Throughput Maximization for Result Multicasting by Admitting Delay-aware Tasks in MEC Networks for High-speed Railways

Junyi Xu, Zhenchun Wei, *Member, IEEE*, Xiaohui Yuan, *Senior Member, IEEE*, Yan Qiao, *Member, IEEE*, Zengwei Lyu, and Jianghong Han

***Abstract*—The rapid expansion of high-speed railways (HSRs) and the growing demand for diverse data services during long journeys require efficient computing services. Mobile Edge Computing (MEC) emerged as a promising platform to fulfill this demand. We envision a scenario wherein passengers interact with each other on the same or different trains in real-time by offloading computationally intensive and delay-sensitive tasks to the track-side MEC networks for HSRs and computation results are multicast to the receivers. To improve the quality of data services, we propose a novel approach to optimize network throughput by admitting as many tasks as possible, subject to delay constraints, and multicasting the maximum number of results. The high mobility of trains and the frequent handovers during train-ground communication are factored into our scheme, which presents significant challenges to jointly consider the dynamic multicast grouping and admission/rejection policies for tasks/results. We introduce the multi-group-shared Group Steiner tree (GST) model and propose an efficient heuristic algorithm that reduces the multicast routing problem to finding a GST for each candidate cloudlet. The effectiveness of our proposed algorithm is demonstrated through simulations and the results are promising.**

***Index Terms*—Edge computing, High-speed railways, Multicasting, Group Steiner Tree problem, Admission control.**

Supplemental Materials: https://github.com/junyixu-git/MEC-HSR-Multicast/blob/main/supplemental.pdf

I. INTRODUCTION

High-speed railways have revolutionized the transportation sector, leading to a spike in train travel. With the widespread use of mobile devices, superior data services for passengers are necessary. Mobile edge computing, which facilitates small cloud infrastructures at the network edge, offers seamless connections to computing services. This reduces response delays and improves network efficiency and user experience. MEC caters to the escalating resource demands of mobile users by enhancing device capabilities in real-time.

The shift from unicast (one-to-one) to multicast (one-to-many) communication has been notable, especially when multiple users request the same service such as video conferencing, online education, and cloud gaming. Early attempts [1], [2] have explored edge caching based content delivery through multicasting in HSR scenarios. Therefore, MEC-based multi-user interactions among passengers could be realized by offloading computation-intensive tasks. However, little research has addressed the fundamental routing problem for multicast in HSR dedicated MEC networks.

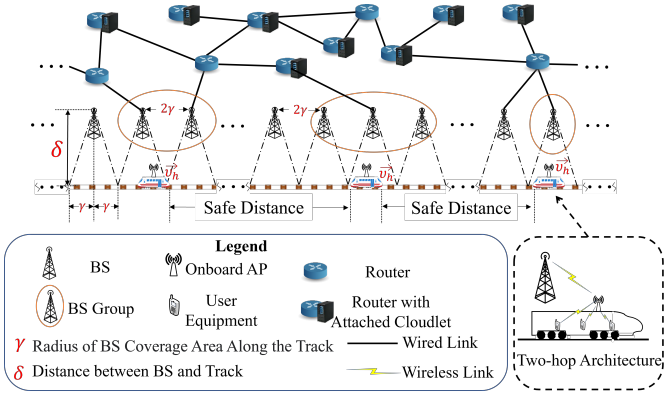


Fig. 1: An example of the scenario, which consists of track segments, trains, and track-side MEC network.

Addressing this gap, the main idea of this work is to propose an admission control method for the HSR-specific MEC network. The objective is to maximize the network throughput by admitting as many offloaded tasks as possible, subject to delay constraints, and multicasting the maximum number of computation results. This approach can significantly improve the provision of data services in HSRs.

Designing an approach to maximize multicast throughput in MEC networks for HSRs raises two questions:

1) How to effectively formulate multicast groups considering the high mobility of trains and frequent handovers during train-to-ground communications?

2) How to plan multicast paths to transfer tasks to the appropriate cloudlet to meet the delay requirements and multicast the results to the users?

Addressing these questions involves four primary challenges:

1) *Managing the base stations (BSs):* During offloading, tasks have delay requirements, and trains frequently handover across BSs. BSs play the crucial roles in building multicast groups, affecting the transmission of computation results.

2) *Dealing with the Sequential Process between Each Task and Its Results:* Unlike traditional multicast, in MEC networks, the computation result exists when and only when the task

J. Xu, Z. Wei, Y. Qiao, Z. Lyu, and J. Han are with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui Province, China, and also with the Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei, Anhui Province, China (e-mail: 2016010090@mail.hfut.edu.cn; weizc@hfut.edu.cn; qiaoyan@hfut.edu.cn; lzw@hfut.edu.cn; hanjh@hfut.edu.cn).

X. Yuan is with the Department of Computer Science and Engineering, University of North Texas, TX, 76203, USA (e-mail: xiaohui.yuan@unt.edu).

is executed. This sequential relationship affects path planning and cloudlet selection.

3) *Discovering the Suitable Multicast Model:* Traditional multicast schemes require precise specifications of a single source with multiple destinations, making these models difficult to cope with the high-speed and dynamic nature of HSRs. We need a model that does not require predefined destinations.

4) *Determining the Appropriate Cloudlet:* Task allocation in MEC networks should consider both data routing and computation time. Cloudlet selection affects routing paths, and consequently, data routing time.

Existing solutions often assume that the requirements and constraints of the HSR scenario are known in advance. However, due to the dynamic nature of HSRs, this is not always the case. Therefore, it is necessary to consider a scheme that can adapt to the changing conditions of HSRs while maximizing multicast throughput. The contributes of our work are as follows:

- We explore multicast tree construction with delays in the MEC network for HSRs, considering unique HSR scenario characteristics and constraints.
- We introduce an efficient heuristic algorithm for one-shot optimization. This algorithm can be extended for a long period by dividing time into equal slots and periodically invoking the proposed algorithm in each slot.
- We develop a scheme to dynamically construct multicast groups for BSs. This correlates inter-train data interaction with data traffic within the MEC network and network routing delay calculation with the prediction of train locations.

The remainder of the paper is organized as follows. Section II reviews the related work. Section III introduces the scenarios and systems models and basic concepts. Section IV introduces the group Steiner Tree-based routing paths and presents the preliminary analysis of delays and costs in the MEC network. Section V gives an Integer Linear Programming (ILP) formulation of the defined problem. Section VI describes our proposed method in detail. Section VII discusses the performance of the proposed algorithm. Section VIII concludes this paper with a summary.

II. Related Work

TABLE I: COMPARISON OF OUR WORK WITH EXISTING STUDIES

Study	Off.	Mul.	Dyn. Group	Delay	Arbi. Volume	User Mobi.
[3]–[5]	✓	✗	✗	✓	✗	✓
[6], [7]	✓	✓	✗	✗	✗	✗
[8], [9]	✓	✗	✗	✓	✗	✗
[10], [11]	✗	✓	✗	✗	✗	✓
[12]–[16]	✓	✗	✗	✓	✗	✓
[17]	✓	✗	✗	✓	✓	✓
[18]–[23]	✓	✓	✓	✓	✗	✓
Our Work	✓	✓	✓	✓	✓	✓

This section provides a review of recent research studies pertinent to MEC with a focus on throughput, multicast, and offloading, as well as the application of MEC for HSRs scenarios and the Internet of Vehicles (IoV). The comparison between our work and existing studies is summarized in Table I, where Off., Mul., Dyn., Arbi., and Mobi. are short for Offloading, Multicast, Dynamic, Arbitrary, and Mobility, respectively.

A. MEC Research on Throughput, Multicast and Offloading

1) Throughput: Deng et al. [3] addressed the critical issue of maximizing long-term throughput for multicell, multi-user MEC systems as MEC becomes increasingly fundamental in bolstering 5G. The authors propose a novel design that considers user association and resource allocation for communication and computing. Li et al. [4] optimized deep neural network inferences using MEC in an Internet of Things (IoT) environment, splitting the DNN inference model between the IoT device and a cloudlet in the MEC network. An innovative dynamic throughput maximization algorithm was introduced by [5] for a wireless powered MEC system, optimizing communication, computation, and energy resources.

2) Multicast: SCT et al. [6] optimized the joint caching and computing policy to minimize the transmission bandwidth. The proposed method employs Schrijver's algorithm, the concave-convex procedure and an alternating direction method of multipliers. Hao et al. [7] addressed the challenge of integrating multicast and MEC, introducing a multicast-aware resource allocation approach for MEC that optimizes computing and caching in multicast scenarios.

3) Offloading: An adaptive service offloading scheme for MEC was proposed to address issues including service latency minimization, optimal revenue maximization, and high-quality service requirement offloading [8]. Samanta et al. [9] introduced RAISE, a resource-agnostic microservice offloading scheme for mobile Industrial IoT devices in edge computing, facilitating the maximization of resource utilization and successful offloading. In [24], the authors present dynamic microservice scheduling for MEC to optimize task offloading, energy efficiency, and quality of service under varying network conditions.

B. MEC in HSR Scenarios

1) Edge Caching: To deliver popular services on trains, Li et al. [10] presented a cache-based scheme with converged wireless broadcast and cellular networks. Xiong et al. [11] suggested a novel scheme that encourages user terminals in HSR scenarios to cache wireless services. Two auction strategies are developed to maximize social incomes and minimize terminal waiting time.

2) Edge Computing: Chen et al. [12] proposed a real-time fault detection framework aided by edge computing for HSR traction systems. Liu et al. [13] introduced the edge computing model for fault detection and diagnosis for traction control systems for HSRs, where their objective is to minimize the execution cost of task offloading. Zhang et al. [14] examined dynamic resource allocation, computation offloading, and energy considerations in HSR networks. Li et al. [15] proposed a genetic algorithm-based scheme for transferring and scheduling predictive computations that are sensitive to mobility. Li et al. [16] took into account a mmWave-based train-ground communication system for the HSRs. Constrained

by the energy consumption of the local device and onboard mobile relays, the problem of minimizing the average task processing latency for all users is formulated. Xu et al. [17] designed a path-finding algorithm for managing unicast-based data traffic from task offloading tailored for HSRs, which solved the challenges of frequent handover in train-ground communication, thus maximizing the network throughput.

C. Multicasting in IoV

The aforementioned studies did not consider multicasting applications for HSR. HSRs are an ecosystem including infrastructure such as high-speed trains (HSTs), stations, and the planning and operational components of rail networks [25]. HSRs differ from highway transportation in the context of the IoV. HSRs mandate a safety distance [26] between trains and the limitations imposed by railroad tracks, resulting in distinct vehicle travel patterns. These particularities of HSR require a tailored approach in research studies. Although multicasting studies have gleaned insights from IoV [18]–[23], the direct application to HSR scenarios is infeasible.

Roger et al. [18] proposed a low-latency multicast scheme to decrease the latency of Vehicle-to-Anything (V2X) communications for autonomous driving applications. By jointly considering coded multicast and edge caching, Bao et al. [19] investigated the minimizing of data traffic and the redundant problem in vehicular ad hoc networks (VANETs). Kadhim et al. [21] presented an energy-efficient multicast routing protocol by introducing software-defined networks and fog computing in vehicular networks. Hui et al. [22] presented a cooperative content delivery system based on games in which BSs collaborate with RSUs to service a group of vehicles using multicast technology. Keshavamurthy et al. [20] analyzed the cloud-based sidelink resource allocation problem for multicast group transmissions in the case of cooperative automated driving (CAD), and a graph-based solution framework is proposed to form clusters and assign the inter-cluster resource block pool. Furthermore, they [23] also explored the multicast group-based vehicle-to-vehicle (V2V) communications for CAD scenarios by allocating the sidelink resource, constrained by reliability requirements and half-duplex limitation.

This is the first study to explore the construction of multicast trees constrained by delays in MEC networks for HSRs. Unlike the existing methods, this study considers that the data volume of the results to be multicast can be arbitrary, while the offloaded tasks have a relatively small data volume. Our task admission scheme relies on grouping the BSs based on the prediction of trains' localizations.

III. NOTATIONS, MODELS AND CONCEPTS

Figure 1 shows a series of high-speed trains traveling on a straight track. Each train is moving at a consistent speed in a uniform straight line, and these trains are maintained at a safe distance from one another. Along the track, a dedicated MEC network is deployed, consisting of BSs, routers, and cloudlets. Each category of these devices is homogeneous within itself, and all are interconnected via wired links. When a train traverses a segment of the railroad within the coverage area of a specific BS, passengers aboard can access the MEC network through the well-known two-hop architecture [14], [16]. Specifically, the passenger's device communicates with the trackside BS via an onboard Access Point (AP), utilizing wireless connections.

This study focuses on the processes from the moment that (i) a set of tasks offloaded by the users have arrived at the BSs, to the moment that (ii) the tasks' results are sent to the target trains. One *computing cloudlet* is chosen to compute all offloaded tasks. A *group-shared multicast routing tree*, which is rooted at the chosen computing cloudlet, is also found to route all tasks and results from the computing cloudlet to the specified *destination BS groups*, subject to various constraints. Tasks with excessive demands will be rejected until a feasible multicast tree can be obtained. The computation results are transmitted from the destination BS groups to the designated trains. The main notations are listed in the supplement [27] due to the limited space.

Let H be the train set. With high-accuracy localization and dynamic tracking by onboard GPS and railway monitoring equipment [2], velocity ν_h and location $l(h)$ of train $h \in H$ can be obtained continuously. Note that different trains may run at different speeds. For simplicity, we use the Cartesian coordinate system [17] to describe the locations of the objects in the system, and we define the direction of motion of the trains as the direction of the x-axis.

A. MEC Network Model

The MEC network is modeled by an undirected graph $G = (V, E)$, where V is the router set and E is the wired link set. Denote $V_c \subset V$ as the routers with the attached cloudlets, and the computing resource of each cloudlet is limited. For convenience, the terms of a router and its attached cloudlet will be interchangeably used unless confusion arises. The cloudlet is implemented using container-based virtualization technology. Let C_v be the computing capacity of each cloudlet $v_c \in V_c$, and the computing capacity is defined as the maximum number of containers in a cloudlet. Let F_v be the number of CPU cycles per second of each container in cloudlet v_c. Data transmission through routers V and links E incurs communication latency. We introduce a centralized controller based on software-defined network (SDN) [2], [28], which is logically deployed in the MEC network for global network management. The discussion of SDN is beyond the scope of this paper.

B. Tasks, Computation Results and Network Throughput

Let $K_H(t)$ be a set of tasks of a train set H arrived at BSs at moment t. The task set from a train h is denoted as $K_h \subseteq K_H(t)$. Let $k_{i,h} \in K_h$ be the ith task of train h. Then $k_{i,h}$ is represented by a triplet $k_{i,h} = \langle H_{i,h}^{des}; f_{i,h}, d_{req} \rangle$, where h is the *source train* for task $k_{i,h}$, $H_{i,h}^{des} \subseteq H$ is the set of the *destination trains*, $f_{i,h}$ is the CPU cycles demanded for computing task $k_{i,h}$, d_{req} is its end-to-end delay requirement. The tasks in $K_H(t)$ are offloaded by the same application, and each task has an identical delay requirement. These assumptions are reasonable because they reflect the common use of the same application, such as online video conferencing or online

gaming, by passengers in the HSR scenario, ensure a consistent Quality of Experience (QoE) for real-time services. The set of destination trains of K_h is denoted by H_h^{des}.

Each train offloads tasks and receives the results. Let $r_{i,h,h'}$ be the result of $k_{i,h}$ sent to train h', $h' \in H_{i,h}^{des}$. Notice that during the multicasting procedure, the computation results of task $k_{i,h}$ are sent to different destinations in the forms of copies and such results are totally the same, that is, $r_{i,h,1} = ... = r_{i,h,h'} = ... = r_{i,h,|H_{i,h}^{des}|}$. Denote by $R_{i,h}$ the results associated with task $k_{i,h}$, i.e., $R_{i,h} = \cup_{h' \in H_{i,h}^{des}} r_{i,h,h'}$. Furthermore, the set of computation results associated with the task set K_h is denoted as R_h, and the set of computation results of $K_H(t)$ is denoted as R_H, where $R_H = \bigcup_{h=1}^{|H|} R_h = \bigcup_{h=1}^{|H|} \left\{ \bigcup_{i=1}^{|K_h|} R_{i,h} \right\}$. Additionally, let R_h^{rec} be the set of computation results received by train h, and R_h^{rec} can be formulated as $R_h^{rec} = \bigcup_{i \in |R_{h'}|} \bigcup_{h' \in H} r_{i,h',h}$, where h is the destination train of each $h' \in H$. For simplicity, unless confusion arises, the terms of the computation result and its copy will be interchangeably used.

We introduce a function $z(\cdot)$ to obtain the data volume of a task/result. For example, $z(k_{i,h})$ is the data volume of task $k_{i,h}$, and $z(K_h)$ is the data volume of tasks in K_h, i.e., $z(K_h) = \sum_{i=1}^{|K_h|} z(k_{i,h})$. We define $\rho_{i,h} \in \mathbb{R}^+$ as the ratio between the volumes of task $k_{i,h}$ and result $r_{i,h,h'}$. The value of $\rho_{i,h}$ can be deduced using program profilers [29], [30].[1] Then the data volume $z(r_{i,h,h'})$ of task $k_{i,h}$ thus is $\rho_{i,h} \cdot z(k_{i,h})$.

Task set $K_H(t)$ is processed in a single cloudlet $v_K \in V_c$, named the computing cloudlet. The **advantages** of computing all the tasks in one cloudlet are that (i) the resources and users' data within a cloudlet can be effectively shared and (ii) the overhead of cross-cloudlet data communication and fusion can be avoided. A cloudlet can compute a task if there exists an idle container. Each task is computed exclusively by a container, and a unit of computing capacity will be consumed. Otherwise, a new container will be initialized as long as the cloudlet has adequate computing capacity.

Considering the QoE requirements of users in interactions, especially in scenarios where interactions are made using videos or images [33]–[35]. Therefore, the network throughput can be measured by how many computation results can be multicasted, where the results can be multicasted only if the corresponding task is admitted. Maximizing network throughput means maximizing the number of multicasted results.

C. Communication Model and Handovers

This section presents the key notations and concepts and a detailed description of the communication model is included in the supplement [27]. Set $B \subset V$ of BSs with routing capabilities is uniformly deployed along the tracks and $B \cap V_c = \emptyset$. The location of BS $b \in B$ is $l(b)$. The coverage radius of each BS is γ. Let $u_b(l(h), l'(h))$ denote the data volume that BS b continuously transmits to train h through a subcarrier during the period train h moves from location $l(h)$ to location $l'(h)$:

$$u_b(l(h), l'(h)) = \int_{l(h)}^{l'(h)} \mathcal{C}_b^{\downarrow}(x) \cdot \frac{|l(b) - x|}{\nu_h} dx, \tag{1}$$

where $\mathcal{C}_b^{\downarrow}(x)$ is the channel capacity of the downlink of BS b taking into account the Doppler effect [17], and $l(b) - \gamma \leq x \leq l(b) + \gamma$. The process of deriving Eq. (1) are put in the supplement [27]. Additionally, a train receives the maximum volume u_{max} of data from one BS when the train moves from $l(b) - \gamma$ to $l(b) + \gamma$, i.e.,

$$u_{max} = u_b \left(l(b) - \gamma, l(b) + \gamma \right). \tag{2}$$

In addition, the following cases of calculations related to Eq. (1) are often required in the rest of the article and all the mentioned calculations can be done by numerical integration [36]. **Case 1**: If volume $z(\cdot)$ and location $l(h)$ are given, location $l'(h)$ needs to be determined. **Case 2**: If volume $z(\cdot)$ and location $l'(h)$ are given, location $l(h)$ needs to be determined.

Due to the safe distances between trains, the BS within a certain geographical area that receives the computation results can only serve one train. The data volume of the results is arbitrary, and the maximum data volume that one single BS can transmit is limited. Thus a train may need to communicate with several BSs, i.e., handovers may occur during wireless transmission. Let $\mathcal{N}_h^{\downarrow}$ be the number of handovers for a train h. Without loss of generality, train h begins to receive the data of the computation results at location $l(h)$, which is in the coverage area of a certain BS b, i.e., $l(h) \geq l(b) - \gamma$. Meanwhile, the data volume of the computation results to be received by train h is $z(R_h^{rec})$. Similar to [17], there are four handover cases to be discussed, and we put the analysis in the supplement [27]. $\mathcal{N}_h^{\downarrow}$ can be expressed by

$$\mathcal{N}_h^{\downarrow} = \begin{cases} \lfloor z(R_h^{rec})/u_{max} \rfloor & \text{Case 1 or Case 3,} \\ \lceil z(R_h^{rec})/u_{max} \rceil & \text{Case 2 or Case 4.} \end{cases} \tag{3}$$

IV. Task Admission and Result Multicasting

A. Tree-based Routing Paths

Admitting task set involves routing each task from the given source(s) to a chosen computing cloudlet, then routing each result from the computing cloudlet to the given destination(s) via multicast. Such routing paths form the multicast tree(s). Next, we introduce the type of multicast tree that is appropriate for our scenario.

There are two ways to construct a multicast tree. One is constructing the per-source tree for the given BSs, and the other is constructing one shared tree covering all the designated BS groups. Lin et al. [37] gave a comparison of the mentioned multicast trees in terms of the number of trees in the network, the computation time, the load on the on-tree routers, and the end-to-end routing distances. The drawbacks of the per-source multicast tree are (i) the network controller needs to maintain a large number of multicast trees, and (ii) the complexities and difficulties of computing the per-source trees are much higher than computing one shared tree, especially considering the game in such a multi-party system while constructing the multiple per-source trees brought by the sharing the network resources. In summary, we adopt the approach of the multi-group shared tree for traffic routing, which means several

[1]The program profilers collect data potentially impacting offloading process outcomes, such as system memory acquired and used, CPU and GPU activity, execution times, function calls, and more [31], [32].

multicast groups share the same multicast tree, where all the data traffic of tasks and results can be routed simultaneously. Such a tree can achieve a better trade-off between the end-to-end delay and the workloads of the on-tree routers.

1) BS groups: To construct such a multicast tree, the BSs need to be divided into groups, and the groups need to be maintained to follow the moving trains. Computation results are first transmitted to different BS groups through multicast. Then the results are redistributed to the BSs in the same group, subject to the delay requirements and the location of trains, where the redistribution scheme is out of the scope of this paper and will be studied in our future work. Finally, a train that passes through the coverage area of a BS group fetches the data buffered in the BS group. We define the BS that caches the offloaded tasks as the *source BS* by correlating the inter-train interaction of data with the traffic within the MEC network. All tasks in K_h have been merged into identical source BS $s_h \in B$ at moment t. Moreover, we define the BSs to which result set R_h will be delivered as the *destination BSs*. Such BSs are denoted by $D_h \subseteq B$. It is worth mentioning that a BS can be both a source BS and a destination BS for the same train. Denote by $B_h^{\downarrow}$ the group of destination BSs for continuously transmitting result set R_h^{rec} to train h. According to Eq. (3), denote b_h^j as the jth BS in group $B_h^{\downarrow}$. Thus, the last BS in group $B_h^{\downarrow}$ can be represented as $b_h^{1+\mathcal{N}_h^{\downarrow}}$. Then group $B_h^{\downarrow}$ can be expressed as $B_h^{\downarrow} = \{b_h^1, ...b_h^j, ..., b_h^{1+\mathcal{N}_h^{\downarrow}}\}$. Moreover, let $\mathcal{U}(b_h^j)$ be the data volume of the result set R_h^{rec} that each BS $b_h^j \in B_h^{\downarrow}$ transmits to train h. According to Eq. (1), Eq. (2), and Eq. (3), we calculate $\mathcal{U}(b_h^j)$ as follows.

$$\mathcal{U}(b_h^j) = \begin{cases} z(R_h^{rec}) & \mathcal{N}_h^{\downarrow} = 0,\ j = 1 \\ z(R_h^{rec}) - \mathcal{U}(b_h^{1+\mathcal{N}_h^{\downarrow}}) - (\mathcal{N}_h^{\downarrow} - 1) \cdot u_{max} & \mathcal{N}_h^{\downarrow} \geq 1,\ j = 1 \\ u_{b_h^j}\left(l(b_h^j) - \gamma,\ l(h) + d_{req} \cdot \nu_h\right) & \mathcal{N}_h^{\downarrow} \geq 1,\ j = \mathcal{N}_h^{\downarrow} + 1 \\ u_{max} & \mathcal{N}_h^{\downarrow} \geq 2,\ 2 \leq j \leq \mathcal{N}_h^{\downarrow}. \end{cases} \quad (4)$$

Let B_H and $B_h \subseteq B_H$ be the BS groups serving train set H and the BS group serving train h, respectively. Train h finishes offloading task set K_h and finishes receiving computation results R_h^{rec} during h's journey in the group B_h. In the rest of the paper, we consider each BS group B_h as a multicast group. A multicast group contains one source BS and a certain number of destination BSs. Thus B_h can be expressed by

$$B_h = \{s_h, ..., B_h^{\downarrow}\} = \{s_h, ..., b_h^1, ...b_h^j, ..., b_h^{1+\mathcal{N}_h^{\downarrow}}\}. \quad (5)$$

Notice that train h will pass these BSs in order and it takes time to route the task from s_h to the computing cloudlet and route the result from the computing cloudlet to b_h^1. In this period, the train is moving and BSs located between s_h and b_h^1 may not play any role. However, in some special cases, a BS may both play the role of source BS s_h and the first destination BS b_h^1 in the group B_h. For train h, we define the *source BS group* of h as the BS group where source BS s_h is located, and the *destination BS group* of h is defined as the BS group where any destination BS $b \in D_h$ is located. A BS group can be both the source BS group and a destination BS group for the same train.

2) Group Steiner Tree: Constructing a multi-group shared multicast tree needs to answer the following questions: *"At least how many paths are needed to connect each group with the root vertex?"* and *"In each group, which BSs are suitable to be the endpoints of the paths that connect the group with the tree's root vertex?"* Therefore, we model the multi-group shared multicast tree as a group Steiner tree (GST) with edge and vertex weights [38]. The GST problem can be described as follows: Given an undirected graph $G = (V, E)$, each edge $e \in E$ has weight w_e and each vertex $v \in V$ has weight w_v, given subsets of vertices $\{g_i\}_k$, each subset g_i is called a group. The goal is to find the minimum cost tree T in G that contains at least one vertex from each group g_i, and the obtained tree T is mentioned as the *Group Steiner Tree*. Such a model can answer the mentioned questions well.

Let T_K be the group Steiner tree, which connects computing cloudlet v_K with BS groups B_H. Clearly, T_K is rooted at v_K. Tree T_K is used to route the data traffic of both the task set $K_H(t)$ and the result set R_H on the MEC network. Considering that G and T_K are undirected graphs, we thus denote $p_h = \langle B_h, ..., v_K \rangle$ as the simple path that connects a BS in the group B_h with root vertex v_K in T_K. We take train h as an example, the routing paths of the task set K_h and the result set R_h involve (i) routing the traffic of task set K_h from the source BS group B_h to the chosen computing cloudlet v_K via p_h, and (ii) routing the data traffic of result set R_h from computing cloudlet v_K to each destination BS group $B_{h'}$ via a path $p_{h'}$, where $h' \in H_h^{des}$. T_K can be expressed by

$$T_K = \sum\nolimits_{h=1}^{|H|} \left(p_h \cup \sum\nolimits_{h'=1}^{|H_h^{des}|} p_{h'} \right). \quad (6)$$

B. Delays and Costs in MEC Networks

All passengers achieve the best QoE when the delay experienced by task set $K_H(t)$ and result set R_H in the multicast tree meets the end-to-end delay requirement of every task $k_{i,h}$. Meanwhile, passengers must pay to obtain services by routing tasks and results on the MEC network. In the following, we define various delays and costs in the MEC network.

1) Delay experienced by Tasks and Computation Results: The total delay experienced by each task $k_{i,h}$ and its result set $R_{i,h}$ in G consists of (i) the *computation delay* in the computing cloudlet, (ii) the *routing delay* on links, and (iii) the *delay corresponding to the BSs*. We define the *network delay* as the sum of the computation delay and the routing delay.

Computation Delay: Let $d_{i,h}^{com}$ be the delay of computing task $k_{i,h}$ in cloudlet v_K. $d_{i,h}^{com}$ can be expressed by

$$d_{i,h}^{com} = f_{i,h}/F_v, \quad (7)$$

where $f_{i,h}$ and F_v are the demanded CPU cycles of task $k_{i,h}$ and each container's CPU cycles per second in v_K, respectively.

Routing Delay: Let d_e and d_v be the delay on link $e \in E$ and router $v \in V$ for transmitting a unit of data associated with a passenger, respectively. Let $d_{i,h}^{rou}$ be the routing delay of task $k_{i,h} \in T_K$, including the delay of routing the task on

path $p_h \subseteq T_K$ and the delay of routing result $r_{i,h,h'}$ on each path $p_{h'} \subseteq T_K$, where $h' \in H_h^{des}$. $d_{i,h}^{rou}$ is calculated with

$$d_{i,h}^{rou} = \left(\sum_{e \in p_h} d_e + \sum_{v \in p_h} d_v \right) \cdot z(k_{i,h}) + \max_{h' \in H_h^{des}} \left\{ \left(\sum_{e \in p_{h'}} d_e + \sum_{v \in p_{h'}} d_v \right) \cdot z(r_{i,h,h'}) \right\}. \tag{8}$$

Delay Associated with BSs: Consider that the distance between BSs in the same group is much smaller than the safe distance between trains, and a BS group only serves one train during its trip across the BS group. Thus, compared with the delay of data transmission on wired links, the delay of the direct communication between BSs in the same group can be neglected. For each BS b_j^h in group $B_h^{\downarrow}$, let $d_{h,b}^{\downarrow}$ be the delay of train h receiving volume $\mathcal{U}(b_h^j)$ of a portion of the result set R_h^{rec}. According to Eq. (1) and Eq. (4), $d_{h,b}^{\downarrow}$ equals $2 \cdot \gamma / v_h$, where $b = b_j^h$ and $2 \leq j \leq \mathcal{N}_h^{\downarrow}$. We temporarily default the formulas for calculating the wireless delay of BS b_h^1 and $b_h^{1+\mathcal{N}_h^{\downarrow}}$, since which two BSs in the group B_h respectively plays the role of b_h^1 and $b_h^{1+\mathcal{N}_h^{\downarrow}}$ need to be determined. A detailed description is given in subsection VI-A. Furthermore, the delay for train h fully receiving the data of R_h^{rec} is

$$d_h^{\downarrow} = \sum\nolimits_{b=1}^{|B_h^{\downarrow}|} d_{h,b}^{\downarrow}. \tag{9}$$

Then the total wireless download delay for downloading result set R_H can be represented by

$$d_H^{\downarrow} = \max\nolimits_{h \in H} \ d_h^{\downarrow}. \tag{10}$$

Network Delay: In summary, according to Eq. (7) and Eq. (8), we define network delay $d_{i,h}^{net}$ for task $k_{i,h}$ by the following formula

$$d_{i,h}^{net} = d_{i,h}^{rou} + d_{i,h}^{com}, \tag{11}$$

and the network delay of the task set $K_H(t)$ is

$$d_H^{net} = \max_{h \in H} \left\{ \max_{1 \leq i \leq |K_h|} d_{i,h}^{net} \right\}. \tag{12}$$

Total Delay: Then, the total delay experienced by task set $K_H(t)$ and result set R_H can be specified as

$$d_H^{total} = d_H^{net} + d_H^{\downarrow}. \tag{13}$$

Furthermore, the total delay cannot be greater than the delay requirement of any task $k_{i,h} \in K_H(t)$, i.e.,

$$d_H^{total} \leq d_{req}. \tag{14}$$

2) Admission cost: The admission cost of a task in G is the sum of the following costs: the processing cost in the computing cloudlet, the routing cost of forwarding the data traffic corresponding to the task, and its computation results along links and routers in the multicast tree. Denote by $c(v)$ and $c(e)$ the costs of one unit of data traffic that consumes the resource on router $v \in V$ and link $e \in E$, respectively. Meanwhile, we denote $c(v_K)$ as the processing cost for computing one unit of data in one container of computing cloudlet v_K. First, denote by $c_{i,h}^{pro}$ the processing cost of task $k_{i,h}$, which can be calculated by

$$c_{i,h}^{pro} = c(v_K) \cdot z(k_{i,h}). \tag{15}$$

Then denote $c_{i,h}^{rou}$ as the routing cost of transferring task $k_{i,h}$ along path p_h to processing cloudlet v_K, i.e.,

$$c_{i,h}^{rou} = \left(\sum_{v \in p_h} c(v) + \sum_{e \in p_h} c(e) \right) \cdot z(k_{i,h}), \tag{16}$$

and let $c_{i,h,h'}^{mul}$ be the cost of multicasting result $r_{i,h,h'}$ in the multicast tree. $c_{i,h,h'}^{mul}$ can be obtained by

$$c_{i,h,h'}^{mul} = \left(\sum_{v \in p_{h'}} c(v) + \sum_{e \in p_{h'}} c(e) \right) \cdot z(r_{i,h,h'}), \tag{17}$$

where $p_h, p_{h'} \subseteq T_K$, and T_K is the feasible multicast routing tree for routing $k_{i,h}$ and its computation results. Let $C_{i,h}$ be the admission cost of admitting task $k_{i,h}$

$$C_{i,h} = c_{i,h}^{pro} + c_{i,h}^{rou} + \sum_{h' \in H_{i,h}^{des}} c_{i,h,h'}^{mul}. \tag{18}$$

Denote by C_H the total cost of admitting task set $K_H(t)$, and C_H can be expressed by

$$C_H = \sum_{h=1}^{|H|} \sum_{i=1}^{|K_h|} C_{i,h}. \tag{19}$$

In practice, MEC service providers need to set their budgets for network operations to make money. Let β be the budget for network operations to admit tasks and multicast results. Thus, total cost C_H must satisfy $C_H \leq \beta$.

V. Problem Definition and an ILP Formulation

Given a track-side MEC network $G = (V, E)$, a set of trains H making a one-dimensional linear motion, each train maintains a safe distance. A set of tasks offloaded by the trains, which arrived at the BSs with the same end-to-end delay requirement. The set of results is multicast to the designated trains. *The delay-aware multicast-oriented throughput maximization problem* in G is to maximize the number of multicast results by admitting as many tasks as possible and finding a group Steiner tree, rooted at a feasible computing cloudlet, such that this tree (i) routes the data traffic of tasks from the source BSs to the root for computation, (ii) then routes the data traffic of the computation results from the root to the designated destination trains, and (iii) the sum of the usage costs of the routers, links, and the computing resources in the tree are minimum while meeting the delay requirement and the resource demands of each task, subject to the resource constraints in G and the motion of the trains.

The problem we define is an NP-hard problem, which is dictated by its special case: By reducing to the Steiner tree problem on graphs, the traditional multicast routing problem that does not take into account resource and delay constraints has been shown to be NP-hard problems [39]. We formulate the problem as the Integer Linear Programming (ILP, additional details are in the supplement [27]). ILP can be computationally

intensive and slow to solve large-scale problems. The process of solving ILP is unintuitive, making it difficult to understand and interpret. For example, it does not inherently provide mechanisms for reallocating tasks, adjusting routing paths, or rejecting tasks/results if the ILP solution is infeasible. The remainder of the paper will develop a series of efficient and extensible procedures and algorithms to solve the problem.

VI. PROPOSED METHOD

The proposed method consists of five phases. **1)** Construct the BS groups for a train set H and reject the tasks that cause the construction faults; **2)** Construct a GST for the groups constructed in phase one, which is rooted at a given cloudlet, without considering the delay requirements of the tasks and the capacity constraints of the computing cloudlet; **3)** Adjust the GST obtained from phase two in a series of auxiliary graphs derived from G, considering the delay and resource constraints that are not taken into consideration in phase two. Reject tasks and results which cause the faults of adjustment; **4)** Construct the bipartite graph to describe the priority relationships between the tasks and results that are routed in the adjusted GST of phase three, and the problem can be reduced to a breadth-first traversing in the bipartite graph; **5)** For all the cloudlets in G, the maximum throughput can be obtained by iteratively applying the procedures in phases two, three, and four.

A. BS Group Construction

The key point for constructing BS group B_h is determining which BSs play the roles of b_h^1 and $b_h^{1+\mathcal{N}_h^\downarrow}$, while the BS playing the role of s_h can be determined. An intuitive idea is that we create a BS group B_h in an extreme case, where no delay requirement will be violated during the period that a train h is covered by each BS in B_h. The created BS group can surely meet the delay requirements in a more general case. The location of train h at moment t is $l(h)$. Train h finishes downloading all results in R_h^{rec} at moment $t + d_{req}$, before h leaves the coverage area of the last BS in B_h. We consider the BS that covers location $l(h)+d_{req}\cdot\nu_h$ as the last BS $b_h^{1+\mathcal{N}_h^\downarrow}$ in group B_h. The data of R_h^{rec} must arrive at a BS in B_h when train h is within the coverage area of b_h^1. We infer that the period during which the train h travels from location $l(h)$ to the location where h begins to receive the data of R_h^{rec}, can be considered as the maximum tolerable network delay d_H^{net}. Let $\widetilde{d_H^{net}}$ be the maximum tolerable value of d_H^{net}. According to Eq. (1) and Eq. (4), the relationship is expressed as follows

$$\begin{aligned} z(R_h^{rec}) = &\int_{l(h)+\widetilde{d_H^{net}}\cdot\nu_h}^{l(b_h^1)+\gamma} C_{b_h^1}^{\downarrow}(x)\cdot\frac{|l(b_h^1)-x|}{\nu_h}dx + \sum_{j=2}^{\mathcal{N}_h^\downarrow} u_{\max} \\ &+ \int_{l(b_h^{1+\mathcal{N}_h^\downarrow})-\gamma}^{l(h)+d_{req}\cdot\nu_h} C_{b_h^{1+\mathcal{N}_h^\downarrow}}^{\downarrow}(x)\cdot\frac{|l(b_h^{1+\mathcal{N}_h^\downarrow})-x|}{\nu_h}dx. \end{aligned} \tag{20}$$

Since $z(R_h^{rec})$, $l(h)$, $l(b_h^1)$, $l(b_h^{1+\mathcal{N}_h^\downarrow})$, d_{req}, ν_h, u_{max} are known, $\widetilde{d_H^{net}}$ can be uniquely determined by the numerical integration described in Section III-C.

B. GST Construction and Adjustment

1) GST construction: Given a cloudlet $v_T \in V_c$ and we assume for a moment that v_T has enough computing capacity. Then a group Steiner tree T_v, which is rooted at v_T, is found in G by using Sun's $|H|$-approximation algorithm [40] such that T_v contains at least one BS of each BS group and the sum of the costs of one unit of resources at vertices and edges in this tree is minimized, without considering the delay requirements of any task in $K_H(t)$.

2) GST Feasibility: We assume that T_v is a feasible multicast tree for the task set $K_H(t)$ and the result set R_H, and the expression of T_v is Eq. (6). Let $K_H^v(t) \subseteq K_H(t)$ and $R_H^v \subseteq R_H$ be the tasks and results that are routed on the tree T_v. Let $p_{h,h'} = p_h \cup p_{h'}$ be the path that passes through root v_T and connects a group B_h and another group $B_{h'}$ in T_v, where $p_h = \langle B_h, ..., v\rangle$. Denote by $P_H = \bigcup_{h=1}^{|H|}\bigcup_{h'=1}^{|H|} p_{h,h'}$ the set of all the paths mentioned in T_v. For each path $p_{h,h'} \in P_H$, we find the tasks that (i) are offloaded by train h and their results are transmitted to h', or (ii) are offloaded by train h' and their results are transmitted to h. Notice that h' equals h when h is both the source train and the destination train for a task. This set of tasks is denoted as $K_p \subseteq K_H^v(t)$. Therefore, the total number $|P_H|$ of paths that are used to route task set $K_H^v(t)$ in T_v is $\frac{(|H|+1)\cdot|H|}{2}$. According to the definition of network delay, delay $d_{i,h}^p$ experienced by task $k_{i,h} \in K_p$ and its result on path $p_{h,h'}$ can be calculated by rewriting Eq. (11) as

$$\begin{aligned} d_{i,h}^p = &\left(\sum_{e\in p_h} d_e + \sum_{v\in p_h} d_v\right)\cdot z(k_{i,h}) \\ &+ \left(\sum_{e\in p_{h'}} d_e + \sum_{v\in p_{h'}} d_v\right)\cdot z(r_{i,h,h'}) + f_{i,h}/F_v. \end{aligned} \tag{21}$$

We introduce a function $d(\cdot)$ to calculate the maximum delay experienced by the data routing of a path or a set of paths. $d(p_{h,h'})$ thus is the maximum delay experienced by task set K_p and its result set on path $p_{h,h'}$, i.e., $d(p_{h,h'}) = \max_{k_{i,h}\in K_p} d_{i,h}^p$. According to Eq. (12), the delay experienced by task set $K_H^v(t)$ and result set R_H^v in T_v can be calculated by $d(T_v)$, i.e., $d(T_v) = \max_{p_{h,h'}\subseteq T_v} d(p_{h,h'})$. If $d(T_v) \le \widetilde{d_H^{net}}$, tree T_v is a feasible solution. Otherwise, tree T_v needs to be adjusted.

3) GST Adjustment in an auxiliary graph: An intuitive idea of GST adjustment is iteratively adjusting every path in P_H until there is no path that violates delay $\widetilde{d_H^{net}}$. A detailed description is given in Procedure 1.

We sort all the paths in P_H in decreasing order of the maximum delay experienced by the tasks and the results on each path (Lines 5-6, Procedure 1). Without loss of generality, suppose that path $p_{h,h'}$ has the maximum delay in T_v $p_{h,h'} = \arg\max_{p_{h,h'}\subseteq T_v} d(T_v)$. Clearly, $d(T_v)$ equals $d(p_{h,h'})$ and $d(p_{h,h'}) > \widetilde{d_H^{net}}$. Thus, path $p_{h,h'}$ is the *bottleneck path* for transmitting task set K_p and its result set in T_v. A delay-constrained-least-cost (DCLC) path, which connects group B_h and group $B_{h'}$, needs to be found to replace $p_{h,h'}$, such that the obtained DCLC path meets delay $\widetilde{d_H^{net}}$, and the costs of data routing on the new path are also minimized. However, planning such a path connecting two groups in an undirected

Procedure 1: Adjusting the Group Steiner Tree with Delay Constraints

Input: A network $G=(V,E)$, group Steiner tree T_v rooted at v_T in G, auxiliary graph $G_v=(V_v,E_v)$ with respect to G and v_T, delay constraint $\widetilde{d_H^{net}}$, BS group $B_H=\{B_1,...,B_h,...,B_{|H|}\}$, Task set $K_H(t)$, Result set R_H.

Output: (i) Adjusted tree T_v, such that $d(T_v)\le\widetilde{d_H^{net}}$; (ii) Task set $K_H^v(t)$ and result set R_H^v

$loop\leftarrow 1$, $K_H^v(t)\leftarrow K_H(t), R_H^v\leftarrow R_H$;
$p_{h,h'}^{dclc}, Q_{delay}\leftarrow\emptyset$; /*$Q_{delay}$ is a max-heap*/
foreach $h\in H$ **do**
  Find path p_h in T_v, and $h'\leftarrow h$;
  while $h'\le|H|$ **do**
    $h'\leftarrow h'+1$, find path $p_{h'}$ in T_v and construct path $p_{h,h'}\leftarrow p_h\cup p_{h'}$, $P_H\leftarrow P_H\cup\{p_{h,h'}\}$, $Insert\ (Q_{delay},p_{h,h'})$; /*Construct set P_H and sort the paths in P_H in decreasing order of their delay*/
    Find task set K_p and calculate routing delay $d_{i,h}^p$ for each task $k_{i,h}\in K_p$, sort task set K_p in increasing order by the delay of each task;
  end
end
while $loop\le|P_H|$ **do**
  $p_{h,h'}\leftarrow ExtractMax\ (Q_{delay})$; /*Get the path with the maximum delay connecting group B_h and B'_h*/
  if $d(p_{h,h'})\le\widetilde{d_H^{net}}$ **OR** $loop>|P_H|$ **then**
    The adjustment is finished, **return** $T_v, K_H^v(t), R_H^v$;
  else if $d(p_{h,h'})>\widetilde{d_H^{net}}$ **then**
    Find the least cost path $p_{h,h'}^{dclc}$ between dummy vertices x'_h and $x''_{h'}$ with delay constraint $\widetilde{d_H^{net}}$ by using Juttner's algorithm [41] in G_v;
    if $p_{h,h'}^{dclc}$ *exists* **then**
      Path $p_{h,h'}^{adj}=p_h^{adj}\cup p_{h'}^{adj}$ in G, which connects group B_h and $B_{h'}$, is derived from $p_{h,h'}^{dclc}$ in G_v;
      Examine the feasibility of p_h^{adj} and $p_{h'}^{adj}$ as follows. For each $p_{h'}\in J_h$, If $d(p_h^{adj}\cup p_{h'})>\widetilde{d_H^{net}}$, then p_h^{adj} is NOT FEASIBLE, otherwise, p_h^{adj} is FEASIBLE.
    end
    if p_h^{adj} *and* $p_{h'}^{adj}$ *are both feasible* **then**
      Replace path $p_{h,h'}$ in T_v with $p_{h,h'}^{adj}$, $J_h\leftarrow J_h\cup\{p_{h'}^{adj}\}$, $J_{h'}\leftarrow J_{h'}\cup\{p_h^{adj}\}$, $loop\leftarrow loop+1$;
    end
    if *(i)* $p_{h,h'}^{dclc}$ *does not exist* **OR** *(ii)* p_h^{adj} *or* $p_{h'}^{adj}$ *is not feasible* **then**
      For each task $k_{i,h}\in K_p$, whose delay $d_{i,h}^p$ is greater than $\widetilde{d_H^{net}}$, REJECT result $r_{i,h,h'}$ of this task, which corresponds to $d_{i,h}^p$, $R_H^v\leftarrow R_H^v\backslash\{r_{i,h,h'}\}$. If the rejected result $r_{i,h,h'}$ is the only result of task $k_{i,h}$, then REJECT task $k_{i,h}$, $K_H^v(t)\leftarrow K_H^v(t)\backslash\{k_{i,h}\}$. $loop\leftarrow loop+1$;
    end
  end
  if $loop\le|P_H|$ **then**
    If $p_h\ne p_h^{adj}$, then remove p_h from Q_{delay}, $Insert\ (Q_{delay},p_h^{adj})$;
    If $p_{h'}\ne p_{h'}^{adj}$, then remove p'_h from Q_{delay}, $Insert\ (Q_{delay},p_{h'}^{adj})$;
  end
end

graph G, when h' equals h, is difficult to solve. Another problem is determining which vertices of the two groups are the terminals of the DCLC path. Next, we transform such a problem into finding the DCLC path in an auxiliary graph, which is described as follows. Denote by $G_v=(V_v,E_v)$ the auxiliary graph for $v_T\in G$. First, we construct graph $G'=(V',E')$ by augmenting G. For each group B_h, we add a dummy vertex x'_h into B' and every dummy vertex cost and delay are set to zero. Then, for each BS $b\in B_h$, an undirected dummy edge $\langle b,x'_h\rangle$ with neither cost nor delay is added into E', i.e., $B'=\bigcup_{h=1}^{|H|}\{x'_h\}\cup B$, and $E'=\bigcup_{h=1}^{|H|}\bigcup_{b\in B_h}\{\langle b,x'_h\rangle\}\cup E$. Second, we construct graph $G''=(V'',E'')$, which is a copy of $G'=(V',E')$. For each $v'\in V'$, and $e'\in E'$, node v'', and edge e'' are added into V'', E'', respectively. Especially, dummy node $x'_h\in G'$ corresponds to dummy node $x''_h\in G''$. Last, each node and each edge in G' and G'' are added into V_v, E_v, respectively. Specially, for root vertex v_T in G, the corresponding vertices in G' and G'' are denoted as v'_T and v''_T. Then, an undirected edge $\langle v'_T,v''_T\rangle$ is added into E_v, where the edge cost is set to the cost of vertex v_T and the edge delay equals the computation delay of using v_T. i.e., $V_v=V'\cup V''$, $E_v=E'\cup\{\langle v'_T,v''_T\rangle\}\cup E''$, and $G_v=G'\cup\{\langle v'_T,v''_T\rangle\}\cup G''$. An example is given in Fig. 2.

In each iteration, Juttner's $(1+\epsilon)$-approximation algo-

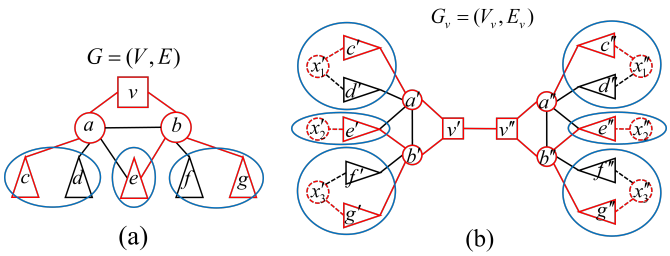


Fig. 2: An example of the auxiliary graph. (a) The original graph, where v is the computing cloudlet, vertices a and b are the routers, vertices c, d, e, f, g are the BSs, the vertices enclosed by a blue cycle make up a BS group, the red lines and red vertices represent the GST rooted at v; (b) The auxiliary graph, where the red lines and red vertices represent the extended GST, and the dummy node and the dummy edges are added to each BS group in the auxiliary graph.

rithm [41] is used to find the DCLC path $p^{dclc}_{h,h'}$ in G_v (Line 15, Procedure 1), which connects vertex x'_h with vertex $x''_{h'}$ and routes task set K_p and its result set. Let $p^{adj}_{h,h'} = p^{adj}_h \cup p^{adj}_{h'}$ be the corresponding path of $p^{dclc}_{h,h'}$ in G, where path p^{adj}_h and path $p^{adj}_{h'}$ are the adjusted paths that connect group B_h with root v_T and group $B_{h'}$ with root v_T, respectively. Then p^{adj}_h and $p^{adj}_{h'}$ are used to replace p_h and $p_{h'}$ in T_v, respectively. However, there may exist multiple paths that violate delay $\widetilde{d^{net}_H}$ in P_H. In each iteration, only one path of $p_{h,h'}$ will be adjusted. Consider that $p_{h,h'} = p_h \cup p_{h'}$ is a combination of path p_h and path $p_{h'}$, therefore adjusted path $p^{adj}_{h,h'} = p^{adj}_h \cup p^{adj}_{h'}$ can be regarded as a joint adjustment of path p_h and path $p_{h'}$. That is, the delay of the adjusted path of p_h in the current iteration may not be feasible for the combinations in the previous iterations. Therefore, the feasibility of the adjusted paths must be examined and a set J_h is maintained to record the historical feasible paths corresponding to B_h in each iteration (Line 18, Procedure 1).

C. Rejection Policy for Tasks and Results in Two Phases

The phase of BS group construction: There may exist a special case when the calculated $\widetilde{d^{net}_H}$ leads to a paradox as follows, if source BS s_h plays the role of b^1_h, and train h receives the results before it enters the coverage area of s_h. Such a paradox can be described by the following formula,

$$l(h) + \widetilde{d^{net}_H} \cdot \nu_h < l(s_h) - \gamma. \tag{22}$$

If the value of $z(R^{rec}_h)$ is too large, then $\widetilde{d^{net}_H}$ may not be feasible. Group B_h will not be created unless results with large data volume are rejected. $\widetilde{d^{net}_H}$ has a lower bound, and it needs to satisfy $\widetilde{d^{net}_H} \geq (l(s_h) - \gamma - l(h))/\nu_h$. Therefore, the value of $z(R^{rec}_h)$ also has an upper bound, which is denoted by $z(\widetilde{R^{rec}_h})$, and it satisfies

$$\begin{aligned} z(\widetilde{R^{rec}_h}) \leq & \int_{l(s_h)-\gamma}^{l(s_h)+\gamma} C^{\downarrow}_{s_h}(x) \cdot \frac{|l(s_h)-x|}{\nu_h} dx + \sum_{j=2}^{\mathcal{N}^{\downarrow}_h} u_{\max} \\ & + \int_{l(b_h^{1+\mathcal{N}^{\downarrow}_h})-\gamma}^{l(h)+d_{req}\cdot\nu_h} C^{\downarrow}_{b_h^{1+\mathcal{N}^{\downarrow}_h}}(x) \cdot \frac{|l(b_h^{1+\mathcal{N}^{\downarrow}_h})-x|}{\nu_h} dx. \end{aligned} \tag{23}$$

The results of R^{rec}_h are carefully selected to make up $\widetilde{R^{rec}_h}$ such that the total volume is less than or equal to $z(\widetilde{R^{rec}_h})$. Such a selection process is a 0-1 knapsack problem, where let $w_{i,h,h'} \in \{0,1\}$ be the binary decision variable to determine whether $r_{i,h,h'}$ is selected to make up set $\widetilde{R^{rec}_h}$. This problem can be solved using Lawler's fast approximation algorithm [42]. The result set R_H can be derived from set $\widetilde{R^{rec}_h}$. If any results of task $k_{i,h}$ do not exist in R_H after the above process, then task $k_{i,h}$ is also removed from $K_H(t)$.

The phase of GST adjustment: In each iteration of adjusting a certain GST T_v (Lines 10-31, Procedure 1), if the DCLC path between dummy vertices x'_h and $x''_{h'}$ cannot be found or the adjusted path is not feasible for an original path $p_{h,h'} \in T_v$ (Line 23, Procedure 1), it indicates that path $p_{h,h'}$ is the bottleneck path of tree T_v. Therefore, the eligibility of tasks / results for passing through $p_{h,h'}$ must be examined. In other words, some tasks/results need to be rejected to reduce latency on path $p_{h,h'}$. The strategy of task/result rejection (Line 24, Procedure 1) is as follows. First, We find task set $K_p \subseteq K^v_H(t)$ that passes through the path $p_{h,h'}$. For each task $k_{i,h} \in K_p$, calculate delay $d^p_{i,h}$ according to Eq. (21). If $d^p_{i,h} > \widetilde{d^{net}_H}$, we reject result $r_{i,h,h'}$, which passes through path $p_{i,h}$. The result $r_{i,h,h'}$ is removed from the set R^v_H. When all the results of a task have been rejected, then this task also needs to be rejected and removed from the task set $K^v_H(t)$.

D. A Graph-based Heuristic for Admitting Tasks and Multicasting Results in a GST

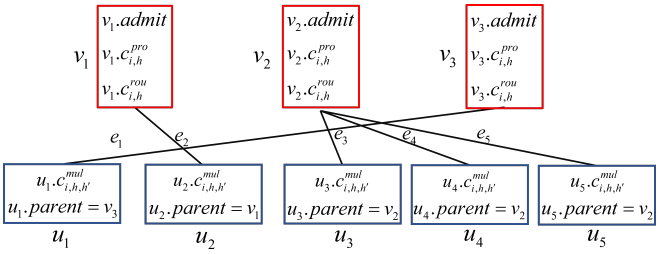


Fig. 3: An example of a bipartite graph $G_b = (V_b, U_b, E_b)$, where node set V_b corresponds to tasks, $v_1.c^{pro}_{i,h} + v_1.c^{rou}_{i,h} \leq v_2.c^{pro}_{i,h} + v_2.c^{rou}_{i,h} \leq v_3.c^{pro}_{i,h} + v_3.c^{rou}_{i,h}$; Node set U_b corresponds to results, $u_1.c^{mul}_{i,h,h'} \leq u_2.c^{mul}_{i,h,h'} \leq u_3.c^{mul}_{i,h,h'} \leq u_4.c^{mul}_{i,h,h'} \leq u_5.c^{mul}_{i,h,h'}$; Edge set E_b represents the priority relationships between the tasks and their results.

There is no need to consider the delay constraints during the procedure of task admission and result multicasting when the multicast tree is determined. We propose a graph-based algorithm to multicast results as many as possible by admitting as many tasks as possible in a GST T_v. We observed that *a task precedes any of its results*, which means that a result can be multicast if and only if its corresponding task has been admitted. Thus, we describe such a priority relationship between task set $K^v_H(t)$ and result set R^v_H by constructing a bipartite graph $G_b = (V_b, U_b, E_b)$. We sort set $K^v_H(t)$ in increasing order of the sum of the processing cost and the routing cost of each task $k_{i,h} \in K^v_H(t)$, then we sort set R^v_H in increasing order of the multicasting cost of each result $r_{i,h,h'} \in R^v_H$. For each task $k_{i,h}$ in the sorted set $K^v_H(t)$, construct node $v_b \in V_b$ with three

Procedure 2: Obtaining the Set of Admitted Tasks and Multicasted Results by Traversing a Given Bipartite Graph

Input: Bipartite graph $G_b = (V_b, U_b, E_b)$
Output: Admitted task set AK and Multicasted result set MR

$TempCost \leftarrow 0$;
$AK, MR \leftarrow \emptyset$; /*$AK$ is the set of admitted tasks and MR is the set of multicasted results*/
foreach *node* $u_b \in U_b$ **do**
  if $u_b.parent.admit = REJECT$ **then**
    CONTINUE to do the next loop;
  else if $u_b.parent.admit = FALSE$ **then**
    if $|AK| \geq |C_v|$ **then**
      BREAK the loop;
    else
      if $TempCost + u_b.parent.c_{i,h}^{rou} + u_b.parent.c_{i,h}^{pro} > \beta$ **then**
        $u_b.parent.admit \leftarrow REJECT$;
      else
        $u_b.parent.admit \leftarrow TRUE$;
        $|AK| \leftarrow u_b.parent$;
        $TempCost \leftarrow TempCost + u_b.parent.c_{i,h}^{rou} + u_b.parent.c_{i,h}^{pro}$;
      end
    end
  end
  if $TempCost + u_b.c_{i,h,h'}^{mul} \leq \beta$ **then**
    $|MR| \leftarrow u_b$;
    $TempCost \leftarrow TempCost + u_b.c_{i,h,h'}^{mul}$;
  else
    BREAK the loop;
  end
end
return AK, MR;

attributes of admission status $v_b.admit$, processing cost $v_b.c_{i,h}^{pro}$ and routing cost $v_b.c_{i,h}^{rou}$. $v_b.admit$ is initialized to $FALSE$, which means that v_b has not been visited and the corresponding task needs to be examined to be admitted. If $v_b.admit$ is set to $REJECT$, it means that the corresponding task is rejected. If $v_b.admit$ is set to $TRUE$, then the corresponding task is admitted. For each result $r_{i,h,h'}$ in the sorted set R_H^v, it constructs node $u_b \in U_b$ with two attributes of multicast cost $u_b.c_{i,h,h'}^{mul}$ and parent $u_b.parent$. If task $k_{i,h}$ corresponds to node v_b and result $r_{i,h,h'} \in R_{i,h}$ corresponds to node u_b, then an edge $e_b \in E_b$ is added between v_b and u_b, where $u_b.parent$ is set to v_b. For ease of description, we implicitly assume that (i) $v_b.c_{i,h}^{pro} + v_b.c_{i,h}^{rou} \leq v_{b'}.c_{i,h}^{pro} + v_{b'}.c_{i,h}^{rou}$ holds if node $v_{b'}$ is on the right side of node v_b; (ii) $u_b.c_{i,h,h'}^{mul} \leq u_{b'}.c_{i,h,h'}^{mul}$ holds if node $u_{b'}$ is on the right side of node u_b; (iii) The nodes in set U_b are at the bottom of the nodes in set V_b. The mentioned procedure continues until all tasks and results have been assigned to the bipartite graph G_b. An example of the mentioned bipartite graph is given in Fig. 3.

Then, maximizing the number of multicasted results in a given GST T_v has been transformed into traversing as many nodes in set U_b in bipartite graph $G_b = (V_b, U_b, E_b)$ as possible, constrained by the budget β and cloudlet capacity C_v. We traverse G_b with left-right and bottom-up strategy, and the details are given in Procedure 2. It first initialized two sets, which are named AK and MR (Line 2, Procedure 2). The nodes corresponding to the admitted tasks are put into AK, and the nodes corresponding to the multicasted results are put into MR. Then we start the traverse from the node, whose $c_{i,h,h'}^{mul}$ is the minimum among all nodes in U_b and every node

Algorithm 1: The HeuAlg

Input: a MEC network $G = (V, E)$, cloudlet set $V_c \subseteq V$, delay constraint $\widetilde{d_H^{met}}$, task set $K_H(t)$, result set R_H
Output: Determine which tasks in set $K_H(t)$ can be admitted and which results in set R_H can be multicasted, return the corresponding multicast tree T_K

$T_K, Q, K_H^v(t), R_H^v \leftarrow \emptyset$;/*Q is a regular queue*/
$TempThroughput$, $MaxThroughput \leftarrow 0$;
Create BS groups B_H and REJECT the tasks and results that caused the construction faults. Then remove the rejected tasks and results from $K_H(t)$ and R_H, respectively;
foreach $v_c \in V_c$ **do**
  Construct and store auxiliary graph $G_v = (V_v, E_v)$. Then find group Steiner Tree T_v, which is rooted at v_c in G, by invoking Sun's $|H|$-approximation algorithm [40];
  $ENQUEUE(Q, T_v)$;
end
while Q ! $= \emptyset$ **do**
  $T_v \leftarrow DEQUEUE(Q)$;
  if $d(T_v) > \widetilde{d_H^{met}}$ **then**
    Adjust T_v and obtain set $K_H^v(t)$, R_H^v by invoking Procedure 1.
  else
    $K_H^v(t) \leftarrow K_H(t)$, $R_H^v \leftarrow R_H$;
  end
  Sort set $K_H^v(t)$ in increasing order of the sum of processing cost $c_{i,h}^{pro}$ and routing cost $c_{i,h}^{rou}$ of each task $k_{i,h} \in K_H^v(t)$;
  Sort set R_H^v in increasing order of multicasting cost $c_{i,h}^{mul}$ of each result $r_{i,h,h'} \in R_H^v$;
  Construct a bipartite graph $G_b = (V_b, U_b, E_b)$ by using set $K_H^v(t)$ and set R_H^v;
  Admit tasks in set AK and multicast results in set MR by invoking Procedure 2. $TempThrougput$ equals the cardinality of set MR;
  if $TempThroughput > MaxThrougput$ **then**
    $MaxThrougput \leftarrow TempThroughput$;
    $T_K \leftarrow T_v$, $K_H(t) \leftarrow K_H^v(t)$, $R_H \leftarrow R_H^v$;
  end
end
return $MaxThrougput$, T_K, $K_H(t)$, R_H;

$u_b \in U_b$ is processed as follows. (i) If the admission status of the parent node of $u_b \in U_b$ is $REJECT$, then u_b will not be multicast, and it continues to check the feasibility of the node on the right side of u_b. (Lines 4-5, Procedure 2). (ii) If the parent node of $u_b \in U_b$ has not been visited and the number of admitted tasks is greater than or equal to capacity C_v, it stops traversing G_b (Lines 7-8, Procedure 2). Otherwise, it examines the parent node of u_b with the sum of the processing cost and routing cost (Lines 9-16, Procedure 2). the parent node of u_b can be put into AK if the admission status is $TRUE$ (Lines 13-14, Procedure 2). (iii) The condition of Line 19 can only be verified if the parent node of $u_b \in U_b$ has been admitted, i.e., node u_b can be put into MR if the total admission cost is less than or equal to the budget β. Otherwise, it stops traversing G_b (Lines 19-24, Procedure 2).

E. Algorithm and Analysis

1) Throughput Maximization Algorithm: By iterative application of the aforementioned procedures, we have the algorithm to solve the network throughput maximization problem, which is given in HeuAlg 1. We construct the set B_H of BS groups and reject the tasks and results causing the faults of construction (Line 4, HeuAlg 1) by applying the procedures described in subsection VI-A and subsection VI-C. Then we construct GST T_v for each cloudlet in set V_c and store the trees in queue Q (Lines 5-8, HeuAlg 1). For each tree T_v in Q (Line 9, HeuAlg 1), we adjust the tree by invoking Procedure 1. Task set $K_H^v(t)$ and result set R_H^v, which are routed in tree T_v, are obtained (Lines 10-15, HeuAlg 1). By applying Procedure 2, we obtain the throughput of each tree T_v (Lines 16-19, HeuAlg 1). Last, the maximum throughput of G is equal to the throughput of the tree in queue Q that has the maximum number of multicasted results (Lines 20-22, HeuAlg 1).

2) Theoretical Conclusions and Analysis: Due to the limited space, we list the important conclusions in the main text as follows, and leave the details of the lemmas and theorems and their analysis in the supplement [27].

Theorem 1. *Procedure 2 has a relative performance guarantee of $1/2$.*

Theorem 2. *Given a train set H with the initial locations and velocities of the trains, a MEC network $G = (V, E)$ with a set V of routers, a subset B of BSs, a subset $V_c \subseteq V$ of cloudlets, and a set E of links, task set $K_H(t)$ and their multicast-oriented computation results with delay requirements and resource demands, there is an algorithm* `HeuAlg` *for the network throughput maximization problem, which takes*

$$O\left(\begin{array}{l} |H|^3|V|\left(|K_H(t)|\left(|V|+|E|\right)+\log|H|\right) \\ +|H|^2|V|\left(|K_H(t)|\log|K_H(t)|+|E|^2\log^4|E|\right) \\ +|V|\left(|H||V|\log|V|+|R_H|\log|R_H|\right) \\ +|R_H|\log(1/\zeta^3)+1/\zeta^4+|B| \end{array}\right) \text{ time,}$$

where ζ is a constant with $\zeta > 0$.

VII. Performance Evaluation

A. Experiment Settings

The network topologies were generated using the GT-ITM tool [43]. Wireless communication parameters, listed in the Supplement [27], were set according to [17], [44]–[47]. The computing capacity of each cloudlet ranged from 100 to 500 [48], and the CPU cycles in a cloudlet were set at 1.5 GHz [49]. Routing costs per task/result along a link were randomly set from 0.01 to 0.1 [50], and the cost of a router to route a task/result varied from 0.05 to 0.1 [28]. The processing cost for a cloudlet to compute a task was randomly drawn from 0.5 to 2 [51], and the delay on a link or in a router ranged from 1 millisecond (ms) to 10 ms [52]. Each task $k_{i,h}$ and its result set $R_{i,h}$ were generated as follows: A task randomly chose its destination set $H_{i,h}^{des}$ from set H of trains. Task volume $z(k_{i,h})$ was uniformly distributed from 0.01MB to 3MB [53], and the demanded CPU cycles $f_{i,h}$ varied from 500Mhz to 2000Mhz [49]. The results were obtained on a machine with a 3.2 GHz Intel i7 6-core CPU and 32GB RAM. Each value in the figures was obtained by taking the mean value of 50 trials.

B. Baselines

Inspired by the benchmark designs in [28], [51], the performance of HeuAlg was evaluated against six baselines: MinCost, MinDelay, RandomSelect, TradeoffSteiner, DelayNFV and DelaySPT. Specially, the path-finding strategies of MinCost, MinDelay and TradeoffSteiner are based on the calculations as follows. The calculation of weight $w_\lambda(e)$ of edge $e \in E$ and weight $w_\lambda(v)$ of node $v \in V$ in a network G need to be done by the following formulas, $w_\lambda(e) = \lambda \cdot c(e) + (1-\lambda) \cdot d(e)$, and $w_\lambda(v) = \lambda \cdot c(v) + (1-\lambda) \cdot d(v)$, where λ is the regulated weight, $c(v)$ and $c(e)$ are the original costs of one unit of data on link e and node v, respectively. $d(e)$ and $d(v)$ are the original delays on link e and node v, respectively. Then, all the baselines always choose the cloudlet with sufficient capacity and minimum cost, where the mentioned formula also calculates the cost with regulated weight λ. Algorithms MinCost, MinDelay, TradeoffSteiner, DelayNFV, and DelaySPT randomly select the BSs from the BS groups constructed by algorithm HeuAlg. The details of all the baselines are as follows.

- **MinCost:** MinCost is a greedy algorithm for unicasting. When a pair of users on different trains need to interact, MinCost needs to find the least weight path using Dijkstra's algorithm [54], and such a path is from a source BS to a specified destination BS for each task and its result, passing through the selected cloudlet. The value of λ for MinCost is set to 1, then finding the routing path with the least sum of the weight of edges and nodes for each task and its result.
- **MinDelay:** Similar to MinCost, MinDelay is also a unicast algorithm. The value of λ for MinDelay is set to 0, which means that MinDelay always routes tasks and results along the path with minimum delay. MinDelay also finds the path with the least weight using Dijkstra's algorithm [54].
- **DelaySPT:** Building on the baseline design in [55], DelaySPT employs a heuristic approach to construct the delay-constrained shortest path tree using Lichen's algorithm [56] for unicast paths to each multicast destination. Each unicast path connects a source BS to a designated destination BS for a specific task and its result, traversing through a chosen cloudlet while balancing cost and delay.
- **UniMax:** UniMax [17] is a unicast-based algorithm to route each task and its result on the MEC network for HSRs.

TABLE II: Different System Parameter Settings

Areas	$\|H\|$	ν_h	$\|B\|$	$\|V\backslash B\|$	$\|V_c\|/\|V\|$	$c(v)$	$c(e)$	$\rho_{i,h}$	d_{req}
Urban	3	[0,40]	25	35	0.5	[0.05,0.075]	[0.01,0.05]	(6,8]	1000
Rural	6	[70,90]	60	10	0.1	[0.075,0.1]	[0.05,0.1]	[2,4]	5000

In line with the idea of finding the root of GST in this paper, UniMax traverses each possible cloudlet in the network and uses the cloudlet as a computing cloudlet to plan the routing path through the cloudlet for each pair of task and result.

• **TradeoffSteiner:** TradeoffSteiner needs to find a Steiner tree with Charikar's algorithm [57] for routing all the tasks and results. Such a tree is rooted at the min-cost cloudlet and spans all the given BSs. λ for the TradeoffSteiner is set to 0.5, which means the path-finding strategy of the algorithm TradeoffSteiner makes a trade-off between the costs and the delays.

• **RandomSelect:** RandomSelect randomly chooses a cloudlet with sufficient capacity as the computing cloudlet, i.e., RandomSelect constructs only one GST, which is rooted at the randomly chosen cloudlet, such that the network throughput is calculated based on this GST only. The other strategies of RandomSelect, including the GST adjustment, rejection policy, and admission policy, are as same as HeuAlg.

• **DelayNFV:** The DelayNFV algorithm [28] was originally designed to find the routing paths for a set of NFV-enabled multicast requests with a delay requirement. DelayNFV's path planning uses a Steiner tree-based algorithm, and we adapt the algorithm to the specific context of this study. Specifically, the computing cloudlet served as the only instance required by the corresponding multicast request.

Based on the above description, we further classify the baseline into Steiner-based algorithms and shortest-path-based algorithms. The Steiner-based algorithms consist of Tradeoff-Steiner, RandomSelect, and DelayNFV. HeuAlg also belongs to the Steiner-based algorithm. The shortest-path-based algorithms consists of MinCost, MinDelay, DelaySPT, and UniMax.

C. Evaluation Metrics

Different algorithms were evaluated using several metrics, and the definition of the metrics in this article are listed as follows.

1) Quality of Computing Service (QoCS): The QoCS is defined as the ratio of the admitted tasks, K_A, to the total tasks that have arrived, $K_H(t)$. This metric shows how efficiently the algorithm allocates the computing resources of the cloudlets within the network.

$$QoCS = |K_A|/|K_H(t)|,$$

where $K_A \subseteq K_H(t)$.

2) Efficiency of Multicast (EoM): EoM is defined as the ratio between the results received by each train h and the total results of admitted tasks. It's important to recognize that an admitted task does not guarantee multicast to the specified destination, thus reflecting the multicast transmission efficiency of the algorithm.

$$EoM = \sum\nolimits_{h=1}^{H} |R_h^{rec}| \Big/ |R_A|,$$

where $R_A \subseteq R_H$ is the results of the admitted tasks.

3) Throughput: Throughput in the MEC network refers to the total computation results received by each train h.

$$Throughput = \sum\nolimits_{h=1}^{H} |R_h^{rec}|.$$

4) Average Service Delay (ASD): ASD is the mean value of the end-to-end delay experienced by each admitted task $k_{j,h} \in K_A$ and its corresponding results in the network.

$$ASD = \sum\nolimits_{k_{j,h} \in K_A} \left(d_{j,h}^{rou} + d_{j,h}^{com} + d_h^{\downarrow}\right) \Big/ |K_A|.$$

5) Operation Cost (OC): OC consists of the sum of the routing cost for each task $k_{i,h} \in K_H(t)$, the processing cost for each admitted task $k_{j,h} \in K_A$, and the multicasting cost for each result $r_{j,h,h'} \in R_A$ to the designated trains.

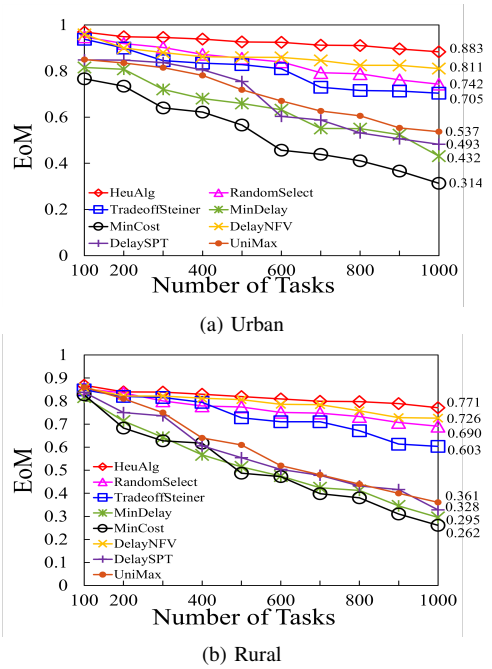
$$OC = \sum\nolimits_{k_{i,h} \in K_A} c_{i,h}^{rou} + \sum\nolimits_{k_{j,h} \in K_A} c_{j,h}^{pro} + \sum\nolimits_{r_{j,h,h'} \in R_A} c_{j,h,h'}^{mul}.$$

D. Discussion on Results

We evaluate the performance of HeuAlg in comparison to baseline algorithms under various system parameters, as outlined in Table II. We consider two typical areas traversed by trains: urban and rural areas. Based on realistic timetables [58], [59], we set the safety distance between neighboring trains at a minimum of 18 km. Train speeds are adjusted according to the area, with urban areas having slower speeds of 0-40 m/s and rural areas at 70-90 m/s. For our discussion, we consider a 50 km track section and a 120 km track section in the urban and rural areas, respectively. Thus, the number of BSs $|B|$ in different areas is set to 25 and 60, respectively. The train numbers $|H|$ are set to 6 and 3 for urban and rural areas, respectively. The network infrastructure varies between different areas, leading to differences in edge network conditions. In the urban area, the edge network is based on the city's backbone network, providing abundant network resources with a large number $|V|$ of nodes and a high density $|V_c|/|V|$ of cloudlets. As trains travel through rural areas, the number of network nodes and the density $|V_c|/|V|$ of cloudlets decrease, resulting in relatively insufficient network resources. Taking into account the network conditions in different areas, tasks offloaded by users exhibit distinct characteristics. In urban areas, users prefer offloading tasks with high data volumes of results and low delay requirements. In contrast, in rural areas, users prefer offloading tasks with low data volumes of results and high delay requirements. The ratio $\rho_{i,h}$ is set according to [60], while the delay requirement d_{req} (millisecond) is set according to [28].

1) Analysis of Efficiency of Multicast: Fig. 4 illustrates the EoM performance of different algorithms. The EoM decreases as the number of tasks admitted increases in all areas, a trend that is particularly pronounced in algorithms such as MinCost and MinDelay. From Fig. 4, it is evident that the HeuAlg algorithm outperforms the other baselines in terms of EoM

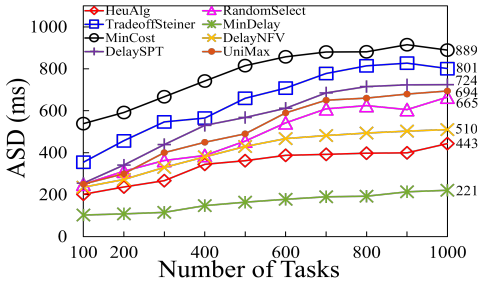
This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3357769



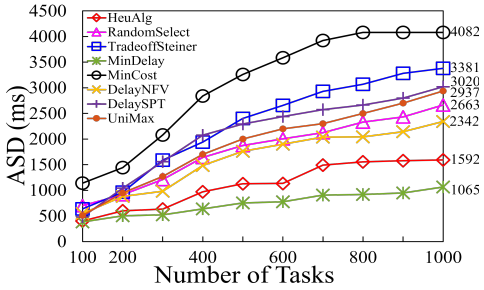
(a) Urban

(b) Rural

Fig. 4: EoM of different algorithms in urban and rural areas.



(a) Urban



(b) Rural

Fig. 5: ASD of various algorithms in urban and rural areas.

in different areas. Specifically, Fig. 4(a) shows the EoM of different algorithms in the urban area. The difference between Steiner-based algorithms and shortest-path-based algorithms increases as the task number increases. At a task number of 1000, the EoM of the HeuAlg algorithm is 8.9%, 19.0%, 25.2%, 64.4%, 79.1%, 104.3%, and 181.2% higher than that of DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost, respectively. Fig. 4(b) depicts the EoM of different algorithms in the rural area. Fig. 4(b) also shows that the EoM of HeuAlg, DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost in the rural area are lower than their EoM in the urban area by 10.3%-12.7%, 10.3%-10.4%, 7.0%-10.1%, 9.6%-14.4%, 1.1%-32.8%, 1.2%-33.5%, 0.2%-31.7% and 0.1%-16.6%, respectively.

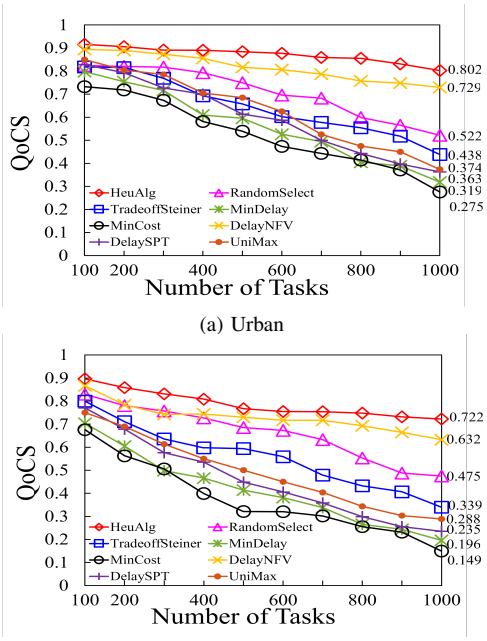
This is because HeuAlg considers the constraints to determine whether they will affect the feasibility of the multicast path. Thus, the results to be multicast are checked individually at both the BS group construction and the GST adjustment. This approach maximizes the number of transmittable results on the multicast path, which improves the efficiency of transmission.

2) Analysis of Average Service Delay: Fig. 5(a) illustrates the average service delay of HeuAlg in the urban environment. The delay curve of HeuAlg outperforms the other existing baselines. When the task number is set to 1000, excluding the MinDelay algorithm, the delay for HeuAlg is lesser than the other baselines by 13.1%–50.1%. The urban setting introduces more complexity, but HeuAlg's ability to adapt and manage resources ensures optimal performance. On the other hand, Fig. 5(b) illustrates the average service delay of HeuAlg with varying numbers of users in the rural environment. We observe that the delay curve of HeuAlg maintains a lower profile compared to other existing baselines, such as RandomSelect, TradeoffSteiner, MinDelay, MinCost, DelayNFV, DelaySPT and UniMax. With the increase in the number of users, the system faces higher demand, leading to increased delay. However, HeuAlg optimally handles this growth, resulting in a lower delay than other baselines (excluding the MinDelay algorithm) by 32.0%–52.9% when the task number is at 1000. The algorithm's efficiency in managing resources contributes to this performance.

The reason is that the baselines take significant amount of time to route tasks to and process them within the appropriate cloudlet during task offloading. Taking into account the delay constraints, task routing and computation time shrink the time available for result routing, and it is necessary to reject certain results to meet these delay requirements. HeuAlg therefore employs an auxiliary graph construction technique to plan task routing, computation, and subsequent result routing. This approach helps to efficiently address the tradeoff between the durations of these three phases under inherent delay constraints.

3) Analysis of Quality of Computing Service: The QoCS performance of HeuAlg is analyzed in comparison to the baseline algorithms: MinCost, MinDelay, TradeoffSteniner, RandomSelect, DelayNFV, DelaySPT and UniMax. As observed in Fig. 6, HeuAlg outperforms the other baselines in terms of QoCS in varying traversing areas. Fig. 6(a) illustrates the QoCS of different algorithms in the urban area. The figure shows that the QoCS decreases as the number of arrived tasks increases. The difference in QoCS among the algorithms is not significant when the task number is low (≤ 200). However, as the task number increases, the effectiveness of Steiner tree-based algorithms, such as HeuAlg and DelayNFV, on QoCS becomes more pronounced. At a task number of 1000, the QoCS of the HeuAlg algorithm is 10.0%, 53.6% 83.1%, 114.4%, 120.9%, 151.4%, 191.6% higher than that of DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost, respectively. Fig. 6(b) presents the QoCS of different algorithms in the rural area. The QoCS still decreases as the task number increases. Furthermore, the

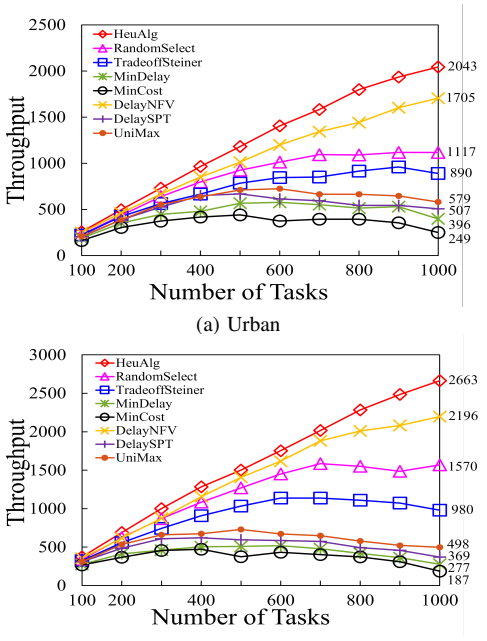
This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3357769



(a) Urban

(b) Rural

Fig. 6: QoCS of different algorithms in urban and rural areas.



(a) Urban

(b) Rural

Fig. 7: Throughput of algorithms in urban and rural areas.

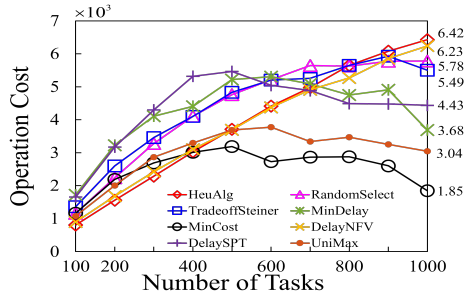
QoCS of HeuAlg, DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost in the rural area are 2.2%-9.9%, 3.1%-13.3%, 0.01%-9.0%, 2.2%-22.6%, 10.7%-22.9%, 7.0%-35.3%, 11.8%-38.6%, and 7.7%-45.8% lower than their QoCS in the urban area, respectively. The rationale behind this is that QoCS is primarily affected by the selection of the computing cloudlet and the path from the BS group to the computing cloudlet.

As mentioned above, HeuAlg integrates task routing into a comprehensive path planning strategy. Meanwhile, HeuAlg also considers each cloudlet with sufficient resources in the network as a potential candidate and then applies path adjustment and rejection policies to determine the most appropriate computing cloudlet. Consequently, the computing cloud determined by HeuAlg outperforms all baselines.

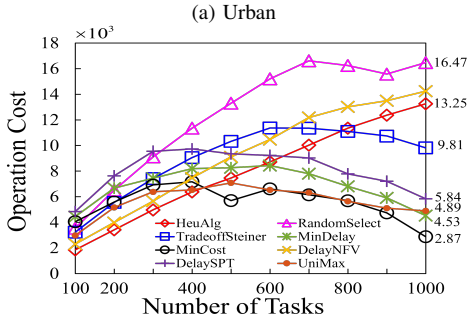
4) Analysis of Throughput: Fig. 7 illustrates the throughput performance of HeuAlg in comparison with other baseline algorithms in both rural and urban environments. In both settings, HeuAlg exhibits a significant advantage, outperforming other algorithms across various numbers of tasks. Fig. 7(a) depicts the throughput of HeuAlg against the baselines in the urban area. Among them, the throughput of HeuAlg and DelayNFV algorithms increases with the number of tasks, the throughput of RandomSelect and TradeoffSteiner gradually converge to a certain value as the number of tasks increases, whereas the throughput curves of the shortest-path-based algorithms show a trend of first increasing and then leveling off with the number of tasks. Meanwhile, HeuAlg has a significant advantage over other baseline algorithms as the number of tasks increases. For example, when the number of tasks is 1000, the throughput of HeuAlg is 19. 8%, 82.9%, 129.6%, 252.8%, 303.0%, 415.9%, and 720.5% higher than that of DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay and MinCost, respectively. Fig. 7(b) shows the throughput of different algorithms in the rural area, and the trends on the throughput curves of algorithms in Fig. 7(b) are similar to that of Fig. 7(a). And as the number of tasks increases, the difference between the throughput of the Steiner-based algorithms and the throughput of shortest-path-based algorithms becomes larger and larger. For example, when the number of tasks is 1000, the throughput of HeuAlg is 21.2%, 69.6%, 171.7%, 434.7%, 621.7%, 861.4%, and 1324.1% higher than that of DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay and MinCost, respectively. Also, in Fig. 7(b) we notice that the difference between HeuAlg's throughput and the baselines' tends to increase further if the number of tasks increases further. Considering that the number of trains involved in multicast communication is higher in the rural area than in the urban area, it is not statistically significant to compare the throughput of the same algorithm in the rural area with that in the urban area.

These trends result from HeuAlg including the steps to identify and adjust the bottleneck paths in Procedure 1 when modifying the candidate GST. Consequently, HeuAlg can dynamically modify the GST structure in response to various task numbers, resource demands, and delay requirements, thus improving multicast throughput.

5) Analysis of Operation Cost: Fig. 8(a) illustrates the variance in operation costs of different algorithms in the urban area. As seen in the figure, the operation costs of all Steiner-based algorithms gradually increase with the task number. Combined with the throughput depicted in Fig. 7, this suggests that Steiner-based algorithms are not significantly influenced by throughput in terms of operation cost. In contrast, the operation costs for shortest-path-based algorithms initially escalate sharply with increasing the task number, followed by a slow decline when the task number is between 500 and 1000. This reaffirms our conclusion from Fig. 7 that shortest-path-based algorithms encounter a performance bottleneck in terms of throughput. Consider the following example



(a) Urban



(b) Rural

Fig. 8: Operation cost of algorithms in urban and rural areas.

where the budget is set at 6000 in an urban area. When HeuAlg, DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost are applied with respective costs of 800, 900, 1000, 900, 600, 500, 600, and 500, none of them exceed this budget. In this case, the throughputs for these algorithms are 1798, 1601, 1117, 958, 725, 667, 576, and 440, respectively. Fig. 8(b) presents the operation costs of different algorithms in a rural area, where the trend is similar to that of Fig. 8(a). The operation costs of Steiner-based algorithms continue to increase steadily with the task number, while the operation costs of shortest-path-based algorithms demonstrate an initial increase due to performance bottlenecks, followed by a decreasing trend as the task number increases. Consider the following example where the budget is set at 11000 in a rural area. When HeuAlg, DelayNFV, RandomSelect, TradeoffSteniner, UniMax, DelaySPT, MinDelay, and MinCost are applied with respective costs of 700, 600, 300, 900, 500, 400, 600, and 400, none of them exceed this budget. In this case, the throughputs for these algorithms are 2017, 1616, 869, 1072, 729, 620, 519, and 471, respectively. It is obvious from these results that HeuAlg outperforms the others by achieving the highest throughput in both urban and rural areas.

The reason is that HeuAlg's routing tree structure, which uses GST, is highly efficient. In scenarios with high task density or low delay requirements, GST ensures that only one base station within a multicast group needs to connect to a small cloud, significantly reducing routing overhead. On the contrary, other algorithms require connecting all designated base stations to the cloudlet, regardless of the conditions. This distinction emphasizes HeuAlg's operational efficiency and cost effectiveness in complex network environments.

Synthesis: The above analysis demonstrates that the Steiner-based algorithm significantly outperforms the shortest-path-based algorithm in multicast scenarios and therefore the unicast algorithm cannot be directly applicable to solving multicast problems. In particular, HeuAlg shows its superior and robust performance in various environments. It offers detailed insights into the adaptability of different algorithms to fluctuating task numbers and environmental conditions. This emphasizes its efficacy in routing computational outcomes to designated locations while adhering to delay constraints. Although latency optimization is not a primary objective of this study, HeuAlg reduces service latency compared to other baseline algorithms in both urban and rural settings. Its ability to adapt to diverse situations and efficiently allocate resources contributes significantly to its improved performance. Additionally, a review of Figures 7 and 8 reveals that HeuAlg effectively balances operational costs. This balance is crucial to maximize network throughput, particularly when operators face budget constraints.

VIII. CONCLUSION

This paper studies the throughput maximization problem for multicasting the results of the offloaded delay-aware tasks in a track-side MEC network in the snapshot scenario, where the tasks are from the same application. We propose a heuristic algorithm by progressively considering the constraints. Tasks are admitted as many as possible such that as many results are multicast to the specified destinations as possible, the resource demands and delay requirements of each task are met, while the total cost of the multicast routing tree is minimized, subject to various resource capacities in the network. The simulation results show that the proposed algorithm is suitable for multicasting the results to passengers on trains in both urban and rural areas.

REFERENCES

[1] X. Wang and H. Liy, "Content delivery for high-speed railway via integrated terrestrial-satellite networks," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2020.

[2] M. Gao and B. Ai, "Efficient hybrid beamforming with anti-blockage design for high-speed railway communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9643–9655, 2020.

[3] Y. Deng, Z. Chen, X. Chen, and Y. Fang, "Throughput maximization for multiedge multiuser edge computing systems," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 68–79, 2021.

[4] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput maximization of delay-aware dnn inference in edge computing by exploring dnn model partitioning and inference parallelism," *IEEE Transactions on Mobile Computing*, 2021.

[5] X. Deng, J. Li, and L. Shi, "Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2271–2288, 2020.

[6] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Bandwidth gain from mobile edge computing and caching in wireless multicast systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3992–4007, 2020.

[7] H. Hao, C. Xu, S. Yang, and L. Zhong, "Multicast-aware optimization for resource allocation with edge computing and caching," *Journal of Network and Computer Applications*, vol. 193, pp. 103–195, 2021.

[8] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3864–3872, 2019.

[9] A. Samanta, T. G. Nguyen, T. Ha, and S. Mumtaz, "Distributed resource distribution and offloading for resource-agnostic microservices in industrial iot," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1184–1195, 2022.

[10] B. Li, J. Xiong, and B. Liu, "Cache-based popular services pushing on high-speed train by using converged broadcasting and cellular networks," *IEEE Transactions on Broadcasting*, vol. 65, no. 3, pp. 577–588, 2018.

[11] J. Xiong, H. Xie, B. Liu, B. Li, and L. Gui, "Cooperative caching services on high-speed train by reverse auction," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9437–9449, 2021.

[12] H. Chen and Z. Li, "Edge computing-aided framework of fault detection for traction control systems in high-speed trains," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1309–1318, 2020.

[13] Y. Liu and W. Li, "Vehicular edge computing model for fault detection and diagnosis of high-speed train," in *2020 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[14] Q. Zhang, H. Zheng, and Z. Zhong, "Energy-aware dynamic computation offloading in high-speed railway networks with d-tdd," in *2020 IEEE 92nd Vehicular Technology Conference*, pp. 1–6, IEEE, 2020.

[15] H. Li, Y. Sun, Y. Zhang, B. Jin, Z. Wang, W. Wu, and C. Fang, "Mobility-aware predictive computation offloading and task scheduling for mobile edge computing networks," in *2021 7th International Conference on Computer and Communications (ICCC)*, pp. 1349–1354, IEEE, 2021.

[16] L. Li, Y. Niu, S. Mao, and B. Ai, "Resource allocation and computation offloading in a millimeter-wave train-ground network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 10, pp. 10615–10630, 2022.

[17] J. Xu, Z. Wei, Z. Lyu, L. Shi, and J. Han, "Throughput maximization of offloading tasks in multi-access edge computing networks for high-speed railways," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9525–9539, 2021.

[18] S. Roger, D. Martin-Sacristan, D. Garcia-Roger, J. F. Monserrat, P. Spapis, A. Kousaridas, and S. Ayaz, "Low-latency layer-2-based multicast scheme for localized v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 2962–2975, 2018.

[19] H. Bao, C. Huang, Z. Tang, Q. Li, Q. Ni, X. Dong, and B. Fu, "Coded multicasting in cache-enabled vehicular ad hoc network," *Computer Networks*, vol. 159, pp. 157–170, 2019.

[20] P. Keshavamurthy, E. Pateromichelakis, D. Dahlhaus, and C. Zhou, "Cloud-enabled radio resource management for co-operative driving vehicular networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2019.

[21] A. J. Kadhim and S. A. H. Seno, "Energy-efficient multicast routing protocol based on sdn and fog computing for vehicular networks," *Ad Hoc Networks*, vol. 84, pp. 68–81, 2019.

[22] Y. Hui, Z. Su, and T. H. Luan, "Collaborative content delivery in software-defined heterogeneous vehicular networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 575–587, 2020.

[23] P. Keshavamurthy, E. Pateromichelakis, D. Dahlhaus, and C. Zhou, "Resource scheduling for v2v communications in co-operative automated driving," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2020.

[24] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6164–6174, 2020.

[25] X. Dai, H. Zhao, S. Yu, D. Cui, Q. Zhang, and H. Dong, "Dynamic scheduling, operation control and their integration in high-speed railways: A review of recent research," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 13994–14010, 2021.

[26] S. Li, L. Yang, and Z. Gao, "Distributed optimal control for multiple high-speed train movement: An alternating direction method of multipliers," *Automatica*, vol. 112, p. 108646, 2020.

[27] "The supplement." https://github.com/junyixu-git/MEC-HSR-Multicast/blob/main/supplemental.pdf, 2023.

[28] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu, "Efficient algorithms for delay-aware nfv-enabled multicasting in mobile edge clouds with resource sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2050–2066, 2020.

[29] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2017.

[30] H. Flores, P. Hui, P. Nurmi, E. Lagerspetz, S. Tarkoma, and J. Manner, "Evidence-aware mobile computational offloading," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1834–1850, 2018.

[31] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 393–413, 2014.

[32] Y. Mao, C. You, J. Zhang, and K. Huang, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[33] F. Gringoli, P. Serrano, and I. Ucar, "Experimental qoe evaluation of multicast video delivery over ieee 802.11 aa wlans," *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2549–2561, 2018.

[34] J. Park, J.-N. Hwang, and H.-Y. Wei, "Cross-layer optimization for vr video multicast systems," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 206–212, IEEE, 2018.

[35] Z. Zhang, M. Zeng, M. Chen, and D. Liu, "Joint user grouping, version selection, and bandwidth allocation for live video multicasting," *IEEE Transactions on Communications*, vol. 70, no. 1, pp. 350–365, 2021.

[36] P. J. Davis and P. Rabinowitz, *Methods of numerical integration*. Academic Press, Inc., 1984.

[37] Y.-D. Lin, Y.-C. Lai, H.-Y. Teng, and C.-C. Liao, "Scalable multicasting with multiple shared trees in software defined networking," *Journal of Network and Computer Applications*, vol. 78, pp. 125–133, 2017.

[38] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," in *2007 IEEE 23rd international conference on data engineering*, pp. 836–845, IEEE, 2007.

[39] C. A. Oliveira, P. M. Pardalos, and M. G. Resende, "Optimization problems in multicast tree construction," in *Handbook of optimization in telecommunications*, pp. 701–731, Springer, 2006.

[40] Y. Sun, X. Xiao, B. Cui, S. Halgamuge, and T. Lappas, "Finding group steiner trees in graphs with both vertex and edge weights," *Proceedings of the VLDB Endowment*, vol. 14, no. 7, pp. 1137–1149, 2021.

[41] A. Juttner and B. Szviatovski, "Lagrange relaxation based method for the qos routing problem," in *2001 IEEE International Conference on Computer Communications (INFOCOM)*, pp. 859–868, IEEE, 2001.

[42] E. L. Lawler, "Fast approximation algorithms for knapsack problems," *Mathematics of Operations Research*, vol. 4, no. 4, pp. 339–356, 1979.

[43] G. I. of Technology, "Gt-itm is a network topology generate tool." https://www.cc.gatech.edu/projects/gtitm/, 2022.

[44] L. Tian and J. Li, "Seamless dual-link handover scheme in broadband wireless communication systems for high-speed rail," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 4, pp. 708–718, 2012.

[45] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[46] C. Zhang, P. Fan, K. Xiong, and P. Fan, "Optimal power allocation with delay constraint for signal transmission from a moving train to base stations in high-speed railway scenarios," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5775–5788, 2015.

[47] L. Wang and B. Ai, "Energy-efficient power control of train–ground mmwave communication for high-speed trains," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7704–7714, 2019.

[48] D. Bruneo, "A stochastic model to investigate data center performance and qos in iaas cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, 2014.

[49] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[50] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Efficient nfv-enabled multicasting in sdns," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2052–2070, 2018.

[51] Y. Ma, W. Liang, J. Wu, and Z. Xu, "Throughput maximization of nfv-enabled multicasting in mobile edge cloud networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 2, pp. 393–407, 2019.

[52] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[53] T. Chen, H. Shan, and X. Wang, "Optimal scheduling for wireless on-demand data packet delivery to high-speed trains," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 9, pp. 4101–4112, 2014.

[54] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[55] X. Zhao, J. Guo, C. T. Chou, A. Misra, and S. K. Jha, "High-throughput reliable multicast in multi-hop wireless mesh networks," *IEEE Transactions on mobile computing*, vol. 14, no. 4, pp. 728–741, 2014.

[56] J. Lichen, L. Cai, J. Li, S. Liu, P. Pan, and W. Wang, "Delay-constrained minimum shortest path trees and related problems," *Theoretical Computer Science*, vol. 941, pp. 191–201, 2023.

[57] M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li, "Approximation algorithms for directed steiner problems," *Journal of Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.

[58] C. A. O. R. S. CORPORATION, "12306 china railway." https://www.12306.cn/en/index.html, 2023.

[59] C. J. R. Company, "Jr shinkansen timetable." https://global.jr-central.co.jp/en/info/timetable/, 2023.

[60] S. Vlahovic and M. Suznjevic, "The impact of network latency on gaming qoe for an fps vr game," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–3, IEEE, 2019.