_i), \text{ and } \xi_{n_i, p_m} \times \xi_{n_j, p_n} = 1,$$

$$\text{C4: } \xi_{n_i, p_m} \in \{0, 1\}.$$

In problem \mathcal{P} , **C1** guarantees that each subtask n_i can be assigned to only one vehicle, **C2** represents that execution of a subtask can not start until all its predecessor subtasks are completed based on the sequential execution property of DAG task. Considering the volatility of V2V links modeled in Section III-B, **C3** ensures that vehicle p_m , scheduled to execute subtask n_i , can successfully receive input data of n_i , where θ in **C3** is the predefined quality of service factor. Besides, **C4** restricts that the scheduling variables are binary.

\mathcal{P} is a 0-1 integer programming problem which is NP-hard. This makes finding time-efficient algorithms to solve the problem difficult, especially in large-scale dynamic networks. Also, solving \mathcal{P} requires determining the scheduling priority of each subtask to preserve the sequential processing requirements imposed by the DAG structure in volatile VC environment, which is challenging since during the execution of subtasks the links among the vehicles may begin to form or vanish. Furthermore, since the completion of subtask n_{exit} is considered as the objective function, if the execution of any intermediate subtasks fails (e.g., there are no feasible vehicles for processing a subtask due to the constrained V2V connections, i.e., **C3** can not be satisfied), the execution of entire DAG task will encounter failure. Motivated by these challenges, we develop RFID, which aims to reduce the completion time of task execution, while providing reliability assurance. RFID will enjoy a polynomial time complexity, which makes it suitable for implementation over large-scale dynamic VC.

IV. RANKING AND FORESIGHT-INTEGRATED DYNAMIC (RFID) TASK SCHEDULING

To tackle \mathcal{P} , we develop RFID which is a unified DAG task scheduling methodology over dynamic networks. RFID conducts task scheduling through three phases. In phase I, a *dynamic downward ranking* mechanism is designed to sort the scheduling priority of different subtasks over dynamic VC, via considering the assignment of their immediate predecessor subtasks, to capture DAG task's sequential execution nature. In phase II, a *resource scarcity-based priority changing* mechanism is developed to overcome possible performance degradations caused by the volatility of VC resources via modifying the scheduling priority of a fraction of subtasks obtained in phase I. Finally, in phase III, a *degree-based weighted earliest finish time* mechanism is introduced to assign the subtask with the highest scheduling priority to the vehicle which offers rapid task execution, which also possesses reliable V2V links. A flow chart of RFID, and the inter-relationship between the above-mentioned three phases is shown by Fig. 3. Before

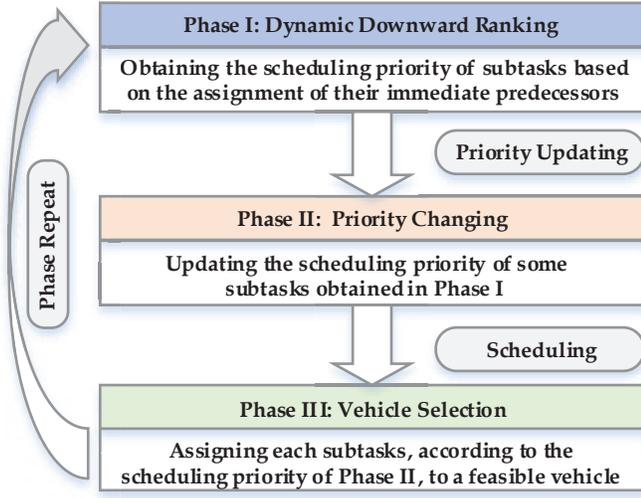


Fig. 3. A flow chart of RFID and the inter-relationship between different phases.

discussing RFID, we first present the definition of *candidate set*, which is considered as a significant part to develop RFID.

Definition 3. (Candidate set). The candidate set $\mathcal{C}^{\text{VC}}(n_i)$ of n_i contains the vehicles that can receive the output data of all the predecessors of n_i , as given by

$$\mathcal{C}^{\text{VC}}(n_i) \triangleq \{p_m : \exp(-\overline{\text{TT}}_{n_j, n_i}(p_n, p_m) \mu_{p_n, p_m}) \geq \theta, p_m \in \mathcal{V}^{\text{VC}}(st_{n_i}), n_j \in \text{pred}(n_i), \xi_{n_j, p_n} = 1\}, \quad (13)$$

where $\text{pred}(n_i)$ is the set of immediate predecessors of n_i , and θ represents the predefined quality of service factor, as defined in C3.

In the following, details of the three phases of RFID are discussed. In each phase, we first highlight the shortcomings of the current state-of-the-art methods, and then, develop our methodology.

A. Phase I: Dynamic Downward Ranking

The major goal of this phase is to determine the scheduling priority of subtasks, based on a metric called *ranking*, where the subtask with a lower rank is considered to have a higher scheduling priority.

1) *Motivation:* Existing methods such as [11]–[13], [16] focus on subtask ranking in static environments, e.g., MEC networks, with multiple fully-connected processors. They determine the scheduling of subtasks based on *downward ranking*, which can be obtained recursively by traversing the DAG downward (i.e., starting from n_{entry} to n_{exit}). For n_{entry} , it is assumed that $\text{rank}(n_{\text{entry}}) = 0$, while for the other subtasks, we have the following (14)

$$\text{rank}(n_i) = \max_{n_j \in \text{pred}(n_i)} \{\text{rank}(n_j) + \overline{\text{CT}}_{n_i} + \overline{\text{TT}}_{n_j, n_i}\}, \quad (14)$$

where $\text{pred}(n_i)$ denotes the set of immediate predecessors of subtask n_i , $\overline{\text{CT}}_{n_i} = \sum_{m=1}^q \text{CT}(n_i, p_m) / q$ is the average computation time of subtask n_i across q static processors,

and $\overline{\text{TT}}_{n_j, n_i} = c_{n_i, n_j} / \bar{B}$ is the average data transmission time associated with edge e_{n_j, n_i}^{A} . \bar{B} denotes the average transmission rate among q static processors.

Since conventional *downward ranking* mainly focuses on static networks (e.g., q and \bar{B} are constants), which, however, does not hold in dynamic VC. To further reveal that existing strategies are difficult to be implemented in our problem setting, we depict a simple example as shown in Fig. 4, where two vehicles are connected by an edge when the distance between them are smaller than the vehicular communication radius. Due to the mobility and the heterogeneous resources of vehicles, the corresponding candidates for scheduling n_2 can be different according to different assignments of n_1 . For example, in Fig. 4, when n_1 is processed on p_1 , we have $\mathcal{C}^{\text{VC}}(n_2) = \{p_1, p_2, p_4, p_5\}$ (although there exists an edge between p_1 and p_3 , p_3 is excluded in the candidate set due to $\Pr(|e_{p_1, p_3}^{\text{VC}}(t)| > \overline{\text{TT}}_{n_1, n_2}(p_1, p_3)) < \theta$). However, when n_1 is processed on p_3 , we have $\mathcal{C}^{\text{VC}}(n_2) = \{p_2, p_3, p_4\}$. Note that the topology of VC changes in the two scenarios considered in Fig. 4, since the processing time of n_1 on p_1 might be different with that on p_3 , which thus results in different values of q and \bar{B} .

As a result, the set of candidate vehicles for processing n_i , can be impacted by different assignments of n_j ($n_j \in \text{pred}(n_i)$) due to dynamics, which should be carefully considered during the subtask ranking mechanism design.

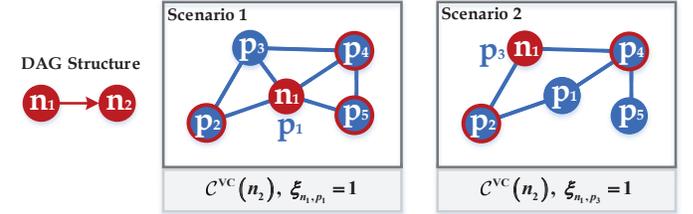


Fig. 4. An example of showing the necessity of using dynamic ranking to determine the priority of each subtask over VC.

2) *Proposed Dynamic Downward Ranking Mechanism:* To address the aforementioned challenges, we propose a *dynamic downward ranking* mechanism, which is tightly coupled with the vehicle selection method described in phase III. We first provide the definition of a *ready subtask*.

Definition 4. (Ready subtask). The set of ready subtask² $\mathbf{n}_{\text{ready}}$ contains the subtasks whose immediate predecessors have already been scheduled, and the corresponding execution order is not bounded by a precedence constraint, i.e., $\forall n_i, n_j \in \mathbf{n}_{\text{ready}}, e_{n_i, n_j}^{\text{A}} \notin \mathcal{E}^{\text{A}}$.

We next model the *dynamic downward ranking* Rank^{D} of each subtask $n_i \in \mathbf{n}_{\text{ready}}$, by leveraging available resources and topological information of VC. In particular, the value of $\text{Rank}^{\text{D}}(n_i)$ can be different according to different assignments of its immediate predecessor subtasks.

²Although $\mathbf{n}_{\text{ready}}$ is time varying, for notational simplicity, we consider $\mathbf{n}_{\text{ready}}$ only changes during the algorithm execution.

For the entry subtask, we have $\text{Rank}^D(n_{\text{entry}}) = 0$. After n_{entry} is scheduled on task owner (according to our basic assumption in previous sections), we determine the *ranking* of the existing subtask $n_i \in \mathbf{n}_{\text{ready}}$ and subsequently assign them to appropriate vehicles (based on the vehicle selection method, which will be described in phase III). To this end, we compute the value of *dynamic downward ranking* as the following (15),

$$\text{Rank}^D(n_i) = \max_{n_j \in \text{pred}(n_i)} \{ \text{Rank}^D(n_j) + \overline{\text{CT}}_{n_i}^D + \overline{\text{TT}}_{n_j, n_i}^D \},$$

$$n_i \in \mathbf{n}_{\text{ready}}, \quad (15)$$

where $\text{pred}(n_i)$ is the set of immediate predecessors of subtask n_i , $\overline{\text{CT}}_{n_i}^D$ is the dynamic average computation time, impacted by the assignment of $n_j \in \text{pred}(n_i)$, as given by

$$\overline{\text{CT}}_{n_i}^D = \frac{\sum_{p_m \in \mathcal{C}^{\text{VC}}(n_i)} \text{CT}(n_i, p_m)}{|\mathcal{C}^{\text{VC}}(n_i)|}, \quad n_i \in \mathbf{n}_{\text{ready}}. \quad (16)$$

Also in (15), $\overline{\text{TT}}_{n_j, n_i}^D$ represents the dynamic average data transmission time, affected by the assignment of $n_j \in \text{pred}(n_i)$, which is shown by

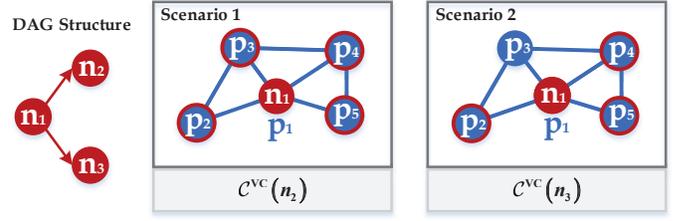
$$\overline{\text{TT}}_{n_j, n_i}^D = \frac{\sum_{p_m \in \mathcal{C}^{\text{VC}}(n_i)} \text{TT}_{n_j, n_i}(p_m, p_m)}{|\mathcal{C}^{\text{VC}}(n_i)|}, \quad n_i \in \mathbf{n}_{\text{ready}}. \quad (17)$$

As a result, different from static network environments, our *dynamic downward ranking* mechanism identifies the suitability of subtask assignment to different vehicles, which will interactively work with the vehicle selection method described in phase III, until the exit subtask has been scheduled.

B. Phase II: Resource Scarcity-based Priority Changing

The goal of this phase is to adjust the scheduling priority of a fraction of subtasks obtained in phase I aiming to overcome possible performance degradations caused by the volatility of VC resources.

1) *Motivation*: Different from static computing environments with stable fully-connected computing servers [11]–[13], dynamics and instability of VC resources can lead to resource scarcity, which leaves heavy impacts on the execution of dependent subtasks. To illustrate this, we depict a simple example in Fig. 5, where after the assignment of n_1 , subtasks n_2 and n_3 become *ready subtask* according to Definition 4. Suppose that $\text{Rank}^D(n_2) < \text{Rank}^D(n_3)$, subtask n_2 will be firstly scheduled to its most preferred vehicle p_1 . However, this case can incur a large completion time increasing, if n_3 is not assigned to p_1 , because p_3 can execute n_2 almost as rapidly. Conversely, since the scale of candidate set for processing n_3 is smaller (e.g., p_3 is infeasible for processing n_3 in Fig. 5), this can raise the completion time when n_3 is not assigned to p_1 , as p_1 is the only vehicle which can process n_1 quickly. Similar situations can be incurred when more than one *ready subtasks* with different transmission requirements compete for a same vehicle, over dynamic VC with volatile resources.



THE ESTIMATED FINISH TIME FOR SUBTASK n_1 AND n_2

	p_1	p_2	p_3	p_4	p_5
n_2	4	8	5	8	8
n_3	5	9	inf.	9	9

Fig. 5. An example showing the necessity of considering resource scarcity in determining the scheduling priority of some specific subtasks over VC.

2) *Proposed Resource Scarcity-based Priority Changing Mechanism*: Motivated by the aforementioned example, we consider dynamic resource availability caused by vehicles' mobility and address the resource scarcity problem via introducing a new metric denoted by $\text{CTI}(n_i)$, which measures the completion time increment imposed by the case where subtask n_i is not scheduled to its most preferred vehicle, defined as

$$\text{CTI}(n_i) = \text{EFT}(n_i, p_m^*) - \min_{p_m \neq p_m^*} \{ \text{EFT}(n_i, p_m) \},$$

$$p_m, p_m^* \in \mathcal{C}^{\text{VC}}(n_i), \quad (18)$$

where p_m^* represents the most preferred vehicle with a minimum value of EFT. Specifically, the minimization term in (18) captures the earliest finish time of having n_i to be executed on the second preferred vehicle.

Consequently, we define the resource-scarcity-based dynamic downward ranking $\text{RSRank}^D(n_i)$, to determine the scheduling priority of each subtask $n_i \in \mathbf{n}_{\text{ready}}$ as follows:

$$\text{RSRank}^D(n_i) = \text{Rank}^D(n_i) - \text{CTI}(n_i), \quad n_i \in \mathbf{n}_{\text{ready}}. \quad (19)$$

As a result, considering the extra completion time increment in assigning each *ready subtask* to the second preferred vehicle (in terms of completion time), can generally bring a change of scheduling priority. For example, in Fig. 5, $\text{Rank}^D(n_2) < \text{Rank}^D(n_3)$ while $\text{RSRank}^D(n_3) < \text{RSRank}^D(n_2)$, and thus RFID schedules n_3 earlier to avoid a large completion time increment.

C. Phase III: Degree-based Weighted EFT for Vehicle Selection

Based on the scheduling priority obtained in Phase II, the goal of this phase is to assign each subtask to the vehicle which can rapidly execute it and have reliable transmission V2V links, which is necessary to transmit the data required for the processing of its successor subtasks.

1) *Motivation*: Heterogeneous earliest finish time algorithm (HEFT) [23], which assigns n_i to p_m to minimize $\text{EFT}(n_i, p_m)$ is widely used in static networks. However, this algorithm is not practical in dynamic VC. We demonstrate this via a simple example shown in Fig. 6, where

the topology of VC changes across three scenarios, since the processing and transmission time of different vehicles can be different, and we have the scheduling order of each subtask as $n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_4$. After n_1 (n_{entry}) is scheduled on p_1 (task owner), n_2 and n_3 are executed on p_2 and p_4 , respectively, according to the corresponding EFT associated with different vehicles, under HEFT. However, it can be seen that the execution of subtask n_4 fails since $\mathcal{C}^{\text{VC}}(n_4) = \emptyset$. Because p_1 and p_3 are feasible to transmit the output data of n_2 , and p_5 is applicable to transmit the output data of n_3 (although p_4 and p_3 are connected, p_3 is infeasible since $\Pr(|e_{p_4, p_3}^{\text{VC}}(t)| > \text{TT}_{n_3, n_4}(p_4, p_3)) < \theta$), while the execution of n_4 requires the output data from all its immediate predecessor subtasks. This example demonstrates that it is not always advantageous to schedule each subtask to the processor that offers the minimum EFT, especially when considering dynamic topologies and resources.

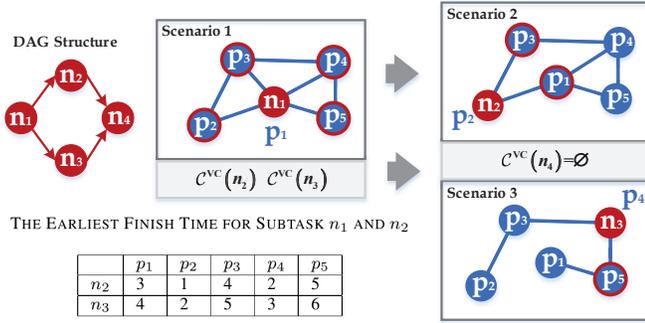


Fig. 6. An example of using HEFT to schedule DAG task over dynamic VC.

2) *Proposed Degree-based Weighted EFT Mechanism*: To develop our alternative methodology to HEFT, we first define *degree set*.

Definition 5. (Degree set). Let $\mathcal{D}^{\text{VC}}(n_i, p_m)$ denotes a degree set containing the vehicles that can receive the data transmission of $\max_{n_j \in \text{succ}(n_i)} \{c_{n_i, n_j}\}$ after n_i is completed on p_m , where $\text{succ}(n_i)$ is the immediate successors set of n_i , which can be given by

$$\mathcal{D}^{\text{VC}}(n_i, p_m) \triangleq \{p_n : \exp\left(-\text{TT}_{n_i, n_j}^{\text{max}}(p_m, p_n) \mu_{p_m, p_n}\right) \geq \theta, p_m \in \mathcal{C}^{\text{VC}}(n_i), p_n \in \mathcal{V}^{\text{VC}}(n_j)\}. \quad (20)$$

And according to (3), $\text{TT}_{n_i, n_j}^{\text{max}}(p_m, p_n)$ is the data transmission time of $\max_{n_j \in \text{succ}(n_i)} \{c_{n_i, n_j}\}$ between p_m and p_n , expressed by

$$\text{TT}_{n_i, n_j}^{\text{max}}(p_m, p_n) = \max_{n_j \in \text{succ}(n_i)} \{c_{n_i, n_j}\} \times \Gamma(\text{PL}(d_{p_m, p_n}(\text{st}_{n_j}))), \xi_{n_i, p_m} = 1, p_m \in \mathcal{C}^{\text{VC}}(n_i), p_n \in \mathcal{V}^{\text{VC}}(n_j), \quad (21)$$

where $\text{succ}(n_i)$ is the immediate successor set of subtask n_i .

We then assign subtask to the vehicle p_m to minimize the

Algorithm 1: Ranking and Foresight-Integrated Dynamic (RFID) Scheduling Scheme

- 1: **Input:** $\mathcal{G}^{\text{A}}, \mathcal{G}^{\text{VC}}(n_{\text{entry}}), w_{n_i}, c_{n_i, n_j}, f_{p_m}$
 - 2: **Output:** Scheduling decisions $\{\xi_{n_i, p_m}\}$
 - 3: Schedule subtask n_{entry} on the task owner
 - 4: **while** there are subtask $n_i \in \mathbf{n}_{\text{ready}}$ **do**
 - 5: $\text{st}_{n_i} = \max_{n_j \in \text{pred}(n_i)} \{\text{AFT}(n_j)\}$
 - 6: Calculate the value of $\text{RSRank}^{\text{D}}(n_i)$ according to (15)-(19)
 - 7: Rank $n_i \in \mathbf{n}_{\text{ready}}$ using the value of RSRank^{D} in a non-decreasing order
 - 8: $N \leftarrow$ unscheduled subtask with lowest RSRank^{D}
 - 9: $L \leftarrow$ immediate successor subtasks of N
 - 10: **for** vehicles p_m existing in $\mathcal{C}^{\text{VC}}(N)$ **do**
 - 11: Calculate $\text{EFT}(N, p_m)$ using (1)-(9)
 - 12: Calculate $\mathcal{D}^{\text{VC}}(N, p_m)$ using (20)-(21)
 - 13: Calculate $\text{EFT}^{\text{W}}(N, p_m)$ according to (22)
 - 14: Return to the beginning of this loop
 - 15: **end for**
 - 16: Schedule subtask N on vehicle p_m such that $\text{EFT}^{\text{W}}(N, p_m) < \text{EFT}^{\text{W}}(N, p_n), p_n \in \mathcal{C}^{\text{VC}}(N) \setminus p_m$
 - 17: **if** there are new ready subtasks **then**
 - 18: Return to Line 4
 - 19: **else**
 - 20: Go to Line 8
 - 21: **end if**
 - 22: **end while**
-

weighted EFT:

$$\text{EFT}^{\text{W}}(n_i, p_m) = \alpha^{\text{T}} \text{EFT}(n_i, p_m) - \alpha^{\text{R}} \Phi(|\mathcal{D}^{\text{VC}}(n_i, p_m)|), p_m \in \mathcal{C}^{\text{VC}}(n_i), \quad (22)$$

where α^{T} , and α^{R} represent the preference on completion time and execution success rate (i.e., reliability), respectively. Also, $\Phi()$, which is chosen to be $\Phi(|\mathcal{D}^{\text{VC}}(n_i, p_m)|) = 0.5 \times |\mathcal{D}^{\text{VC}}(n_i, p_m)|$, is a monotone increasing function. Specifically, in (22) we not only consider the performance (in terms of completion time) when n_i is assigned to p_m , but also look ahead to the number of vehicles (i.e., $|\mathcal{D}^{\text{VC}}(n_i, p_m)|$) the can receive the data transmission of $\max_{n_j \in \text{succ}(n_i)} \{c_{n_i, n_j}\}$, after n_i is completed on p_m . Note that a larger value of $|\mathcal{D}^{\text{VC}}(n_i, p_m)|$ can lead to a lower value for $\text{EFT}^{\text{W}}(n_i, p_m)$ (i.e., a vehicle with general computation ability but commendable transmission potential can be given with high weight).

D. Summary

In RFID, the scheduling priority of each subtask relies heavily on the selected vehicles of its immediate predecessor subtasks, which is also influenced by the current resources supply. Then, a degree-based weighted EFT is leveraged to assign subtask to the vehicle, which can offer faster processing with reliable transmission links.

Algorithm 1 shows the details of RFID. It can be verified that the time complexity of RFID is $\mathcal{O}((n+r)^2 \cdot p^2)$ where

n , r , and p are the number of subtasks within the DAG task, the maximum number of successors per DAG task, and the maximum number of vehicles involved in the VC, respectively.

E. A Toy Example

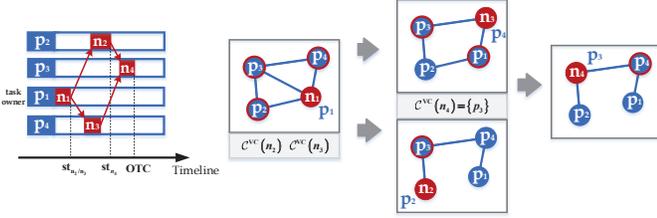


Fig. 7. Subtask-vehicle mapping.

A feasible mapping among subtasks and vehicles associated with VC_2 (given by Fig. 2) is shown in Fig. 7, where the left subplot represents a specific processing procedure (i.e., the assignment of subtasks to vehicles); while the right subplot depicts the changing topology of VC_2 and corresponding $C^{VC}(n_i)$ during the scheduling procedure. Since n_1 (i.e., n_{entry}) is scheduled on p_1 (i.e., task owner), subtasks n_2 and n_3 become *ready subtasks*. After sorting n_2, n_3 under resource-scarcity based dynamic downward ranking mechanism, n_3 is scheduled firstly to its most desirable vehicle, based on its candidate vehicle set $C^{VC}(n_2) = \{p_1, p_2, p_3, p_4\}$. Then, according to the degree-based weighted EFT, n_3 and n_2 are scheduled on p_4 and p_2 , respectively. Finally, subtask n_4 becomes *ready subtask*, and is scheduled on vehicle p_3 , which can receive the output data both of n_2 and n_3 .

V. PERFORMANCE EVALUATION

We conduct comprehensive simulations to evaluate the performance of RFID. To quantify the performance of our proposed methodology, we consider three key performance metrics: *i*) overall DAG task completion time, *ii*) execution success rate, and *iii*) running time of the algorithm.

A. Simulation Setup

Parameter setting of VC: We consider a real-world traffic region (shown in Fig. 8(a)) with size of 1km \times 1km in Xiamen, Fujian, China, obtained from OpenStreetMap [30]. SUMO [31] is utilized to import mobile vehicles and subsequently form a realistic VC over the simulation region in Fig. 8(b). Assuming that vehicles located within 500m from each other are connected via one-hop V2V links and form an undirected graph (VC). We conduct simulations upon considering different number of vehicles to better capture various vehicle density associated with a VC, and different VC's topologies. Each vertex in the VC graph represents a vehicle with certain computing capability, which follows a normal distribution with mean 20MHz and variance 0.2 [29]. The weight of each link (edge) connecting two vertices represents the residual contact duration. In addition, a monotone increasing function $\Gamma(\text{PL}(d_{p_m, p_n}(t))) = 0.15 \times \text{PL}(d_{p_m, p_n}(t)) + 0.001$ is applied to determine the transmission time between different vehicles.

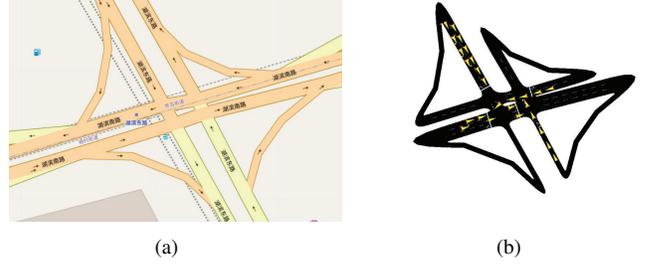


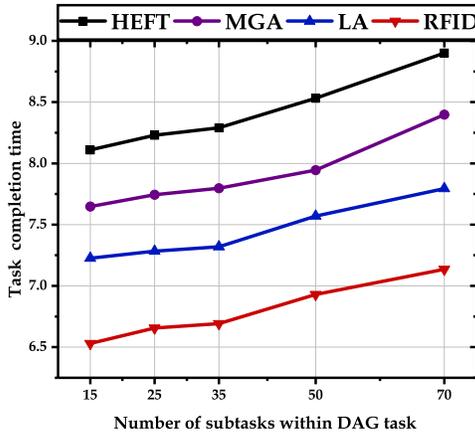
Fig. 8. VC network visualization.

Parameter setting of DAG task: We implement a DAG generator [17] to randomly generate different types of DAG task, regarding the number of subtasks, communication-to-computation ratio (CCR) of subtasks, and the number of layers. Specifically, subtask belongs to the same layer can be processed in parallel and each layer in the DAG contains at least one subtask. Besides, the first and the last layer are occupied only by n_{entry} , and n_{exit} , respectively. The computing workload of each subtask obeys a normal distribution with mean 3Mclock cycles and 0.2 variance [16]. The corresponding transmission data size of each edge follows a normal distribution with mean 1.2Mbit and variance 0.2 [16]. Also, we consider a real DAG task, i.e., molecular dynamics code DAG, to achieve better performance evaluation, which has been adopted in many existing works, such as [21], [23].

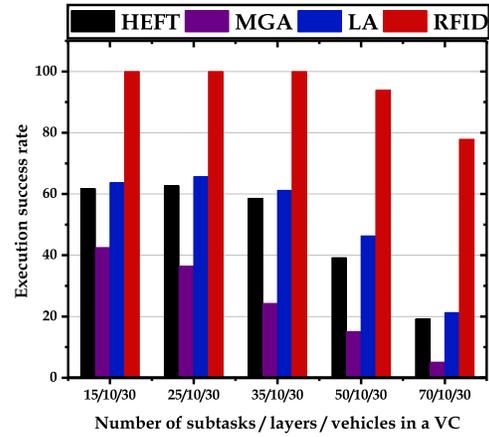
B. Benchmarks

To better verify the performance of our proposed RFID, three benchmarks are considered below, as inspired by some existing works:

- *Heterogeneous Earliest Finish Time (HEFT) [23]:* HEFT firstly ranks all the subtasks according to their average completion time, which is computed recursively from the entry subtask, i.e., conventional *downward ranking*. Then, the subtask with the highest scheduling priority is assigned to the vehicle that can process the subtask in the shortest time without considering the corresponding transmission constraints (14).
- *Lookahead (LA) [11]:* As an improvement of HEFT, LA ranks subtasks similar to HEFT. Then, it assigns subtask n_i with the highest scheduling priority to vehicle p_m , which can minimize the maximum completion time of subtask n_j , where $n_j \in \text{succ}(n_i)$, after n_i completes on p_m . In this paper, one-step LA is considered; namely, we only care about the immediate successors of each current scheduling subtask.
- *Modified Genetic Algorithm (MGA) [18]:* MGA consists of three key parts. First, an integer encoding is employed to denote the assignment between subtasks and vehicles. Then, to guarantee the feasibility of generated solutions, the definition of relatives (that is, two subtasks with dependency) is used to avoid the mis-operation of crossover (a subtask cannot be scheduled earlier than its

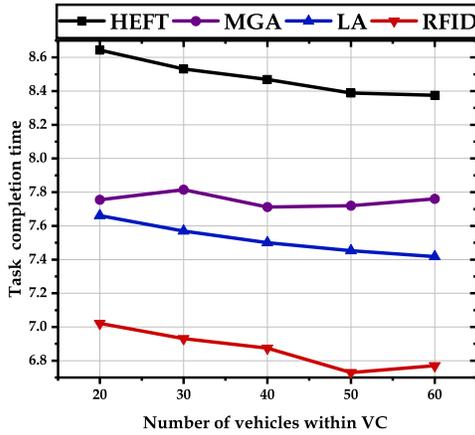


(a)

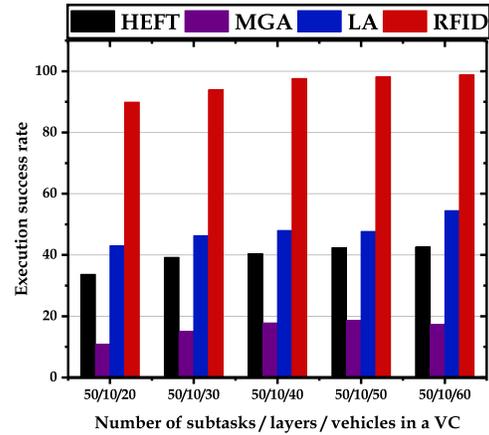


(b)

Fig. 9. Performance evaluations upon considering different number of subtasks within DAG task, for randomly generated DAG tasks.



(a)



(b)

Fig. 10. Performance evaluations upon considering different number of vehicles within VC, for randomly generated DAG tasks.

predecessor). Finally, the mutation is adopted to improve the fitness of the candidate solutions.

C. Simulation Results of Randomly Generated DAG Tasks

Performance comparisons and evaluations are conducted, in terms of average completion time and execution success rate, by considering diverse numbers of subtasks, vehicles, layers of DAG tasks, and CCR of the subtask. Besides, 1000 independent iterations are simulated, as benefited by the Monte Carlo method.

1) *Impacts of diverse numbers of subtasks*: Fig. 9(a) shows the performance evaluation of the overall DAG task completion time, with an increasing number of subtasks (from 15 to 70). The number of layers is set as 10; CCR is set by 1, and the number of vehicles involved in the initial VC is set as 30. As can be seen from Fig. 9(a), our proposed RFID outperforms the other three baseline methods with a much faster task

completion time. Specifically, since low resource demands can lead to weak resource competition among vehicles, the curve of task completion time between 25 subtasks and 35 subtasks is relatively flat. Compared to LA, although completion time is not the only indicator for optimization, RFID can reduce the overall DAG task completion time with the help of changing the scheduling priority of some specific subtasks according to their resource availability. In summary, the performance improvement of RFID in terms of task completion time is 19.48% better than HEFT, 14.61% better than MGA, 9.63% better than LA at 15 subtasks; and is 19.83% better than HEFT, 15.02% better than MGA, and 8.43% better than LA at 70 subtasks.

Fig. 9(b) illustrates the performance with an increasing number of subtasks (from 15 to 70) on the execution success rate. It can be observed that RFID significantly increases the corresponding execution success rate. Notably, there exists a

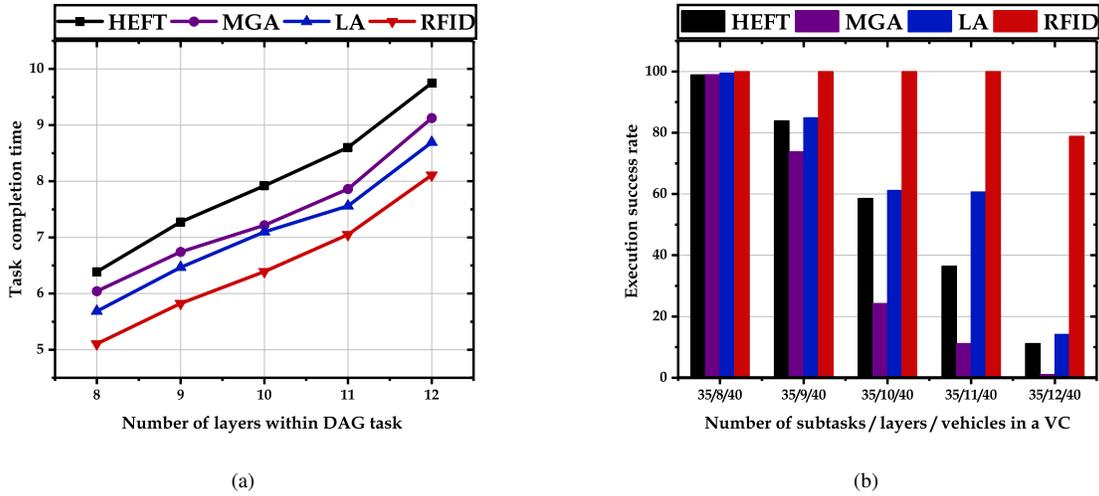


Fig. 11. Performance evaluations upon considering different number of layers, for randomly generated DAG tasks.

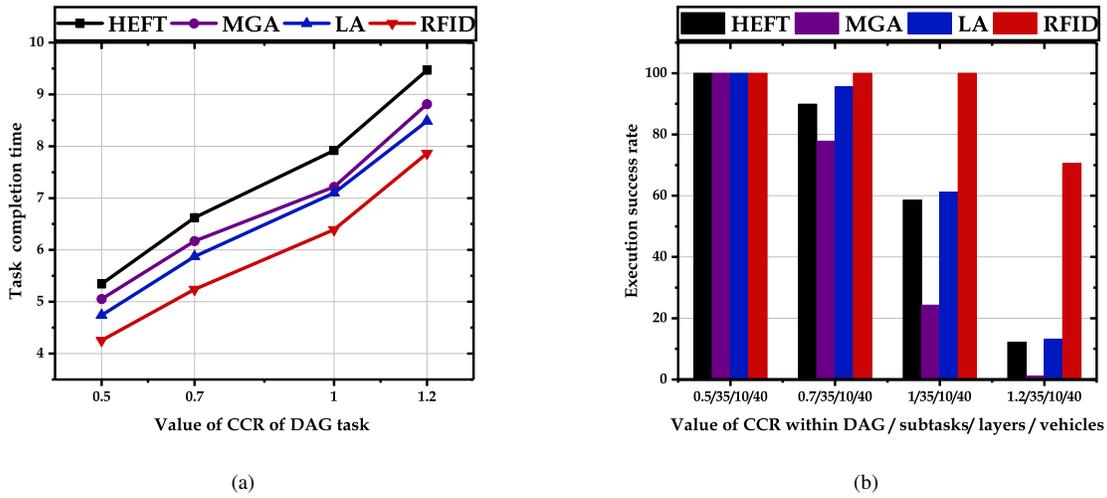


Fig. 12. Performance evaluations upon considering various CCR, for randomly generated DAG tasks.

decreasing execution success rate in the curve of HEFT, MGA, and LA, mainly due to the increasing number of subtasks, which requires heavily V2V connection time (i.e., the higher requirement on residual V2V contact duration). Besides, in Fig. 9(b), MGA has the worst performance on execution success rate because of its randomly generated solution space. In comparison with HEFT, LA slightly enhances the execution success rate by considering the impact of scheduling the successors of each assigned subtask. Also, RFID outperforms LA by increasing the probability of scheduling each subtask to the reliable connected vehicles by considering the degree of each vehicle. In summary, the performance improvement of RFID in terms of success rate is 38.3% better than HEFT, 57.5% better than MGA, 36.6% better than LA at 15 subtasks; and is 58.7% better than HEFT, 72.8% better than MGA, and 56.6% better than LA at 70 subtasks.

2) Impacts of diverse numbers of vehicles involved in VC:

Fig. 10(a) demonstrates the performance by increasing the number of vehicles from 20 to 60 on the overall DAG task completion time. The number of layers is set as 10; the CCR is set by 1; and the number of subtask is set by 50. It can be seen from Fig. 10(a) that the increasing number of vehicles can significantly accelerate the completion of the DAG tasks thanks to more sufficient resources. However, after more than 50 vehicles, task completion time can not be reduced significantly, even slightly increasing. This phenomenon mainly owes to the redundancy of available resources complicating VC's topology, which thus may cause subtasks to be assigned to vehicles with poor computing capability. Specifically, the growing resource supply impacts less on MGA, since each subtask is randomly scheduled on the vehicles at the beginning, while the chromosome crossing procedure of MGA may fail to ensure the feasibility of the newly generated solution (i.e., the connection constraints of newly assigned subtasks are

not taken into account in advance). To this end, an increasing number of vehicles in MGA does not imply an increase in the number of feasible solutions, which thus fails to bring an obvious decrease in task completion time. In summary, the performance improvement of RFID in terms of the task completion time is 18.76% better than HEFT, 9.45% better than MGA, 8.33% better than LA at 20 vehicles; and is 19.16% better than HEFT, 12.76% better than MGA, and 8.72% better than LA at 60 vehicles.

Fig. 10(b) evaluates the performance of execution success rate upon considering the different numbers of vehicles involved in a VC. It can be observed that an increasing number of vehicles can bring a larger execution success rate due to more available resources. The performance trends of MGA, HEFT, and LA in terms of the execution success rate are similar to that of Fig. 10(b). Specifically, the performance improvement of RFID in terms of the execution success rate is 56.2% better than HEFT, 79% better than MGA, 46.9% better than LA at 20 vehicles; and is 56.2% better than HEFT, 81.5% better than MGA, and 44.3% better than LA at 60 vehicles.

3) *Impacts of diverse numbers of layers of DAG task:* Fig. 11(a) shows the impacts of changing the number of layers from 8 to 12 on the overall DAG task completion time. The number of subtasks is set by 35; CCR is set as 1, and the number of vehicles is set by 40. Interestingly, increasing the number of layers can significantly lead to a larger overall DAG task completion time due to unsatisfying parallelism of the DAG task. Specifically, when the parallelism of a DAG task keeps decreasing (namely, the number of layers keeps increasing), more subtasks require sequential execution, which causes an increase in the completion time of the DAG task. Notably, when the number of layers of a DAG task equals that of subtasks (e.g., chess game task shown in Fig. 1), local computing will become the best choice to avoid frequent data transmission among vehicles. In addition, as the number of layers increases, MGA and LA perform almost equivalently, which indicates that the decrease of the average number of successors of each subtask can result in an ineffectiveness when adopting LA, which concerns the impact of scheduling of the successors of the each assigned subtask. Specifically, the performance improvement of RFID in terms of the overall DAG task completion time is 20.08% better than HEFT, 15.5% better than MGA, 10.22% better than LA at 8 layers, and is 16.82% better than HEFT, 11.15% better than MGA, and 6.74% better than LA at 12 layers.

Fig. 11(b) depicts the execution success rate performance upon considering different layers (from 8 to 12) in a DAG task. It can be observed from Fig. 11(b) that all the four algorithms show a decreasing trend in execution success rate as the number of layers increases, especially for HEFT, MGA, and LA. The key reasons are that increasing the number of layers can lead to increased data transmission time between interdependent subtasks and ready time at the vehicles. For example, given a small number of layers (e.g., one layer contains many subtasks), most subtasks can be processed in parallel, which thus results in fast completion. More importantly, the

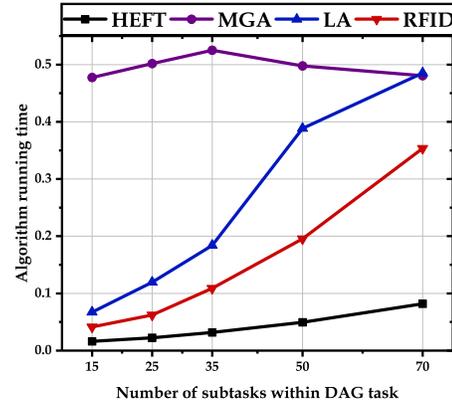


Fig. 13. Performance evaluations upon considering different number of subtasks associated with DAG task, on algorithm's running time.

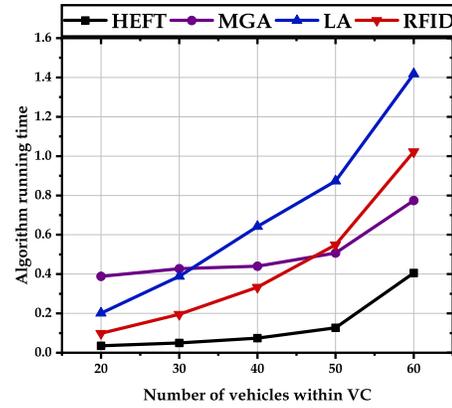


Fig. 14. Performance evaluations upon considering different number of vehicles associated with VC, on algorithm's running time.

VC's topology during task scheduling may stay relatively stable due to the high parallelism of the DAG task. Using MGA can cause sharp performance degradation as the number of layers changes from 9 to 10, which indicates that layers heavily impact the random-based subtask-vehicle assignment mechanism. In summary, the performance improvement of RFID in terms of the execution success rate is 1.1% better than HEFT, 1% better than MGA, 0.5% better than LA at 8 layers; and is 67.7% better than HEFT, 78.8% better than MGA, and 64.7% better than LA at 12 layers.

4) *Impacts of considering various CCR of DAG Task:* Different values of CCR are tested in Fig. 12(a). The number of subtasks and layers are set by 35 and 10, respectively, while the number of vehicles is set as 40. As can be seen from Fig. 12(a), the overall DAG task completion time rises as the CCR of each subtask increases due to the growing data transmission time. Additionally, MGA and LA show similar performance trends in Fig. 11(a), which indicates that an increase in the CCR of the DAG task conforms to a decrease in the parallelism of the DAG task and thus results in a significant raising in data

transmission time. In summary, the performance improvement of RFID in terms of the task completion time is 20.39% better than HEFT, 15.79% better than MGA, 10.23% better than LA at 0.5 CCR; and is 16.95% better than HEFT, 10.77% better than MGA, and 7.34% better than LA at 1.2 CCR.

Fig. 12(b) compares the performance on execution success rate by considering CCR from 0.5 to 1.2. Similar to Fig. 11(b), given a small CCR, due to the fast data transmission, VC's topology stays relatively stable during the execution of subtasks, and all the algorithms can enjoy a high execution success rate. In summary, the performance improvement of RFID in terms of the execution success rate is as same as HEFT, MGA, and LA at 0.5 CCR and is 58.4% better than HEFT, 69.5% better than MGA, and 57.4% better than LA at 1.2 CCR.

D. Comparison of Running Time of Different Algorithms

In this subsection, we conduct a performance comparison in terms of the algorithm running time by considering various numbers of subtasks associated with randomly generated DAG tasks and different vehicle densities.

1) *Impacts of diverse numbers of subtasks:* Fig. 13 shows the impact on algorithm running time caused by a varying numbers of subtasks within a DAG task (from 15 to 70). The CCR of the DAG task is set by 1; the number of layers is set as 10, and the number of the vehicle is considered by 30. Due to the predefined initial population number of MGA (mainly used to generate random assignments), the running time of MGA is slightly affected by increasing the number of subtasks. Besides, in comparison with the other three algorithms, MGA has the longest running time caused by the internal iteration time for convergence. Also, HEFT outperforms other algorithms in running time since increasing the number of subtasks can lead to an extra increasing time in evaluating the performance of corresponding successors, which significantly raises the running time of LA. Since RFID only needs to evaluate the transmission data size between different subtasks and successors, the corresponding running time remains relatively lower as compared to LA. In summary, although the running time of RFID is higher than that of HEFT, its overall DAG task completion time and execution success rate significantly outperforms HEFT, as verified by Fig. 9-Fig. 12.

2) *Impacts of diverse number of vehicles:* Fig. 14 compares that performance upon considering a varying number of vehicles (from 20 to 60) on the algorithm's running time. The number of subtasks and layers are set by 50 and 10, respectively, while the CCR of the DAG task is set as 1. The growing vehicles can bring rising running time since more resource providers should be considered during task scheduling. Specifically, LA's running time is significantly impacted by the number of vehicles, which mainly owes to the increased time spent on evaluating the performance of each vehicle to determine the assignment of each subtask. Similarly, in RFID, more vehicles can complicate the topology of VC, resulting in an increased running time for calculating

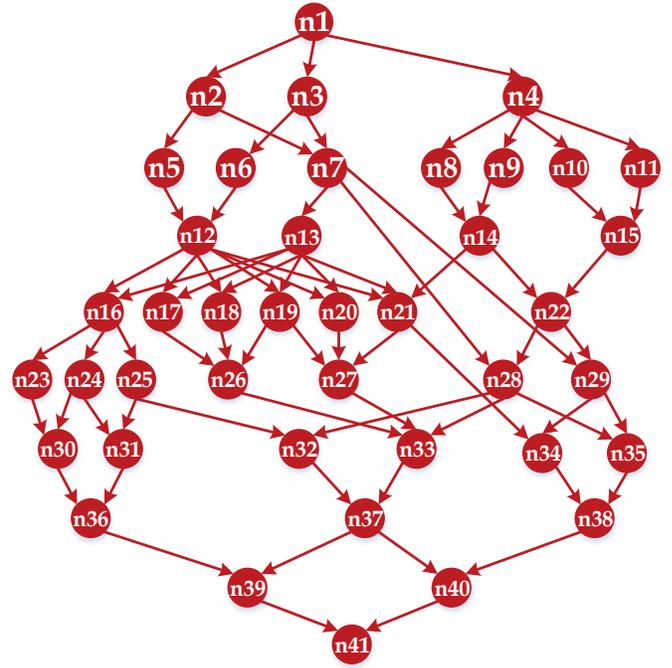


Fig. 15. The DAG of the molecular dynamics code. [21] [23]

TABLE II
PERFORMANCE COMPARISON UNDER DIFFERENT METRICS FOR MOLECULAR DYNAMICS CODE DAG

Metric	HEFT	MGA	LA	RFID
Overall completion time	7.7240	7.0638	6.8537	6.2233
Execution success rate	72.3	62.7	72.7	100
Algorithm running time	0.0499	0.6575	0.1409	0.1268

the degree of each vehicle. In summary, although the running time of RFID is higher than that of HEFT, its overall DAG task completion time and the execution success rate is significantly better than HEFT as verified in Fig. 9-Fig. 12.

E. Simulation Results for Real Application DAG Task

Fig. 15 depicts a real-world DAG task of a modified molecular dynamic code [21], [23]. Table II presents the performance comparison of different algorithms regarding the overall DAG task completion time, the execution success rate, and the algorithm running time. It can be observed that LA and RFID algorithms exhibit a higher running time than others (i.e., HEFT and MGA). However, regarding overall DAG task completion time, the performance improvement of RFID is 19.43% better than HEFT, 11.9% better than MGA, 9.19% better than LA, and is 27.7% better than HEFT, 37.3% better than MGA, 27.3% better than LA in terms of execution success rate. In summary, simulation results verify that our proposed RFID algorithm offers an efficient and commendable reference in scheduling DAG tasks over dynamic VCs.

VI. CONCLUSION

In this paper, we investigate the DAG task scheduling problem over dynamic VC with the goal of minimizing the overall DAG task completion time while ensuring high execution success rate. We formulate DAG task scheduling as a 0-1 integer programming problem, which is NP-hard. To tackle the problem, we propose RFID, which considers the availability and scarcity of vehicles' resources in determining the scheduling priority of different subtasks. Subsequently, RFID selects the processing vehicles based on their degree (i.e., feasible V2V connections) and resources to reduce the latency of task processing while ensuring a high reliability. Comprehensive simulations are conducted to evaluate the performance of the proposed RFID while considering multiple existing benchmarks for performance comparison. Performance comparisons reveal that our proposed RFID outperforms the existing methods in terms of the overall DAG task completion time and the execution success rate while having a marginally higher running time. Several future directions can be considered, such as the cooperation among different VCs, resource competitions among multiple DAG takes, and auction-based task allocation mechanisms.

REFERENCES

- [1] M. Shojafar, N. Cordeschi and E. Baccarelli, "Energy-Efficient Adaptive Resource Management for Real-Time Vehicular Cloud Services," *IEEE Trans. on Cloud Comput.*, vol. 7, no. 1, pp. 196-209, 1 Jan.-March 2019.
- [2] M. Barbera, S. Kosta, A. Mei, and J. Stefa, "To Offload or Not To Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 1285-1293.
- [3] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637-646, Oct. 2016.
- [4] W. He, G. Yan and L. D. Xu, "Developing Vehicular Data Cloud Services in the IoT Environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1587-1595.
- [5] J. Zhao, Q. Li, Y. Gong and K. Zhang, "Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944-7956, Aug. 2019.
- [6] R. Florin, A. Ghazizadeh, P. Ghazizadeh, S. Olariu, and D. C. Marinescu, "Enhancing Reliability and Availability through Redundancy in Vehicular Clouds," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1061-1074, 2021.
- [7] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795-2808, October 2016.
- [8] 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.
- [9] S. Bi, and Y. J. Zhang, "Computation Rate Maximization for Wireless Powered Mobile-Edge Computing with Binary Computation Offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177-4190, 2018.
- [10] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-Edge Computation Offloading for Ultra-dense IoT Networks," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4977-4988, 2018.
- [11] L. F. Bittencourt, R. Sakellariou and E. R. M. Madeira, "DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm," *Proc. Eur. Conf. Parallel Process.*, 2010, pp. 27-34.
- [12] G. C. Sih and E. A. Lee, "A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 2, pp. 175-187, Feb. 1993.
- [13] H. Arabnejad and J. G. Barbosa, "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682-694, March 2014.
- [14] H. Kanemitsu, M. Hanada and H. Nakazato, "Clustering-based Task Scheduling in a Large Number of Heterogeneous Processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 11, pp. 3144-3157, 1 Nov. 2016.
- [15] M. Aggarwal, R. D. Kent and A. Ngom, "Genetic Algorithm based Scheduler for Computational Grids," *Int. Symp. High Perform. Comput. Syst. and Appl.*, 2005, pp. 209-215.
- [16] Y. Sahni, J. Cao, L. Yang and Y. Ji, "Multihop Offloading of Multiple DAG Tasks in Collaborative Edge Computing," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4893-4905, 15 March 2021.
- [17] L.-C. Canon, M. El Sayah, and P.-C. Héam, "A Comparison of Random Task Graph Generation Methods for Scheduling Problems," *Proc. Eur. Conf. Parallel Process.*, 2019, pp. 61-73.
- [18] F. Sun et al., "Cooperative Task Scheduling for Computation Offloading in Vehicular Cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049-11061, Nov. 2018.
- [19] H. Liu, H. Zhao, L. Geng and W. Feng, "A Policy Gradient based Offloading Scheme with Dependency Guarantees for Vehicular Networks," *IEEE Global Commun.*, 2020, pp. 1-6.
- [20] Z. Liu et al., "Topology-Aware Dynamic Computation Offloading in Vehicular Networks," *IEEE Veh. Technol. Conf.*, 2021, pp. 1-5.
- [21] S. Sundar and B. Liang, "Offloading Dependent Tasks with Communication Delay and Deadline Constraint," *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Honolulu, HI, USA, 2018, pp. 37-45.
- [22] C. Shu, Z. Zhao, Y. Han, G. Min and H. Duan, "Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1678-1689, March 2020.
- [23] H. Topcuoglu, S. Hariri and Min-You Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260-274, March 2002.
- [24] J. Ghaderi, S. Shakkottai, and R. Srikant, "Scheduling Storms and Streams in The Cloud," *ACM Trans. Modeling and Performance Eval. of Comput. Syst.*, vol. 1, no. 4, pp. 1-14, 2016.
- [25] L. Shi, Z. Zhang, and T. Robertazzi, "Energy-Aware Scheduling of Embarrassingly Parallel Jobs and Resource Allocation in Cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1607-1620, 2017.
- [26] M. LiWang, S. Hosseinalipour, Z. Gao, Y. Tang, L. Huang, and H. Dai, "Allocation of Computation-Intensive Graph Jobs over Vehicular Clouds in IoV," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 311-324, 2019.
- [27] M. LiWang, Z. Gao, S. Hosseinalipour, and H. Dai, "Multi-Task Offloading over Vehicular Clouds under Graph-based Representation," *IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1-7.
- [28] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856-868, Jan. 2019.
- [29] S. Misra and S. Bera, "Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2071-2078, Feb. 2020.
- [30] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12-18, Oct.-Dec. 2008.
- [31] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," *Int. Conf. Intell. Transp. Syst. (ITSC)*, 2018, pp. 2575-2582.
- [32] J. Yu, X. Hao, Z. Cui, P. He, and T. Liu, "Boosting Fairness for Masked Face Recognition," *Proc. IEEE/CVF Int. Conf. Comput. Vision (ICCV) Workshops*, Virtual, Oct. 2021, pp. 1531-1540.
- [33] T. Sarkar, Z. Ji, K. Kim, A. Medouri, and M. Salazar-Palma, "A Survey of Various Propagation Models for Mobile Communication," *IEEE Antennas Propag. Mag.*, vol. 45, no. 3, pp. 51-82, Jun. 2003.
- [34] M. Liwang, Z. Gao, and X. Wang, "Energy-Aware Graph Task Scheduling in Software-Defined Air-Ground Integrated Vehicular Networks," [arXiv:2008.01144](https://arxiv.org/abs/2008.01144), 2021.
- [35] X. Zhu, Y. Li, D. Jin, and J. Lu, "Contact-Aware Optimal Resource Allocation for Mobile Data Offloading in Opportunistic Vehicular Networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7384-7399, 2017.
- [36] M. Giordani, T. Shimizu, A. Zanella, T. Higuchi, O. Altintas and M. Zorzi, "Path Loss Models for V2V mmWave Communication: Performance Evaluation and Open Challenges," *IEEE 2nd Connected and Autom. Vehicles Symp. (CAVS)*, 2019, pp. 1-5.