Decentralized AI Service Placement, Selection and Routing in Mobile Networks

Jinkun Zhang Stefan Vlaski Kin Leung
Imperial College London, UK

***Abstract*—The rapid development and usage of large-scale AI models by mobile users will dominate the traffic load in future communication networks. The advent of AI technology also facilitates a decentralized AI ecosystem where small organizations or even individuals can host AI services. In such scenarios, AI service (models) placement, selection, and request routing decisions are tightly coupled, posing a challenging yet fundamental tradeoff between service quality and service latency, especially when considering user mobility. Existing solutions for related problems in mobile edge computing (MEC) and data-intensive networks fall short due to restrictive assumptions about network structure or user mobility. To bridge this gap, we propose a decentralized framework that jointly optimizes AI service placement, selection, and request routing. In the proposed framework, we use *traffic tunneling* to support user mobility without costly AI service migrations. To consider nonlinear queueing delays, we formulate a non-convex problem to optimize the tradeoff between service quality and the end-to-end latency. We derive the node-level KKT conditions and develop a decentralized Frank–Wolfe algorithm with a novel messaging protocol. Numerical evaluations are used to validate the proposed approach and show substantial performance improvements over existing methods.**

I. Introduction

The rapid adoption of AI services (e.g., OpenAI's GPT series) is fundamentally changing the traffic load and dynamics of modern communication networks. While AI services today are primarily offered by major companies, predictions (e.g., [1]) point towards a more decentralized future AI ecosystem, where small organizations or even individual users can host their own AI models, presumably in decentralized networks with flexible scales and arbitrary topologies.

This poses significant challenges for both users and networks. Users have the options of selecting from multiple pre-trained AI models offered by different providers. These models provide different levels of service quality (e.g., accuracy) and latency, requiring users to carefully select the one that best aligns with their preferences [2]. The network, on the other hand, should carefully place the models to keep network congestion and user latency under control. Recent studies on *AI as a network service* examine model placement and resource optimization under latency/accuracy goals [3], and selection across models with heterogeneous QoS [4], but most assume centralized control or limited topologies and do not target fully decentralized settings.

First, AI service placement and selection are very similar to those in mobile edge computing (MEC). In MEC, service selection focuses on choosing the most appropriate service instance to balance performance and efficiency under constraints such as latency, quality of service (QoS), and hardware limitations [5]. Service placement aims to provide popular services or network content close to users to reduce access delay and network load [6]. However, these MEC approaches generally rely on hierarchical control and are not designed for decentralized scenarios. Although another line of studies for content and computation placement support arbitrary decentralized topologies [7], [8], they assume pure static networks and overlook user mobility.

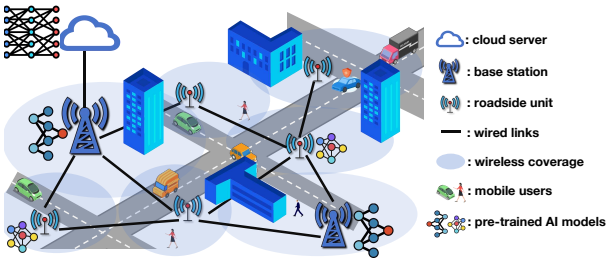


Fig. 1: An example edge-cloud vehicular network. Mobile users have multiple pre-train AI model options.

On the other hand, unlike traditional MEC, AI models introduce additional challenges. Most notably, the size of pre-trained AI models is growing exponentially (e.g., GPT-4 has roughly 1.8 trillion parameters), making it increasingly difficult for edge servers to host. In conventional MEC systems, when users move between wireless access points (APs), *service migration* is commonly employed to maintain seamless service delivery [9], [10]. It transfers the service instance and its runtime state to a server closer to the user's new location, assuming the service is lightweight enough to be moved. However, the cost of real-time migrating large AI models is considered impractical [11]. To address this, we adopt a more classic and realistic solution: *traffic tunneling* [12], where the user's original AP serves as an anchor. When the user moves, responses from the remote server are first routed back to this anchor node, which then forwards the results to the user's new location. This approach eliminates the need to migrate large-scale AI models, instead incurring overhead in the form of additional traffic flows. A positive feedback loop exists between request latency and tunneling flows, which may overload the network if not appropriately handled. To our knowledge, traffic tunneling-based AI service placement and selection has never been studied.

In this work, we address the above gaps by developing a novel framework that jointly optimizes AI service placement, selection, and request routing under user mobility. The proposed framework supports arbitrary network topologies and operates fully decentralized via traffic tunneling. A key contribution of our approach is the use of congestion-dependent

nonlinear costs, which can capture the crucial queueing effects on network links and processing units. Rather than modeling individual requests, we analyze the time-averaged system behavior under homogeneous assumptions. We tackle the non-convex optimization problem first with fixed service placement and then extend to the general case. We derive node-level Karush–Kuhn–Tucker (KKT) conditions, which indicate intuitively myopic node behavior. We then give decentralized online Frank–Wolfe-based algorithms that converge to these conditions via a novel messaging protocol to obtain gradients.

Our major contributions are summarized as follows:

- We propose a decentralized framework that jointly handles AI service placement, selection, and routing under user mobility using a tunneling mechanism.
- We formulate a utility-minus-cost optimization problem with congestion-aware costs, and derive node-level KKT conditions for both fixed placement and the general case.
- We design a decentralized messaging protocol and algorithms convergent to the KKT conditions, then validate their performance through numerical evaluations.

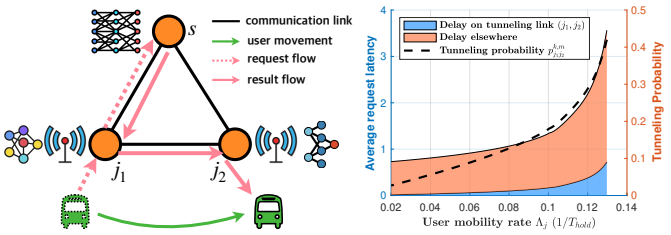
The paper first tackle the problem under fixed service placement in Section II and III, then extend to joint service placement in Section IV and provide numerical evaluation results in Section V. Due to space limit, we put the proofs of propositions and theorems in supplementary document [13].

II. Fixed AI Service Placement

A. AI-driven network with mobile users

Network and AI services. We consider a directed, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with arbitrary topology, where $\mathcal{V}$ are static nodes (e.g., APs, RSUs, edge servers), and $\mathcal{E}$ are links. For node i, let $\mathcal{N}_i$ denote i's neighbors $\{j : (i,j) \in \mathcal{E}\}$. Nodes in $\mathcal{V}$ have heterogeneous capabilities of communication with neighbors, hosting pre-trained AI models, and serving inference requests.Mobile users access the network by associating with nodes. Let $\mathcal{U}$ be the set of users, and assume user $u \in \mathcal{U}$ is associated with node $v_u(t)$ at time slot t. Communication and computation in the network are driven by a set of AI tasks $\mathcal{K}$. Every task $k \in \mathcal{K}$ can be fulfilled by a set of (different) pre-trained models $\mathcal{M}_k$ in the network. We define a *service* as a pair (k,m) for $k \in \mathcal{K}$ and $m \in \mathcal{M}_k$[1], and $\mathcal{S}$ be the set of services. In this section, we assume fixed service placement: each (k,m) is hosted by a known, non-empty set $\mathcal{X}_{k,m} \subseteq \mathcal{V}$.

Requests handling. During time slot t, user u issues a request for task k with probability $r_u^k(t) \in [0,1]$. For task k, user u can specify models in $\mathcal{M}_k$ by *model selection decision* $s_u^{k,m}(t) \in [0,1]$, i.e., a fraction of $s_u^{k,m}(t)$ of rate $r_u^k(t)$ is assigned to model $m \in \mathcal{M}_k$ with $\sum_{m\in\mathcal{M}_k} s_u^{k,m}(t) = 1$. After model selection, requests (k,m) for $m \neq 0$ are routed through the network in a hop-by-hop distributed manner to a node that hosts the service (i.e., in set $\mathcal{X}_{k,m}$). If the user remains stationary, the inference result is then delivered back along the reversed path of the request (we defer the mobility handling right after). The network routing is controlled by routing decisions $\phi_{ij}^{k,m}(t)$. Specifically, requests for (k,m) arrive at node i from two sources: (i) exogenous requests, issued by users currently associated with i, and (ii) endogenous requests, forwarded from neighboring nodes. Of all arrival (k,m)-requests, node i forwards a fraction of $\phi_{ij}^{k,m}(t) \in [0,1]$ to each neighbor $j \in \mathcal{N}_i$, with the flow conservation $\sum_{j\in\mathcal{N}_i} \phi_{ij}^{k,m}(t) = \mathbb{1}_{i\notin\mathcal{X}_{k,m}}$. Namely, if $i \in \mathcal{X}_{k,m}$, it provide service (k,m) and act as a sink of requests; otherwise, it forwards all arrival requests to neighbors. Such a hop-by-hop routing scheme aligns naturally with decentralized control and has been widely adopted in data- and computation-intensive networks [8], [14]. Conceptually similar to MoE gating [15], we consider choice-dependent utility, capturing heterogeneous AI outputs: for each fulfilled request of service (k,m), the user obtains a utility $u_{k,m} \geq 0$, reflecting the inference quality (e.g., accuracy or user satisfaction) of the selected model.



(a) Tunneling on (j_1, j_2) (b) Latency $D_{j_1,k,m}$ and $p_{j_1j_2}^{k,m}$

Fig. 2: Traffic tunneling and impact on request latency.

Traffic tunneling. Since large AI models are impractical to migrate in real time, we adopt *traffic tunneling* to handle user mobility. Suppose user u issues a request for service (k,m) at time t, the node $i = v_u(t)$ serves as an "anchor", so that the result is always sent back to i upon completion. Let $D_{i,k,m}^{\text{o}}$ denote the "static" round-trip latency of the request, measured from it being generated until the result returns to i. If the user has moved during this time, i.e., $v_u(t + D_{i,k,m}^{\text{o}}) = j \neq i$, then the anchor node i will forward the result to the new access point j to reach the user. We assume tunneling occurs over at most one hop, i.e., $v_u(t + D_{i,k,m}^{\text{o}}) \in \mathcal{N}_{v_u(t)}$.[2] Such tunneling scheme (Figure 2(a)) aligns with our decentralized setting, however, inducing a positive feedback loop: tunneling adds extra transmission, increasing overall latency, which in turn triggers more tunneling. We aim to manage the tunneling flow to stabilize and optimize the system.

B. Time-homogeneous formulation

Time-homogeneous network. We adopt time-homogeneous approximations to simplify the system dynamics. Assume the aggregated request rates received at each AP are quasi-static, and the request for k received at node i is time-invariant r_i^k,

$$\sum\nolimits_{u\in\mathcal{U}:v_u(t)=i} r_u^k(t) = r_i^k, \quad \forall t. \tag{1}$$

We also replace the time- and user-dependent service selection $s_u^{k,m}(t)$ with a node-based time-invariant variable $s_i^{k,m}$,

$$s_u^{k,m}(t) = s_i^{k,m}, \quad \forall u : v_u(t) = i, \ \forall t. \tag{2}$$

[1] $m = 0$ for lightweight local models (e.g., onboard obstacle detection).

[2] Multi-hop transitions are extremely rare in practice.

We denote by vector $\boldsymbol{s} = [s_i^{k,m}]$ the *global service selection variable*, with the following constraint holds,

$$\sum\nolimits_{m\in\mathcal{M}_k} s_i^{k,m} = 1, \quad \forall i \in \mathcal{V}, k \in \mathcal{K}. \tag{3}$$

Similarly, we use time-invariant routing variable $\phi_{ij}^{k,m}$ with

$$\phi_{ij}^{k,m} = \phi_{ij}^{k,m}(t), \quad \forall t, \tag{4}$$

and denote by vector $\boldsymbol{\phi} = [\phi_{ij}^{k,m}]$ the *global routing variable* with the flow conservation given by

$$\sum\nolimits_{j\in\mathcal{N}_i} \phi_{ij}^{k,m} = \mathbb{1}_{i\notin\mathcal{X}_{k,m}}, \quad \forall i \in \mathcal{V}, (k,m) \in \mathcal{S}. \tag{5}$$

We remark that in practice, $\boldsymbol{\phi}$ can be implemented by simple probabilistic request forwarding, i.e., when node i receives a request for (k,m), it forwards it to j with probability $\phi_{ij}^{k,m}$.

Service latency. Request latency is incurred at both traversed links and the service-hosting node. We assume both delays are dependent on the link flow or node workload, capturing the crucial queueing effect. Let $f_{ij}^{k,m}$ be the steady-state request rate of service (k,m) on link $(i,j) \in \mathcal{E}$, given by

$$f_{ij}^{k,m} = \phi_{ij}^{k,m} t_i^{k,m}, \tag{6}$$

where $t_i^{k,m}$ is the total received request rate for (k,m) at node i, given recursively by

$$t_i^{k,m} = r_i^k s_i^{k,m} + \sum\nolimits_{j\in\mathcal{N}_i} f_{ji}^{k,m}. \tag{7}$$

Let $L_{k,m}^{\text{req}}$ and $L_{k,m}^{\text{res}}$ be the size of request and result packets, respectively, then the total flow rate on (i,j) is

$$F_{ij} = F_{ij}^{\text{o}} + F_{ij}^{\text{tun}}, \tag{8}$$

where F_{ij}^{o} is the *static flow* given by

$$F_{ij}^{\text{o}} = \sum\nolimits_{(k,m)\in\mathcal{S}} \left(L_{k,m}^{\text{req}} f_{ij}^{k,m} + L_{k,m}^{\text{res}} f_{ji}^{k,m} \right), \tag{9}$$

and F_{ij}^{tun} is the *tunneling flow* (extra flow due to traffic tunneling), given later in (16). We denote the expected packet delay on link (i,j) by $d_{ij}(F_{ij})$, where $d_{ij}(\cdot)$ is non-decreasing and convex. e.g., the average M/M/1 queue sojourn time [16]

$$d_{ij} = 1/(\mu_{ij} - F_{ij}), \tag{10}$$

where μ_{ij} is the service rate (i.e., capacity), given $F_{ij} < \mu_{ij}$.

Let $W_{k,m}$ be the single-request computation workload for (k,m), the total workload at node i is

$$G_i = \sum\nolimits_{(k,m):i\in\mathcal{X}_{k,m}} W_{k,m} t_i^{k,m}, \quad \forall i \in \mathcal{V} \tag{11}$$

Similarly, let $c_i(G_i)$ be the expected request delay at node i, where $c_i(\cdot)$ is non-decreasing convex, incorporating delay due to computation processing and congestion effect.

The end-to-end latency of a request comprises three parts: (i) request transmission delay, (ii) delay at the service-hosting node, and (iii) result transmission delay (on the reversed path). Thus, the static round-trip delay $D_{i,k,m}^{\text{o}}$ is given by

$$\sum\nolimits_{p\in\mathcal{P}_i^{k,m}} \mathbb{P}_p \left(c_{p_{|p|}} + \sum\nolimits_{\ell=1}^{|p|-1} \left(d_{p_\ell p_{\ell+1}} + d_{p_{\ell+1} p_\ell} \right) \right) + d_{\text{AP}},$$

where p is a *routing path*, i.e., node sequence $(p_1, p_2, \cdots, p_{|p|})$ with $\phi_{p_l p_{l+1}} > 0$; set $\mathcal{P}_i^{k,m}$ denotes all paths for service (k,m) starts from i, i.e., $p_1 = i$, $p_{|p|} \in \mathcal{X}_{k,m}$; constant d_{AP} is the user-AP wireless access delay;and $\mathbb{P}_p = \prod_{l=1}^{|p|} \phi_{p_l p_{l+1}}$ is the probability that path p is taken under $\boldsymbol{\phi}$.

Moreover, the expected end-to-end latency of service (k,m) issued originally at i is:

$$D_{i,k,m} = D_{i,k,m}^{\text{o}} + \sum\nolimits_{j\in\mathcal{N}_i} p_{ij}^{k,m} d_{ij}, \tag{12}$$

where $p_{ij}^{k,m}$ is the *tunneling probability*, i.e., the chance that user moved to j during time $D_{i,k,m}^{\text{o}}$, given later in (15). [3]

Tunneling flow. We assume users have homogeneous movement patterns with the classic *continuous-time Markov chain* model [17]. Let $\lambda_{ij} \geq 0$ be user transition rate from i to j, and let $\Lambda_i = \sum_{j\in\mathcal{N}_i} \lambda_{ij}$, the user association time at i denoted by T_i^{hold} follows an exponential distribution

$$f_{T_i^{\text{hold}}}(T) = \Lambda_i e^{-\Lambda_i T}, \quad T \geq 0. \tag{13}$$

After association ends, the probability of transition to j is:

$$q_{ij} = \lambda_{ij}/\Lambda_i. \tag{14}$$

Therefore, the tunneling probability during $D_{i,k,m}^{\text{o}}$ is

$$p_{ij}^{k,m} = q_{ij} \mathbb{P}\{D_{i,k,m}^{\text{o}} > T_i^{\text{hold}}\} = q_{ij} \left(1 - e^{-\Lambda_i D_{i,k,m}^{\text{o}}} \right) \tag{15}$$

and recall that tunneling flow on (i,j) is incurred by i forwarding result packets of users moved from i to j, we have

$$F_{ij}^{\text{tun}} = \sum\nolimits_{(k,m)\in\mathcal{S}} L_{k,m}^{\text{res}} r_i^k s_i^{k,m} p_{ij}^{k,m}. \tag{16}$$

Figure 2(b) illustrates that tunneling can significantly increase overall service latency as user mobility intensifies.

Quality-latency tradeoff. Let η be the system's quality-latency tradeoff preference, we maximize the average request utility minus latency over service selection and routing:

$$\max_{\boldsymbol{s},\boldsymbol{\phi}} Q = \frac{\sum_{i\in\mathcal{V}} \sum_{k\in\mathcal{K}} r_i^k \sum_{m\in\mathcal{M}_k} s_i^{k,m} \left(\eta u_{k,m} - D_{i,k,m} \right)}{\sum_{i\in\mathcal{V}} \sum_{k\in\mathcal{K}} r_i^k}$$
$$\text{s.t. } \boldsymbol{s} \geq \boldsymbol{0},\ \boldsymbol{\phi} \geq \boldsymbol{0},\ \text{ and (3),(5) hold} \tag{P0}$$

Problem (P0) captures the tradeoff between AI service quality and experienced latency across all requests. It is difficult as $D_{i,k,m}$ is highly coupled through links along the paths. We reformulate (P0) into a more tractable form.

$$\min_{\boldsymbol{s},\boldsymbol{\phi}} J = \sum_{(i,j)\in\mathcal{E}} D_{ij} + \sum_{i\in\mathcal{V}\cup\mathcal{U}} C_i - \sum_{i\in\mathcal{V}} \sum_{(k,m)\in\mathcal{S}} \hat{u}_{k,m} r_i^k s_i^{k,m}$$
$$\text{s.t. } \boldsymbol{s} \geq \boldsymbol{0},\ \boldsymbol{\phi} \geq \boldsymbol{0},\ \text{ and (3),(5) hold} \tag{P1}$$

where $D_{ij} = F_{ij} d_{ij}$, $C_i = G_i c_i$, and $\hat{u}_{k,m} = \eta u_{k,m} - d_{AP} \mathbb{1}_{m\neq 0}$ is the modified utility.

Proposition 1. *With fixed* $\{r_i^k\}$, (P1) *is equivalent to* (P0). *Specifically, it holds* $J = -\left(\sum_i \sum_k r_i^k\right) Q$ *for any* $(\boldsymbol{s}, \boldsymbol{\phi})$.

We remark that (P1) is non-convex in $(\boldsymbol{s}, \boldsymbol{\phi})$. While (P1) structurally resembles previous models (e.g., [18]), the inclusion of tunneling introduces significant complexity. Theorem 1 gives a set of KKT necessary optimality conditions for (P1). It aligns naturally with decentralized decision-making, and nodes need only act "myopically" based on the marginal costs. e.g., for service selection, user at i should assign new requests to the minimum-marginal model $\partial J/\partial s_i^{k,m}$; for routing, node i should forward marginal incoming requests of (k,m) to the neighbor j minimizing $\partial J/\partial \phi_{ij}^{k,m}$.

[3] For local models $m = 0$, we assume $D^{i,k,0} = c_u$ with a constant c_u.

Theorem 1. *Suppose* $(\boldsymbol{\phi}, \boldsymbol{s})$ *optimally solves* (P1)*, then*

$$\frac{\partial J}{\partial s_i^{k,m}} \begin{cases} = \min_{n\in\mathcal{M}_k} \frac{\partial J}{\partial s_i^{k,n}}, & \text{if } s_i^{k,m} > 0, \\ \geq \min_{n\in\mathcal{M}_k} \frac{\partial J}{\partial s_i^{k,n}}, & \text{if } s_i^{k,m} = 0, \end{cases} \tag{17a}$$

$$\frac{\partial J}{\partial \phi_{ij}^{k,m}} \begin{cases} = \min_{l\in\mathcal{N}_i} \frac{\partial J}{\partial \phi_{il}^{k,m}}, & \text{if } \phi_{ij}^{k,m} > 0, \\ \geq \min_{l\in\mathcal{N}_i} \frac{\partial J}{\partial \phi_{il}^{k,m}}, & \text{if } \phi_{ij}^{k,m} = 0. \end{cases} \tag{17b}$$

In general, condition (17) only guarantees the necessity for optimality. Nevertheless, for a simplified system with linear (congestion-independent) costs, it is sufficient for optimality.

Proposition 2. *Suppose* $r_i^k > 0$ *for all* $i \in \mathcal{V}$ *and* $k \in \mathcal{K}$*. If* $d_{ij}(F_{ij}) = d_{ij}$ *with constant* $d_{ij} > 0$ *for all* $(i,j) \in \mathcal{E}$*, and* $c_i(G_i) = c_i$ *with constant* $c_i > 0$ *for all* $i \in \mathcal{V} \cup \mathcal{U}$*, then any* $(\boldsymbol{s}, \boldsymbol{\phi})$ *feasible to* (P1) *and satisfying* (17) *is a global optimizer.*

III. Decentralized Algorithm Design

In this section, we propose a decentralized method to obtain $(\boldsymbol{s}, \boldsymbol{\phi})$ that satisfies KKT condition (17). We first decompose gradients $\partial J/\partial s_i^{k,m}$ and $\partial J/\partial \phi_{ij}^{k,m}$ involved in (17). Let

$$\tilde{t}_i^{k,m} = \sum\nolimits_{j\in\mathcal{N}_i} f_{ji}^{k,m} \tag{18}$$

denote the endogenous arrival rate for (k,m) at i, then $t_i^{k,m} = r_i^k s_i^{k,m} + \tilde{t}_i^{k,m}$. For $m \neq 0$, let $\delta_i^{k,m}$ be the marginal latency caused by increased $\tilde{t}_i^{k,m}$, and $\tau_i^{k,m}$ be the marginal latency caused by tunneling increased exogenous arrival rate $r_i^k s_i^{k,m}$,

$$\delta_i^{k,m} = \partial \left(\sum\nolimits_{(p,q)\in\mathcal{E}} D_{pq} + \sum\nolimits_{p\in\mathcal{V}\cup\mathcal{U}} C_p \right) / \partial \tilde{t}_i^{k,m}, \tag{19}$$

$$\tau_i^{k,m} = L_{k,m}^{\text{res}} \sum\nolimits_{j\in\mathcal{N}_i} D'_{ij}(F_{ij}) p_{ij}^{k,m}. \tag{20}$$

Then the gradients can be decomposed using $\delta_i^{k,m}$ and $\tau_i^{k,m}$.

Theorem 2. *For* $m = 0$ *(local models),*

$$\partial J/\partial s_i^{k,0} = r_i^k (W_{k,m} c_u - \hat{u}_{k,m}). \tag{21a}$$

For $m \neq 0$*,*

$$\partial J/\partial s_i^{k,m} = r_i^k \left(\delta_i^{k,m} + \tau_i^{k,m} - \hat{u}_{k,m} \right), \tag{21b}$$

$$\frac{\partial J}{\partial \phi_{ij}^{k,m}} = t_i^{k,m} \left(L_{k,m}^{req} \frac{\partial J}{\partial F_{ij}^o} + L_{k,m}^{res} \frac{\partial J}{\partial F_{ji}^o} + \delta_j^{k,m} \right). \tag{21c}$$

For $i \in \mathcal{X}_{k,m}$*,*

$$\delta_i^{k,m} = W_{k,m} C_i'(G_i). \tag{22a}$$

For $i \notin \mathcal{X}_{k,m}$*,* $\delta_i^{k,m}$ *is recursively given by*

$$\delta_i^{k,m} = \sum_{j\in\mathcal{N}_i} \phi_{ij}^{k,m} \left(L_{k,m}^{req} \frac{\partial J}{\partial F_{ij}^o} + L_{k,m}^{res} \frac{\partial J}{\partial F_{ji}^o} + \delta_j^{k,m} \right). \tag{22b}$$

We will give $\partial J/\partial F_{ij}^{\text{o}}$ in (26). Theorem 2 is the first to generalize the classic recursive gradient decomposition in [14] to analytically incorporate user mobility. When users are static $\lambda_{ij} = 0$ and request/result sizes simplify to $L_{k,m}^{\text{req}} = 1$, $L_{k,m}^{\text{res}} = 0$, the above recovers the analysis in [14] exactly. Based on Theorem 2, using only local and neighbor information, we design a Decentralized Messaging Protocol (DMP) to estimate $\partial J/\partial s_i^{k,m}$ and $\partial J/\partial \phi_{ij}^{k,m}$. We provide general ideas of DMP and refer readers to [13] for details. Figure 3 illustrates DMP at a relay node i connected with APs j_1, j_2 and server s.

In DMP, two types of control messages, `MSG1` and `MSG2`, are propagated in the network. The intuition of DMP builds on classic recursive messaging schemes [8]. In [8], control messages propagate upstream along request paths, to inform each node of downstream network status and thus adjust local routing decisions. In our case, this idea is retained in `MSG2`, which propagates $\delta_i^{k,m}$ upstream based on (22). However, (22b) depends on $\partial J/\partial F_{ij}^{\text{o}}$ that cannot be obtained locally. We thus use a new pre-stage message `MSG1`, which propagates downstream to compute $\partial J/\partial F_{ij}^{\text{o}}$ before initiating `MSG2`. This two-stage messaging enables fully decentralized gradient estimation even with mobility-induced tunneling.

To estimate gradients, node i obtain d_{ij}, d'_{ij}, D'_{ij}, q_{ij}, λ_{ij}, r_i^k locally, and estimates $D_{i,k,m}^{\text{o}}$ via request RTT. Then, it calculates $\tau_i^{k,m}$ (by (20)), and B_{ij}, $m_i^{k,m}$ defined as:

$$B_{ij} := \Lambda_i q_{ij} d'_{ij} \left(\sum\nolimits_{(k,m)\in\mathcal{S}} r_i^{k,m} \phi_{ij}^{k,m} e^{-\Lambda_i D_{i,k,m}^{\text{o}}} \right), \tag{23}$$

$$m_i^{k,m} := \Lambda_i r_i^{k,m} e^{-\Lambda_i D_{i,k,m}^{\text{o}}} \left(\sum\nolimits_{j\in\mathcal{N}_i} D'_{ij} q_{ij} \right). \tag{24}$$

Messages `MSG1` are propagated downstream to calculate:

$$M_i^{k,m} = \sum\nolimits_{l\in\mathcal{N}_i} \phi_{li}^{k,m} M_l^{k,m} + m_i^{k,m}. \tag{25}$$

After obtaining $M_i^{k,m}$ for all services, node i calculates

$$\frac{\partial J}{\partial F_{ij}^{\text{o}}} = D'_{ij} + \sum\nolimits_{k,m} L_{k,m}^{\text{res}} \phi_{ij}^{k,m} M_i^{k,m} d'_{ij} / (1 - B_{ij}) \tag{26}$$

Theorem 3. *Suppose* $\boldsymbol{\phi}$ *is loop-free, then* $\partial J/\partial F_{ij}^o$ *is given by* (26) *with variable* $M_i^{k,m}$ *recursively defined in* (25)*.*

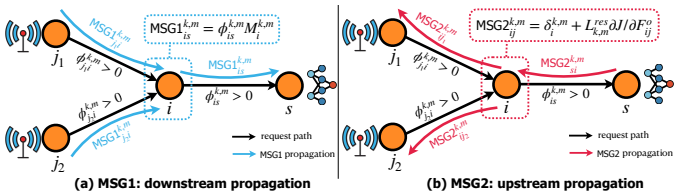


Fig. 3: Illustration of `MSG1` and `MSG2` propagation in DMP.

After obtaining gradients via DMP, each node independently updates its variables $(\boldsymbol{s}, \boldsymbol{\phi})$ through a Frank-Wolfe update similar to [8]. We assume the network starts at $\boldsymbol{s}(0), \boldsymbol{\phi}(0)$, with loop-free $\boldsymbol{\phi}(0)$ and finite $J(0)$. Let $\boldsymbol{s}_i^k = [s_i^{k,m}]_{m\in\mathcal{M}_k}$ and $\boldsymbol{\phi}_i^{k,m} = [\phi_{ij}^{k,m}]_{j\in\mathcal{V}}$, for $n \geq 0$,

$$\boldsymbol{s}_i^k(n+1) = \boldsymbol{s}_i^k(n) + \alpha(n) \left(\boldsymbol{d}_{i,k}^s(n) - \boldsymbol{s}_i^k(n) \right), \tag{27a}$$

$$\boldsymbol{\phi}_i^{k,m}(n+1) = \boldsymbol{\phi}_i^{k,m}(n) + \alpha(n) \left(\boldsymbol{d}_{i,k,m}^{\phi}(n) - \boldsymbol{\phi}_i^{k,m}(n) \right), \tag{27b}$$

the update directions $\boldsymbol{d}_{i,k}^s(n)$ and $\boldsymbol{d}_{i,k,m}^{\phi}(n)$ are given by:

$$\begin{aligned} \boldsymbol{d}_{i,k}^s(n) &= \arg\min\nolimits_{\boldsymbol{d}\in\mathcal{D}_{i,k}^s} \left\langle \nabla_{i,k}^s J(n), \boldsymbol{d} \right\rangle, \\ \boldsymbol{d}_{i,k,m}^{\phi}(n) &= \arg\min\nolimits_{\boldsymbol{d}\in\mathcal{D}_{i,k,m}^{\phi}} \left\langle \nabla_{i,k,m}^{\phi} J(n), \boldsymbol{d} \right\rangle, \end{aligned} \tag{28}$$

$\alpha(n)$ is the step sizes; $\nabla_{i,k}^s J = [\partial J/\partial s_i^{k,m}]_{m\in\mathcal{M}_k}$ and $\nabla_{i,k,m}^{\phi} J = [\partial J/\partial \phi_{ij}^{k,m}]_{j\in\mathcal{V}}$ are gradients; $\mathcal{D}_{i,k}^s$ and $\mathcal{D}_{i,k,m}^{\phi}$

are feasible sets, defined by

$$\mathcal{D}^s_{i,k} = \left\{ \boldsymbol{s}^k_i \geq \mathbf{0} : (3) \text{ holds} \right\},$$

$$\mathcal{D}^\phi_{i,k,m} = \left\{ \boldsymbol{\phi}^{k,m}_i \geq \mathbf{0} : (5) \text{ holds; } \phi^{k,m}_{ij} = 0, \forall j \in \mathcal{B}^{k,m}_i \right\}.$$

Set $\mathcal{B}^{k,m}_i \subseteq \mathcal{V}$ is the *blocked node set* invented by [14] that guarantees $\boldsymbol{\phi}(n)$ are loop-free throughout the algorithm. Since $\mathcal{D}^s_{i,k}$ and $\mathcal{D}^\phi_{i,k,m}$ are standard simplices, linear programming (28) admits closed-form solutions $\boldsymbol{d}^s_{i,k}(n) = \boldsymbol{e}_{m^*_{i,k}(n)}$ and $\boldsymbol{d}^\phi_{i,k,m}(n) = \boldsymbol{e}_{j^*_{i,k,m}(n)}$, where $\boldsymbol{e}_k$ is the standard basis vector with the k-th entry being 1 and all others being 0, and

$$m^*_{i,k}(n) = \arg\min_{m' \in \mathcal{M}_k} \partial J / \partial s^{k,m'}_i(n), \tag{29a}$$

$$j^*_{i,k,m}(n) = \arg\min_{j' \in \mathcal{Q}^{k,m}_i} \partial J / \partial \phi^{k,m}_{ij'}(n). \tag{29b}$$

This aligns with our aforementioned intuitions of (17), i.e., choosing the service/forwarding decisions with minimum marginal costs. Algorithm 1 summarizes our local update.

Algorithm 1: Local Frank-Wolfe update (LFW)

Output: Variables $\boldsymbol{s}(n)$, $\boldsymbol{\phi}(n)$ for $n \geq 1$.
Determine sets $\mathcal{B}^{k,m}_i$ by network routing protocol.
At end of n-th slot, node i do
  Obtain $\partial J/\partial \boldsymbol{s}(n)$ and $\partial J/\partial \boldsymbol{\phi}(n)$ by DMP.
  Determine indices $m^*_{i,k}$ and $j^*_{i,k,m}$ by (29).
  Update $\boldsymbol{s}^k_i(n+1)$ and $\boldsymbol{\phi}^{k,m}_i(n+1)$ by (27).

Theorem 4. *Suppose ∇J is L-continuous, $\alpha(n)$ satisfies $\sum_{n=1}^\infty \alpha(n) = \infty$ and $\sum_{n=1}^\infty \alpha(n)^2 < \infty$, Algorithm 1 converges to a limit point $(\boldsymbol{s}^*, \boldsymbol{\phi}^*) = \lim_{n\to\infty}(\boldsymbol{s}(n), \boldsymbol{\phi}(n))$, where $(\boldsymbol{s}^*, \boldsymbol{\phi}^*)$ satisfies condition* (17).

IV. Optimized AI Service Placement

Extended system model. We extend our framework to jointly optimize AI service placement (i.e., determining sets $\mathcal{X}_{k,m}$) via binary variable $x^{k,m}_i$, namely, $x^{k,m}_i = 1$ if i hosts model m for task k, and 0 otherwise.Let $\boldsymbol{x} = [x^{k,m}_i]$ denote the *global service placement decision*, with the following hold

$$\sum\nolimits_{(k,m)\in\mathcal{S}} L^{\text{mod}}_{k,m}\, x^{k,m}_i \leq R_i, \quad \forall\, i \in \mathcal{V}, \tag{30}$$

where $L^{\text{mod}}_{k,m}$ is the resource occupancy of (k,m) and R_i is the capacity of node i. Then (5) becomes

$$\sum\nolimits_{j\in\mathcal{N}_i} \phi^{k,m}_{ij} = 1 - x^{k,m}_i, \tag{31}$$

implying node i either hosts service (k,m) or forwards all arriving (k,m) requests to its neighbors. To address the combinatorial difficulty, we adopt a relaxation approach similar to [8]. We treat $x^{k,m}_i$ as independent Bernoulli random variables with $y^{k,m}_i = \mathbb{E}[x^{k,m}_i] \in [0,1]$ being the probability that i hosts (k,m). We thus optimize over vector $\boldsymbol{y} = [y^{k,m}_i]$, with

$$\sum\nolimits_{(k,m)} L^{\text{mod}}_{k,m} y^{k,m}_i \leq R_i, \quad y^{k,m}_i + \sum\nolimits_{j\in\mathcal{N}_i} \phi^{k,m}_{ij} = 1. \tag{32}$$

The expected computation workload at node i is now

$$G_i = \sum\nolimits_{(k,m)\in\mathcal{S}} W_{k,m}\, y^{k,m}_i\, t^{k,m}_i, \quad \forall i \in \mathcal{V}. \tag{33}$$

With these extensions, the joint optimization problem for service placement, selection, and routing is cast as

$$\min_{\boldsymbol{s},\boldsymbol{\phi},\boldsymbol{y}} \quad J = \sum_{i,j} D_{ij} + \sum_i C_i - \sum_i \sum_{k,m} \hat{u}_{k,m} r^k_i s^{k,m}_i$$
$$\text{s.t.} \quad \boldsymbol{s} \geq \mathbf{0}, \boldsymbol{\phi} \geq \mathbf{0}, \mathbf{1} \geq \boldsymbol{y} \geq \mathbf{0}, (3),(5),(32) \text{ hold} \tag{P2}$$

Theorem 5 (KKT with service placement)**.** *Suppose $(\boldsymbol{s}, \boldsymbol{\phi}, \boldsymbol{y})$ optimally solves* (P2)*, then* (17) *holds. It also holds that*

$$\xi^{k,m}_i \begin{cases} \geq \min_{(k',m')\in\mathcal{S}: y^{k',m'}_i > 0} \xi^{k',m'}_i, & \text{if } y^{k,m}_i = 1, \\ \leq \min_{(k',m')\in\mathcal{S}: y^{k',m'}_i > 0} \xi^{k',m'}_i, & \text{if } y^{k,m}_i = 0, \\ = \min_{(k',m')\in\mathcal{S}: y^{k',m'}_i > 0} \xi^{k',m'}_i, & \text{o.w.} \end{cases} \tag{34}$$

where $\xi^{k,m}_i = \left(\min_{j\in\mathcal{N}_i} \partial J/\partial \phi^{k,m}_{ij}\right) / L^{mod}_{k,m}$.

Condition (34) indicates that each node i prioritizes hosting services based on the marginal latency reduction per unit of hosting resource, captured by $\xi^{k,m}_i$. Moreover, Theorem 2 applies with the recursive decomposition (22) generalized to

$$\delta^{k,m}_i = y^{k,m}_i W_{k,m} C'_i + \sum_{j\in\mathcal{N}_i} \phi^{k,m}_{ij} \left(L^{\text{req}}_{k,m} \frac{\partial J}{\partial F^{\text{o}}_{ij}} + L^{\text{res}}_{k,m} \frac{\partial J}{\partial F^{\text{o}}_{ji}} + \delta^{k,m}_j \right).$$

We omit the decentralized algorithm for the optimized AI service placement case as it is almost a replica of Sec III. With $\boldsymbol{y}$ involved, the simplification (29b) no longer holds. We present in [13] a valid simplification for Frank-Wolfe update.

V. Numerical Evaluation

We conduct a flow-level numerical evaluation of the proposed algorithm and baselines on both synthetic and real-world network topologies: **`grid`**: A 5×5 grid network. **`MEC`**: A 3-level 3-ary tree with same-parent nodes linearly connected, representing a typical hierarchical MEC architecture. **`ER`**: A connectivity-guaranteed Erdős-Rényi graph with edge probability $p = 0.15$. **`D-Tel`**: The backbone topology of Deutsche Telekom. **`SW`**: A Watts-Strogatz small-world network.

We summarize key parameters of these scenarios in Table I. Moreover, we assume $r^k_i = 1$, $L^{\text{req}}_{k,m} = 0.25$, $L^{\text{res}}_{k,m} = 0.75$, and $L^{\text{mod}}_{k,m}$ follow the sequence $[10, 20, 30, \ldots]$, with corresponding utilities $u_{k,m} = [0.1, 0.3, 0.5, \ldots]$. Delays on links and nodes are approximated via a third-order Taylor expansion, with μ_{ij} and ν_i as service rates. Mobility transition probabilities $q_{ij} \in [0,1]$ are u.a.r. with $\sum_j q_{ij} = 1$. We set the latency–utility tradeoff parameter to $\eta = 1$.

Beyond the proposed decentralized protocol **`DMP-LFW-P`**, we implement the following baselines: **`LFW-Greedy`**: Uses `DMP` and `LFW`, with each node greedily serving the most popular services (based on $t^{k,m}_i$) until capacity is filled. **`Static-LFW`**: A static variant of [8], approximating gradients as $\partial J/\partial F^{\text{o}}_{ij} = D'_{ij}$ without propagating `MSG1`, thus ignoring tunneling. **`SM`**: Models latency under service migration, assuming entire models are transferred between same-layer nodes upon user transitions. **`LPR`** [19]: Solves a linear program for model selection and routing under greedy placement, using marginal delays $d_{ij}|_{F_{ij}=0}$ and $c_i|_{G_i=0}$. **`MaxTP`**: A flow-level approximation of backpressure-based scheduling that minimizes the maximum local queue size.

Name	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{K}\rvert$	$\lvert\mathcal{S}\rvert$	μ_{ij}	ν_i	Λ_i	R_i
grid	9	5	15	10	10	0.10	20
MEC	13	5	20	10	10	0.10	20
ER	30	20	40	15	15	0.15	30
D-Tel	68	30	100	15	15	0.15	30
SW	120	45	150	20	20	0.15	30

TABLE I: Scenarios

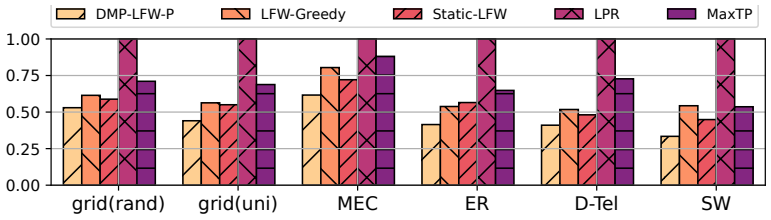


Fig. 4: Normalized objective J in all scenarios

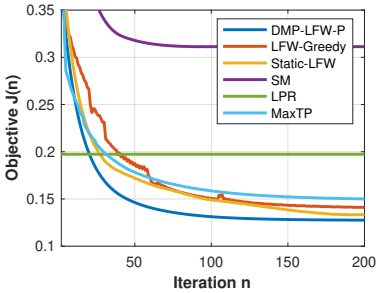


Fig. 5: Convergence trajectory

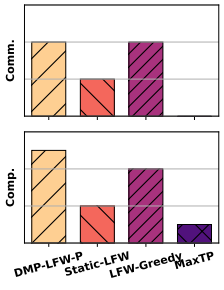


Fig. 6: Overhead

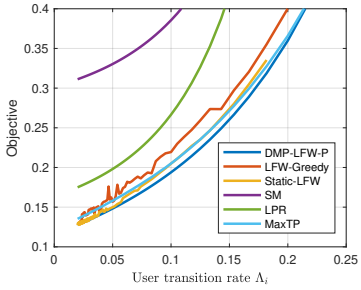


Fig. 7: J v.s. Λ_i

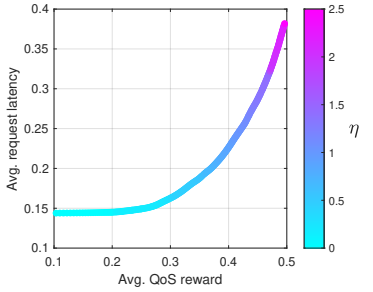


Fig. 8: QoS-latency tradeoff

We set $\alpha = 0.05$ and construct $\mathcal{B}_i^{k,m}$ to resemble a service-specific DAG with maximal edge coverage. Figure 4 shows the normalized convergent J across scenarios excluding SM. For grid, we compare grid(rand) (random q_{ij}) and grid(uni) (uniform q_{ij}). DMP-LFW-P consistently outperforms all baselines, achieving up to 17% improvement over the second best. Gains are more prominent under directional user movement. LFW-Greedy and Static-LFW ablate joint service placement and tunneling awareness. LPR performs the worst by ignoring congestion, while MaxTP is second worst due to not directly optimizing latency. Notably, our method yields increasing benefits as network scale grows.

We further investigate grid in detail. Figure 5 illustrates convergence trajectories. Figure 6 compares per-node communication and computation overheads. All distributed methods exhibit per-node complexity $O(|\mathcal{S}||\mathcal{N}_i|)$; we evaluate computational load via its coefficient and communication via average control messages exchanged. Figure 7 shows objective J versus user transition rate Λ_i. As mobility increases (e.g., $\Lambda_i \geq 0.1$), congestion sharply rises, degrading performance across all methods. In this high-mobility regime, MaxTP approaches the performance of DMP-LFW-P.

Figure 8 illustrates the tradeoff between QoS and latency under different preference η. Each point on the curve reflects the converged state of DMP-LFW-P. Higher η leads to increased average QoS, but also to superlinearly growing latency, indicating an increasing marginal delay for each QoS gain.

REFERENCES

[1] F. T. Council, "Ai meets decentralization: How blockchain is democratizing ai," *Forbes*, 2024, https://www.forbes.com/sites/digital-assets/2024/11/11/ai-meets-decentralization-how-blockchain-is-democratizing-ai/.

[2] A. M. Hadjkouider, C. A. Kerrache, A. Korichi, Y. Sahraoui, C. T. Calafate, S. Dhelim, and A. Adnane, "A review of service selection strategies in mobile iot networks," *IEEE Open Journal of the Communications Society*, 2024.

[3] L. Huang, Y. Wu, J. M. Parra-Ullauri, R. Nejabati, and D. Simeonidou, "Ai model placement for 6g networks under epistemic uncertainty estimation," in *ICC 2024*, 2024.

[4] N. Hudson, H. Khamfroush, M. Baughman, D. E. Lucani, K. Chard, and I. Foster, "Qos-aware edge ai placement and scheduling with multiple implementations in faas-based edge computing," *Future Generation Computer Systems*, vol. 157, pp. 250–263, 2024.

[5] H. Wu, S. Deng, W. Li, J. Yin, X. Li, Z. Feng, and A. Y. Zomaya, "Mobility-aware service selection in mobile edge computing systems," in *2019 IEEE international conference on web services (ICWS)*, 2019.

[6] C. Li, Q. Zhang, C. Huang, and Y. Luo, "Optimal service selection and placement based on popularity and server load in multi-access edge computing," *Journal of Network and Systems Management*, 2023.

[7] Y. Cai, J. Llorca, A. M. Tulino, and A. F. Molisch, "Joint compute-caching-communication control for online data-intensive service delivery," *IEEE Transactions on Mobile Computing*, 2023.

[8] J. Zhang and E. Yeh, "Congestion-aware routing and content placement in elastic cache networks," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 1471–1480.

[9] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Transactions on Networking*, 2019.

[10] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multi-cell mobile edge computing: Joint service migration and resource allocation," *IEEE Transactions on Wireless Communications*, 2021.

[11] J. Tu, L. Yang, and J. Cao, "Distributed machine learning in edge computing: Challenges, solutions and future directions," *ACM Computing Surveys*, 2025.

[12] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[13] J. Zhang, S. Vlaski, and K. K. Leung, "Decentralized ai service placement, selection and routing in mobile networks," https://drive.google.com/file/d/1uQtHiZ_AtxZeLdDJU_JeKg_4pgk7tnPo/view?usp=drive_link, 2025.

[14] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE transactions on communications*, 1977.

[15] H. Li and L. Duan, "Theory of mixture-of-experts for mobile edge computing," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[16] D. Bertsekas and R. Gallager, *Data networks*. Athena Scientific, 2021.

[17] B. O. Jalel, V. Véronique *et al.*, "Continuous time markov chain traffic model for urban environments," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[18] J. Zhang and E. Yeh, "Loam: Low-latency communication, caching, and computation placement in data-intensive computing networks," in *23rd International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 2025.

[19] B. Liu, Y. Cao, Y. Zhang, and T. Jiang, "A distributed framework for task offloading in edge computing networks of arbitrary topology," *IEEE Transactions on Wireless Communications*, 2020.