Optimal Congestion-aware Routing and Offloading in Collaborative Edge Computing

Jinkun Zhang, Yuezhou Liu, Edmund Yeh
Department of Electrical and Computer Engineering, Northeastern University, Boston, US
zhang.jinku@northeastern.edu, liu.yuez@northeastern.edu, eyeh@ece.neu.edu

***Abstract*—Collaborative edge computing (CEC) is an emerging paradigm where heterogeneous edge devices collaborate to fulfill computation tasks, such as model training or video processing, by sharing communication and computation resources. Nevertheless, when considering network congestion, the optimal data/result routing and computation offloading strategy of CEC still remains an open problem. In this paper, we formulate a flow model of partial-offloading and multi-hop routing in CEC network with arbitrarily topology and heterogeneous communication/computation capability. In contrast to most existing works, our model applies to tasks with non-negligible result size, and allows data sources to be distinct from the result destination. We propose a network-wide cost minimization problem with congestion-aware convex cost functions. Such convex cost covers various performance metrics and constraints, such as average queueing delay with limited processor capacity. Although the problem is non-convex, we provide necessary conditions and sufficient conditions for the global-optimal solution, and devise a fully distributed algorithm that converges to the optimum in polynomial time. Our proposed method allows asynchronous individual updating, and is adaptive to changes of network parameters. Numerical evaluation shows that our method significantly outperforms other baseline algorithms in multiple network instances, especially in congested scenarios.**

I. Introduction

Recent years have seen an explosion in the number of mobile and IoT devices. Many of the emerging mobile applications, such as VR/AR, autonomous driving, are computation-intensive and time-critical. Mobile devices running these applications generate a huge amount of data traffic which is predicted to reach 288EB per month in 2027 [1]. It is becoming impractical to direct all computation requests and their data to the central cloud due to limited backhaul bandwidth and high associated latency. Edge computing has been proposed as a promising solution to provide computation resources and cloud-like services in close proximity to the mobile devices.

In edge computing, requesters offload their computation to the edge servers, where the network topology is typically hierarchical. Extending the idea of edge computing is a new concept called collaborative edge computing (CEC), in which the network structure is more flexible. In addition to point-to-point offloading, CEC permits multiple stakeholders (mobile devices, IoT devices, edge servers, or cloud) to collaborate with each other by sharing data, communication resources, and computation resources to accomplish computation tasks [2]. CEC improves the utilization efficiency of resources so that computation-intensive and time-critical services can be better completed at the edge. Mobile devices equipped with computation capabilities can collaborate with each other through D2D communication [3]. Edge servers can also collaborate with each other for load balancing or further with the central cloud to offload demands that they cannot accommodate [4]. Furthermore, CEC is needed when there is no direct connection between devices and edge servers. Consider unmanned aerial vehicle (UAV) swarms or autonomous cars in rural areas, computation-intensive tasks of UAVs or cars far away from the wireless access point should be collaboratively computed or offloaded through multi-hop routing to the server with the help of other devices [3], [5].

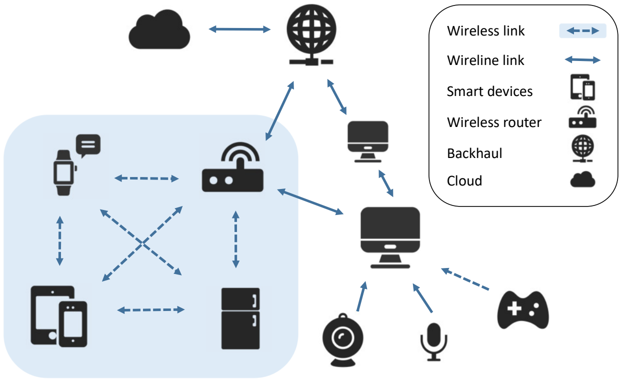


Fig. 1: Sample system topology involving IoT network on the edge

We wish to study a general framework of CEC which enables various types of collaboration among stakeholders. In particular, we consider a multi-hop network with arbitrary topology, where the nodes collaboratively finish multiple computation tasks. Nodes have heterogeneous computation capabilities and some are also data sources (sensors and mobile users) that generate data for computation tasks. Each task has a requester node for the computation result. We allow partial offloading introduced in [6], i.e., a task could be partitioned into multiple components and separately offloaded. For example in video compression, the original video can be chunked into blocks and compressed separately at multiple devices, and the results could then be merged.

Finishing a task requires the routing of data from possibly multiple data sources to multiple nodes for computation and the routing of results to the destination (task requester). We aim for a joint routing (how to route the data/result) and computation offloading (where to compute) strategy that minimizes the total communication and computation costs.

ISBN 978-3-903176-49-2 © 2022 IFIP

Joint routing and computation offloading has been investigated in a number of prior contributions. Sahni et al. [2] [7] adopt the model where each task is only performed once with the exact release time known, which we refer as *single-instance model*. Zhang et al. [8] consider a *flow model* where data collection and computation of each task are performed continuously, and time-averaged costs are measured based on the rates of data and result flows. Most existing studies in CEC assume the data of a tasks is provided by the requester itself [5] [9] [10]. Although Sahni et al. [7] [3] consider arbitrary data sources, the network is assumed to be fully connected, or with predefined routing paths. The communication cost in previous studies is often assumed to be solely due to input data transmission [11] [2], and the results are transmitted simply along the reverse path of input data [12] [13]. However, the size of computation results is not negligible in many practical applications, e.g., federated or distributed machine learning [14], file decompression, and image enhancement. For communication, Hong et al. [5], [15] formulate heterogeneous link transmission speeds, and Sahni et al. [7] consider link bandwidth constraints. However, most works assume the link costs be linear functions of the traffic. Xiang et al. [16] study routing and computation offloading jointly with network slicing, and propose a heuristic algorithm. They consider a flow model, with non-linear delay functions, but without the consideration of computation results.

Distinct from the above studies, in this paper, our formulation simultaneously 1) adopts the flow model on CEC network with arbitrary multi-hop topology, 2) allows the requester node to be distinct from data sources, 3) optimizes routing for both data and results of non-negligible size, and 4) models network congestion by considering non-linear communication and computation costs. Our detailed contributions are:

- To the best of our knowledge, we are the first to jointly formulate partial offloading and routing for both data and results in arbitrary network with congestible links.
- We provide the global optimal routing and offloading strategy for a non-convex total cost minimization problem, by studying a set of sufficient optimality conditions.
- We devise a fully distributed algorithm that converges to the optimum with asynchronous implementation, and is adaptive to changes of network parameters.
- Through extensive experimentation, we show the advantages of the proposed algorithm over baselines in different network instances, especially in congested scenarios.

II. System Model and Problem Formulation

We begin by presenting our formal model of a collaborative edge computing network where multiple stakeholders collaborate to carry out computation tasks. Such networks are motivated by real-word applications such as IoT networks, connected vehicles and UAV swarms. An example that involves an IoT network at the edge is shown in Fig. 1. We summarize in Table I the notation used in this paper.

$G=(V,E)$	Network graph G, set of nodes V and links E
M	Number of computation types
(d,m)	Task with destination d and computation type m
S	Set of all tasks
$r_i(d,m)$	Exogenous input data rate for task (d,m) at node i
$\mathcal{O}(i)$, $\mathcal{I}(i)$	Out-neighbors and in-neighbors of node i
$f_{ij}^-(d,m)$	Data flow rate of task (d,m) on link (i,j)
$f_{ij}^+(d,m)$	Result flow of rate task (d,m) on link (i,j)
$g_i(d,m)$	Computational input rate of (d,m) at i
a_m	Ratio of result size over input data of computation m
$t_i^-(d,m)$	Total data flow rate of (d,m) at i
$t_i^+(d,m)$	Total result flow rate of (d,m) at i
$\phi_{ij}^-(d,m)$	Fraction of $t_i^-(d,m)$ forwarded to node j $(j \neq 0)$
$\phi_{i0}^-(d,m)$	Fraction of $t_i^-(d,m)$ assigned to computation at i
$\phi_{ij}^+(d,m)$	Fraction of $t_i^+(d,m)$ forwarded to node j
g_i^m	Sum of computational input rates of type m at i
w_{im}	Weight for computation type m at i
F_{ij}	Total flow on link (i,j)
G_i	Computation workload at i
$D_{ij}(F_{ij})$	Communication cost (e.g. queueing delay) on (i,j)
$C_i(G_i)$	Computation cost (e.g. CPU runtime) at node i
T	Sum of all communication and computation costs

TABLE I: Major notations

Network and tasks. We model the network with a directed and strongly connected graph $G = (V, E)$ where V is the set of nodes and E is the set of links. Computations are performed by the nodes, mapping input data to results of non-negligible size, e.g., image/video compression, message encoding/decoding and model training. The input data and results for computation are transmitted through the links. Nodes and links are assumed to have heterogeneous computation and communication capabilities, respectively. We assume that $M \in \mathbb{N}^+$ different types of computations are performed.

Communication and computation are task-driven, where a task involves 1) routing input data from potentially multiple data sources to computation sites, 2) computing, and 3) delivering the results to a given destination. For example, in an IoT monitor application, data sources could be sensors on different smart devices and the destination could be a user's cellphone. The data collected from the sensors is analyzed and processed in the network before being delivered to the user.

We denote a task by a pair (d, m), where $d \in V$ is the destination node and $m \in [M]$ is the specified computation type, where $[M] = \{1, 2, \dots, M\}$. We denote by S be the set of all tasks. To incorporate partial offloading [6], we assume the exogenous input data is chunked into *data blocks* of equal size. Computation could be independently performed on each data block, and a *result block* is generated accordingly. For tasks of computation type m, we assume the result blocks are also of equal size, which is $a_m \in \mathbb{R}^+$ times the data block size. Such assumption is adopted in many partial offloading studies that consider result size, e.g. [12], where typically $a_m \leq 1$. We also allow $a_m > 1$ for special computation types with the result size larger than input data, e.g., video rendering, image super-resolution or file decompression.

Data and result flows. In contrast to the single-instance model where each task is performed only once [2], we adopt a flow model similar to [8]. We assume the exogenous input data blocks of each task are continuously injected into the network in the form of flows with certain rates, and the computations are continuously performed. In the network, flows of data blocks, i.e. *data flows*, are routed as computational input to nodes with computation resources. After computation, flows of result blocks, i.e., *result flows*, are generated and routed to corresponding destinations.

We assume the exogenous input data blocks of task (d, m) are injected into the network with rate $r_i(d, m) \geq 0$ (bit/s) at node i. [1] Let $f^-_{ij}(d, m) \geq 0$ denote the data flow rate (bit/s) on link (i, j) for task (d, m). Let $g_i(d, m) \geq 0$ be the data flow rate forwarded to node i's computation unit for task (d, m), referred as the *computational input rate*. As the size of a result block is a_m times the size of a data block, the result flow rate generated from computation at node i for task (d, m) is $a_m g_i(d, m)$ (bit/s). Let $f^+_{ij}(d, m) \geq 0$ be the result flow rate on link (i, j) for task (d, m). We further let $t^-_i(d, m)$ and $t^+_i(d, m)$ be the total data flow rate and total result flow rate for task (d, m) at node i, respectively, given as

$$t^-_i(d, m) = \sum\nolimits_{j \in \mathcal{I}(i)} f^-_{ji}(d, m) + r_i(d, m),$$
$$t^+_i(d, m) = \sum\nolimits_{j \in \mathcal{I}(i)} f^+_{ji}(d, m) + a_m g_i(d, m),$$

where $\mathcal{I}(i) = \{j \big| (j, i) \in E\}$. Similarly, we define $\mathcal{O}(i) = \{j \big| (i, j) \in E\}$. We demonstrate the flow model in Fig.2.

Routing and offloading strategy. The network performs hop-by-hop multi-path routing. For routing of data and result flows, let $\phi^-_{ij}(d, m)$ and $\phi^+_{ij}(d, m) \in [0, 1]$ be the fraction of data and result flows forwarded to node j by node i for task (d, m), respectively. For computation offloading, let $\phi^-_{i0}(d, m) \in [0, 1]$ be the fraction of data flow for task (d, m) forwarded to local computation unit of i. Thus, for all $(d, m) \in S$ and $i \in V$,

$$\begin{aligned} f^-_{ij}(d, m) &= t^-_i(d, m)\phi^-_{ij}(d, m), \quad \forall j \in V \\ g_i(d, m) &= t^-_i(d, m)\phi^-_{i0}(d, m), \\ f^+_{ij}(d, m) &= t^+_i(d, m)\phi^+_{ij}(d, m). \quad \forall j \in V \end{aligned}$$

Note that $\phi^-_{ij}(d, m) = \phi^+_{ij}(d, m) \equiv 0$ if $(i, j) \notin E$. We denote by vector $\boldsymbol{\phi}$ the global routing and offloading strategy.

To ensure all tasks are fulfilled, every data block must be eventually forwarded to some computation unit, and every result block must be delivered to the corresponding destination. Specifically, the data flows are either forwarded to nearby nodes or to local computation unit, and the result flows exit the network at the destination. Therefore, for all $(d, m) \in S$ and $i \in V$,

$$\sum_{j \in \{0\} \cup V} \phi^-_{ij}(d, m) = 1, \quad \sum_{j \in V} \phi^+_{ij}(d, m) = \begin{cases} 1, \text{ if } i \neq d, \\ 0, \text{ if } i = d. \end{cases} \tag{1}$$

Communication cost. We assume the communication cost on link (i, j) is $D_{ij}(F_{ij})$, where F_{ij} is the total flow rate on (i, j),

$$F_{ij} = \sum\nolimits_{(d,m) \in S} \left(f^+_{ij}(d, m) + f^-_{ij}(d, m) \right),$$

and $D_{ij}(\cdot)$ is an increasing, continuously differentiable and convex function. Such convex costs subsume a variety of existing cost functions including commonly adopted linear cost [17]. It also incorporates performance metrics that reflect the network congestion status. For example, provided that μ_{ij} is the service rate of an M/M/1 queue [18] with $F_{ij} < \mu_{ij}$, $D_{ij}(F_{ij}) = F_{ij}/(\mu_{ij} - F_{ij})$ gives the average number of blocks waiting for or under transmission on link (i, j), and is proportional to the average time for a block from entering the queue to transmission completion. One can also approximate the link capacity constraint $F_{ij} \leq C_{ij}$ (e.g., in [19]) by a smooth convex function that goes to infinity when approaching the capacity limit C_{ij}.

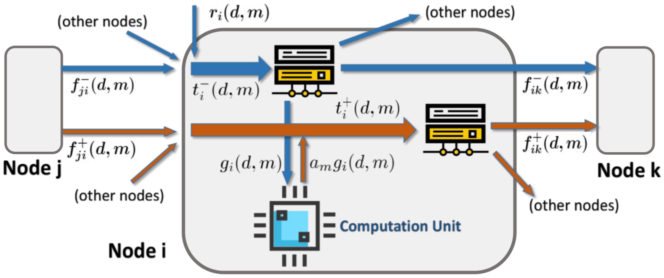


Fig. 2: Example of data and result flows for nodes $j \to i \to k$

Computation cost. Let $g^m_i = \sum_{d:(d,m) \in S} g_i(d, m)$ be the total computational input rate for type m task at node i, and define the computation workload at node i as

$$G_i = \sum\nolimits_{m \in [M]} w_{im} g^m_i,$$

where $w_{im} > 0$ is the weight for type m at node i. We assume the computation cost at node i is $C_i(G_i)$, where $C_i(\cdot)$ is an increasing, continuously differentiable and convex function. For instance, if the computation of type m requires c_m CPU cycles per bit of input data. By setting $w_{im} = c_m$ and $C_i(G_i) = G_i$, we have $C_i(G_i) = \sum_{m \in [M]} c_m g^m_i$, measuring the total CPU cycles. Alternatively, let v^m_i be the computation speed of type m at i and $w_{im} = c_m / v^m_i$, we have $C_i(G_i) = \sum_{m \in [M]} c_m g^m_i / v^m_i$, measuring the CPU runtime. Similar to the communication cost, the convexity assumption of $C_i(\cdot)$ subsumes congestion-dependent computation costs.

Note that for a network with heterogeneous computation resources, our formulation is more flexible than that in [2] [20], where a task always yields the same computation cost wherever it is performed. Our model captures the fact that in practical edge networks, the workload for a certain task may be very different depending on where it is computed, e.g., some parallelizable tasks are easier at nodes employing GPU acceleration, but slower at others.

Problem formulation. We aim at minimizing the total cost of links and devices for both communication and computation,

$$\begin{aligned} \min_{\boldsymbol{\phi}} \quad & T = \sum_{(i,j) \in E} D_{ij}(F_{ij}) + \sum_{i \in V} C_i(G_i) \\ \text{subject to} \quad & \boldsymbol{\phi} \geq \mathbf{0} \text{ and (1) holds.} \end{aligned} \tag{2}$$

Note that problem (2) is not convex in $\boldsymbol{\phi}$, and we do not explicitly impose any link or computation capacity constraints in (2) since they are already incorporated in the cost functions.

[1] Note that we allow multiple nodes i for which $r_i(d, m) > 0$, representing multiple data sources; $r_d(d, m)$ could also be positive, representing computation offloading with locally provided data.

III. Optimality Conditions

In this section, we first establish a set of KKT necessary conditions for (2), and demonstrate by example that such necessary conditions could lead to sub-optimal solutions. Then, we establish a set of sufficient optimality conditions.

Necessary condition. We start by giving closed-form derivatives of T. Our analysis follows [21] and makes non-trivial extensions to data and result flows, as well as in-network computation. For $(i,j) \in E$ and $(d,m) \in S$, the marginal cost for a marginal increase of $\phi_{ij}^-(d,m)$ consists of two components, 1) the marginal communication cost on link (i,j), and 2) the marginal cost of increasing exogenous input rate $r_j(d,m)$. Similarly, the marginal cost of increasing $\phi_{i0}^-(d,m)$ consists of the marginal computation cost at i, and the marginal cost of increasing result traffic $t_i^+(d,m)$. Formally,

$$\frac{\partial T}{\partial \phi_{ij}^-(d,m)} = t_i^-(d,m)\left[D'_{ij}(F_{ij}) + \frac{\partial T}{\partial r_j(d,m)}\right], \text{if } j \neq 0$$
$$\frac{\partial T}{\partial \phi_{i0}^-(d,m)} = t_i^-(d,m)\left[w_{im}C'_i(G_i) + a_m\frac{\partial T}{\partial t_i^+(d,m)}\right], \tag{3}$$

and the marginal cost of increasing $\phi_{ij}^+(d,m)$ is given by

$$\frac{\partial T}{\partial \phi_{ij}^+(d,m)} = t_i^+(d,m)\left[D'_{ij}(F_{ij}) + \frac{\partial T}{\partial t_j^+(d,m)}\right], \tag{4}$$

where the term $\partial T/\partial r_i(d,m)$ is a weighted sum of marginal costs for out-going links and local computation unit, namely,

$$\frac{\partial T}{\partial r_i(d,m)} = \sum_{j\in\mathcal{O}(i)} \phi_{ij}^-(d,m)\left[D'_{ij}(F_{ij}) + \frac{\partial T}{\partial r_j(d,m)}\right]$$
$$+ \phi_{i0}^-(d,m)\left[w_{im}C'_i(G_i) + a_m\frac{\partial T}{\partial t_i^+(d,m)}\right]. \tag{5}$$

Similarly, the term $\partial T/\partial t_i^+(d,m)$ is given by

$$\frac{\partial T}{\partial t_i^+(d,m)} = \sum_{j\in\mathcal{O}(i)} \phi_{ij}^+(d,m)\left[D'_{ij}(F_{ij}) + \frac{\partial T}{\partial t_j^+(d,m)}\right]. \tag{6}$$

One could calculate $\partial T/\partial r_i(d,m)$ and $\partial T/\partial t_i^+(d,m)$ recursively by (5) and (6), since the result flow does not introduce any marginal cost at the destination, i.e., $\partial T/\partial t_d^+(d,m) = 0$.

Therefore, a set of KKT necessary conditions for the optimal solution to (2) is given in Lemma 1.

Lemma 1: Let $\boldsymbol{\phi}$ be the global optimal solution to (2), then for all $i \in V$, $(d,m) \in S$ and $j \in \{0\} \cup \mathcal{O}(i)$,

$$\frac{\partial T}{\partial \phi_{ij}^-(d,m)} \begin{cases} = \min\limits_{k\in\{0\}\cup\mathcal{O}(i)} \frac{\partial T}{\partial \phi_{ik}^-(d,m)}, & \text{if } \phi_{ij}^-(d,m) > 0, \\ \geq \min\limits_{k\in\{0\}\cup\mathcal{O}(i)} \frac{\partial T}{\partial \phi_{ik}^-(d,m)}, & \text{if } \phi_{ij}^-(d,m) = 0, \end{cases}$$

and for all $j \in \mathcal{O}(i)$,

$$\frac{\partial T}{\partial \phi_{ij}^+(d,m)} \begin{cases} = \min\limits_{k\in\mathcal{O}(i)} \frac{\partial T}{\partial \phi_{ik}^+(d,m)}, & \text{if } \phi_{ij}^+(d,m) > 0, \\ \geq \min\limits_{k\in\mathcal{O}(i)} \frac{\partial T}{\partial \phi_{ik}^+(d,m)}, & \text{if } \phi_{ij}^+(d,m) = 0. \end{cases}$$

Proof sketch. The set of KKT conditions is attained by setting the derivatives of Lagrangian function to 0, combined with the complementary slackness. See [22] for the full proof. □

Note that the condition in Lemma 1 is not a sufficient condition for optimality. An example for such non-sufficiency is provided in Fig.3: The only task is $(4,m)$ with exogenous input data occurring only at node 1, the global strategy and marginal costs are shown in the figure. It is easy to verify that conditions in Lemma 1 are satisfied. However, by increasing ϕ_{24}^- and decreasing ϕ_{23}^-, the derivative $\partial T/\partial r_2$ will decrease and thus $\partial T/\partial \phi_{12}^-$ will decrease. In this case, the total cost T could be reduced by increasing ϕ_{12}^- and decreasing ϕ_{14}^-.

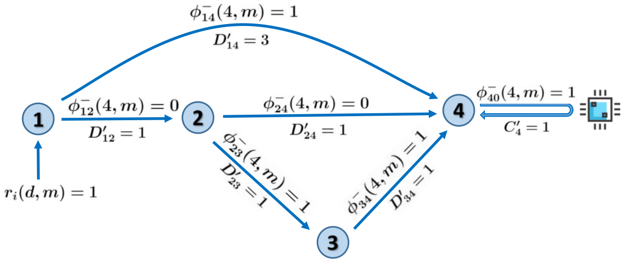


Fig. 3: A non-optimal situation satisfying the condition in Lemma 1

Sufficient condition. The underlying intuition for the non-sufficiency of Lemma 1 is that the condition automatically holds if $t_i^-(d,m) = t_i^+(d,m) = 0$, regardless of the actual routing/offloading strategy at node i. Nevertheless, since $t_i^-(d,m)$ and $t_i^+(d,m)$ exist in (3) and (4) respectively for all j, we remove them and propose a revised set of conditions specified in Theorem 1, which is shown to be sufficient for global optimality.

Theorem 1: Let $\boldsymbol{\phi}$ be feasible for (2). If for all $i \in V$, $(d,m) \in S$ and $j \in \{0\} \cup \mathcal{O}(i)$,

$$\delta_{ij}^-(d,m) \begin{cases} = \min\limits_{k\in\{0\}\cup\mathcal{O}(i)} \delta_{ik}^-(d,m), & \text{if } \phi_{ij}^-(d,m) > 0, \\ \geq \min\limits_{k\in\{0\}\cup\mathcal{O}(i)} \delta_{ik}^-(d,m), & \text{if } \phi_{ij}^-(d,m) = 0, \end{cases}$$

and for all $j \in \mathcal{O}(i)$,

$$\delta_{ij}^+(d,m) \begin{cases} = \min\limits_{k\in\mathcal{O}(i)} \delta_{ik}^+(d,m), & \text{if } \phi_{ij}^+(d,m) > 0, \\ \geq \min\limits_{k\in\mathcal{O}(i)} \delta_{ik}^+(d,m), & \text{if } \phi_{ij}^+(d,m) = 0, \end{cases}$$

then $\boldsymbol{\phi}$ is a global optimal solution to (2), where $\delta_{ij}^-(d,m)$ and $\delta_{ij}^+(d,m)$ are defined as

$$\delta_{ij}^-(d,m) = \begin{cases} D'_{ij}(F_{ij}) + \frac{\partial T}{\partial r_j(d,m)}, & \text{if } j \neq 0, \\ w_{im}C'_i(G_i) + a_m\frac{\partial T}{\partial t_i^+(d,m)}, & \text{if } j = 0, \end{cases}$$
$$\delta_{ij}^+(d,m) = D'_{ij}(F_{ij}) + \frac{\partial T}{\partial t_j^+(d,m)}. \tag{7}$$

Proof sketch. The global optimality follows by a fact that T is jointly convex in $f_{ij}^-(d,m)$, $f_{ij}^+(d,m)$ and $g_i(d,m)$, and a mapping exists from $\boldsymbol{\phi}$ to them. See [22] for the full proof. □

Theorem 1 is the main theoretical result of this paper and leads to a practical algorithm, which is proposed in Section IV. As a simple illustration of the difference between Theorem

1 and Lemma 1, we further assume the network in Fig. 3 has linear communication costs. It turns out that for any routing scheme satisfying Theorem 1, we must have $\phi_{12}^-(4,m)=1$ and $\phi_{24}^-(4,m)=1$, precisely indicating the shortest path $1\to 2\to 4$ for data flow.

IV. Distributed and Adaptive Algorithm

In this section, we introduce a distributed algorithm that converges to the global optimal solution of (2) specified by Theorem 1. The proposed algorithm is a variant of scaled gradient projection. It uses carefully designed scaling matrices to attain better convergence property, and is adaptive to moderate changes of exogenous input rates. It also allows asynchronous variable update for different nodes. Our method is based on [23], and further distinguishes data and result flows by extending the broadcasting protocol.

Algorithm overview. Existence of routing loops generates redundant flow circulation, wastes network resources and causes potential instability. Therefore, we consider strategy $\boldsymbol{\phi}$ with *loop-free* property. For task (d,m), we say there is a *data path* from node i to node j if there is a sequence of nodes $n_1,\cdots,n_L$ for which $(n_l,n_{l+1})\in E$ and $\phi_{n_l n_{l+1}}^-(d,m)>0$ for $l=1,\cdots,L-1$, with $n_1=i$ and $n_L=j$. We say $\boldsymbol{\phi}$ has a *data loop* if there exists some (d,m) and i, j, such that i has a data path to j, and vice versa. The concepts of *result path* and *result loop* are defined similarly. We say strategy $\boldsymbol{\phi}$ is *loop-free* if it has neither a data loop nor a result loop.[2]

We assume the network starts with a feasible and loop-free strategy $\boldsymbol{\phi}^0$, where the initial total cost is finite. Let

$$\boldsymbol{\phi}_i^-(d,m)\equiv\left[\phi_{ij}^-(d,m)\right]_{j\in\{0\}\cup V},\ \boldsymbol{\phi}_i^+(d,m)\equiv\left[\phi_{ij}^+(d,m)\right]_{j\in V},$$

$$\boldsymbol{\delta}_i^-(d,m)\equiv\left[\delta_{ij}^-(d,m)\right]_{j\in\{0\}\cup V},\ \boldsymbol{\delta}_i^+(d,m)\equiv\left[\delta_{ij}^+(d,m)\right]_{j\in V}.$$

At the t-th iteration, each node i updates its routing strategy for data flow of task (d,m), i.e., $\boldsymbol{\phi}_i^-(d,m)$, as follows,

$$\boldsymbol{\phi}_i^-(d,m)^{t+1}= \left[\boldsymbol{\phi}_i^-(d,m)^t-\left(M_i^-(d,m)^t\right)^{-1}\cdot\boldsymbol{\delta}_i^-(d,m)^t\right]^+_{M_i^-(d,m)^t,\mathcal{D}_i^-(d,m)^t}, \tag{8}$$

where $M_i^-(d,m)^t$ is a positive semi-definite diagonal scaling matrix, $\mathcal{D}_i^-(d,m)^t$ is the feasible set of $\boldsymbol{\phi}_i^-(d,m)^{t+1}$, and operator $[\cdot]^+_{A,\mathcal{D}}$ denotes the vector projection scaled by matrix A onto a convex set $\mathcal{D}$, that is,

$$[\boldsymbol{v}_0]^+_{A,\mathcal{D}}=\arg\min_{\boldsymbol{v}\in\mathcal{D}}(\boldsymbol{v}-\boldsymbol{v}_0)^TA(\boldsymbol{v}-\boldsymbol{v}_0).$$

Formula (8) is equivalent to solving the following QP (quadratic programming) [23],

$$\begin{aligned}\boldsymbol{\phi}_i^-(d,m)^{t+1}=\ &\arg\min_{\boldsymbol{v}\in\mathcal{D}_i^-(d,m)^t}\ \boldsymbol{\delta}_i^-(d,m)^t\cdot(\boldsymbol{v}-\boldsymbol{\phi}_i^-(d,m)^t)\\&+(\boldsymbol{v}-\boldsymbol{\phi}_i^-(d,m)^t)^TM_i^-(d,m)^t(\boldsymbol{v}-\boldsymbol{\phi}_i^-(d,m)^t),\end{aligned} \tag{9}$$

where the set $\mathcal{D}_i^-(d,m)^t$ is given by constraints $\boldsymbol{\phi}_i^-(d,m)\geq\mathbf{0}$ and

$$\sum\nolimits_{j\in\{0\}\cup V}\phi_{ij}^-(d,m)=1;\quad \phi_{ij}^-(d,m)=0,\forall j\in\mathcal{B}_i^-(d,m)^t,$$

where $\mathcal{B}_i^-(d,m)^t$ is the set of *blocked nodes* of data flow for (d,m) at i, to guarantee the feasibility and loop-free property. The result strategy $\boldsymbol{\phi}_i^+(d,m)$ is updated in a similar manner as (9) with "-" replaced with "+". Note that $\phi_{dj}^+(d,m)\equiv 0$ for all $j\in V$ as destinations are sinks of result flows.

Algorithm 1: Scaled Gradient Projection (SGP)

```
Start with $t=0$ and a loop-free $\phi^{\mathbf{0}}$ with $T^0<\infty$.
Every node $i$ is informed of $A_{ij}(T^0)$ and $A(T^0)$.
do
    Perform broadcast stage 1) , obtain $\partial T/\partial t_i^+(d,m)$ and $h_i^+(d,m)$ for all $i$ and $(d,m)$.
    Perform broadcast stage 2), obtain $\partial T/\partial r_i(d,m)$ and $h_i^-(d,m)$ for all $i$ and $(d,m)$.
when end of iteration $t$;
do
    Calculate $\delta_{ij}^-(d,m)$ and $\delta_{ij}^+(d,m)$ by (7).
    Calculate $\mathcal{B}_{ij}^-(d,m)$ and $\mathcal{B}_{ij}^+(d,m)$.
    Calculate $M_i^+(d,m)$ and $M_i^-(d,m)$ by (10).
    Solve individual optimization problem (9) and update $\boldsymbol{\phi}_i^-(d,m)$, $\boldsymbol{\phi}_i^+(d,m)$.
at Each node i, at beginning of iteration $t+1$;
```

The proposed method is summarized in Algorithm 1. We emphasize that our method is not purely gradient-based, as the gradients are replaced by $\boldsymbol{\delta}_i^-(d,m)$ and $\boldsymbol{\delta}_i^+(d,m)$ in (8) (see their definition in Theorem 1). We next describe in detail the calculation of $\boldsymbol{\delta}_i^-(d,m)$, $\boldsymbol{\delta}_i^+(d,m)$, scaling matrices and blocked node sets. We then give the asynchronous convergence result and analyze the algorithm complexity.

Marginal cost broadcast. Each node i needs to calculate vectors $\boldsymbol{\delta}_i^-(d,m)$ and $\boldsymbol{\delta}_i^+(d,m)$ following (7), which requires the knowledge of several marginal costs. Suppose the closed-form of $D_{ij}(\cdot)$ and $C_i(\cdot)$ are known, nodes can directly measure $D_{ij}'(F_{ij})$ and $C_i'(G_i)$ while transmitting on link (i,j) and performing local computation. To recursively obtain $\partial T/\partial r_i(d,m)$ and $\partial T/\partial t_i^+(d,m)$ from (5) and (6), respectively, a two-stage distributed broadcast protocol is introduced:

1) Broadcast for $\partial T/\partial t_i^+(d,m)$: Node i first waits until it receives messages carrying $\partial T/\partial t_j^+(d,m)$ from all its downstream neighbor $j\in\mathcal{O}(i)$ for which $\phi_{ij}^+(d,m)>0$. Then, node i calculates its own $\partial T/\partial t_i^+(d,m)$ according to (6) with the measured $D_{ij}'(F_{ij})$ and received $\partial T/\partial t_j^+(d,m)$. Next, node i broadcasts $\partial T/\partial t_i^+(d,m)$ to all its upstream neighbors $k\in\mathcal{I}(i)$ for which $\phi_{ki}^+(d,m)>0$. Note that this stage starts with the destination d, where d broadcasts $\partial T/\partial t_d^+(d,m)=0$ to its upstream neighbors in $\mathcal{I}(d)$.

2) Broadcast for $\partial T/\partial r_i(d,m)$: Similar as in stage 1), the exogenous input marginal $\partial T/\partial r_i(d,m)$ could be calculated from (5) via broadcast. Note that besides all $\partial T/\partial r_j(d,m)$ from downstream neighbors j, node i must also obtain $\partial T/\partial t_i^+(d,m)$ before calculating $\partial T/\partial r_i(d,m)$. Thus the broadcast of $\partial T/\partial r_i(d,m)$ takes place after the broadcast of $\partial T/\partial t_i^+(d,m)$. The broadcast of $\partial T/\partial r_i(d,m)$ starts with the last node of each data path, where at these nodes we have $\partial T/\partial r_i(d,m)=w_{im}C_i'(G_i)+a_m\partial T/\partial t_i^+(d,m)$.

[2] We allow loops concatenated by a data path and a result path of the same task, which occurs in scenarios where the destination is the data source.

With the loop-free property, the broadcast procedure above is guaranteed to traverse throughout the network and terminate within a finite number of steps.

Blocked nodes. To achieve feasibility and the loop-free property, following [21], we let $\mathcal{B}_i^-(d,m)^t$ be the set of nodes to which node i is forbidden to forward any data flow for task (d,m) at iteration t, and let $\mathcal{B}_i^+(d,m)^t$ be the set to which i is forbidden to forward any result flow.

The intuition behind blocked nodes is as follows: Combining Theorem 1 with (5), if $\boldsymbol{\phi}$ is a global optimal solution to (2), $\partial T/\partial t_i^+(d,m)$ should be monotonically decreasing along any result path toward the destination node where $\partial T/\partial t_d^+(d,m) = 0$. We thus require that node i should not forward any result flow to a neighbor j if either 1) $\partial T/\partial t_j^+(d,m) > \partial T/\partial t_i^+(d,m)$, or 2) it could form a result path containing some link (p,q) such that $\partial T/\partial t_q^+(d,m) > \partial T/\partial t_p^+(d,m)$. A similar requirement is applied to $\partial T/\partial r_i(d,m)$ and data paths.

Practically, the information needed to determine blocked node sets could be piggy-backed on the broadcast messages previously described with light overhead. The loop-free property is maintained throughout the algorithm if such a blocking mechanism is implemented in each iteration.

Scaling matrix. The scaling matrices $M_i^-(d,m)^t$ and $M_i^+(d,m)^t$ are introduced to improve the convergence speed [23]. It also guarantees the convergence from arbitrary feasible and loop-free initial point $\boldsymbol{\phi}^0$ with finite initial cost. The intuition is to provide diagonal matrices that upper bound the Hessian matrices, as the Hessians are typically difficult to compute and invert. Specifically, $M_i^+(d,m)^t$ is given by

$$\begin{aligned} M_i^+(d,m)^t = \frac{t_i^+(d,m)^t}{2} \times \operatorname{diag}\{A_{ij}(T^0) + \\ \left|\mathcal{O}(i)\backslash\mathcal{B}_i^+(d,m)^t\right| h_j^+(d,m)^t A(T^0)\}_{j\in\mathcal{O}(i)\backslash\mathcal{B}_i^+(d,m)^t}, \end{aligned} \tag{10}$$

where T^0 initial total cost at $\boldsymbol{\phi}^0$, $h_j^+(d,m)^t$ is the maximum path length among all existing result paths for (d,m) from j to destination d, operator $diag$ forms a diagonal matrix, and

$$A_{ij}(T^0) = \sup_{T<T^0} D_{ij}''(F_{ij}), \quad A(T^0) = \max_{(i,j)\in E} A_{ij}(T^0).$$

The definition of $M_i^-(d,m)^t$ is a repetition of the above, except with "+" replaced by "-". Note that $h_j^+(d,m)^t$, $h_j^-(d,m)^t$ could also be piggy-backed on the broadcast messages.

Asynchronous convergence. The proposed algorithm allows the network to update the variables one node at a time. Such asynchrony may be caused by practical constraints such as the broadcast delay in a large-scale network. We assume that at the t-th iteration, only one node i updates either $\boldsymbol{\phi}_i^-(d,m)$ or $\boldsymbol{\phi}_i^+(d,m)$ for one task $(d,m) \in S$. Let

$$\mathcal{T}_{\boldsymbol{\phi}_i^-(d,m)} = \left\{t \middle| \text{ node } i \text{ update its } \boldsymbol{\phi}_i^-(d,m) \text{ at iteration } t\right\},$$

with $\mathcal{T}_{\boldsymbol{\phi}_i^+(d,m)}$ defined similarly, Then Theorem 2 holds.

Theorem 2: Assume the network start with a feasible and loop-free initial point $\boldsymbol{\phi}^0$ and the initial total cost T^0 is finite. Let $\boldsymbol{\phi}^t$ be the variable generated by Algorithm 1 at the t-th iteration, and T^t be the corresponding total cost. Then $T^{t+1} < T^t$ for all $t \geq 1$. Moreover, if

$$\lim_{t\to\infty} \left|\mathcal{T}_{\boldsymbol{\phi}_i^-(d,m)}\right| = \infty, \quad \lim_{t\to\infty} \left|\mathcal{T}_{\boldsymbol{\phi}_i^+(d,m)}\right| = \infty,$$

then the sequence $\left\{\boldsymbol{\phi}^t\right\}_{t\to\infty}$ converges to a limit $\boldsymbol{\phi}^*$, where $\boldsymbol{\phi}^*$ is feasible and loop-free, and $\boldsymbol{\phi}^*$ optimally solves (2) with the condition in Theorem 1 holding (see [23] for the proof).

Complexity. We assume that the variables of all nodes are updated once every time slot of duration Δt, and every broadcast message is sent once in every slot. There are $2|E|$ transmissions of broadcast messages corresponding to a task in one slot, and thus totally $2|S||E|$ transmissions per slot, with on average $2|S|/\Delta t$ per link/second and at most $2\bar{d}|S|$ for each node, where $\bar{d}$ is the largest out-degree among all nodes. We assume the broadcast messages are sent in a out-of-band channel. Let t_c be the maximum transmission time for a broadcast message, and $\bar{h}$ be the maximum hop number for all data paths and result paths. Then the completion time of the broadcast procedure is at most $2\bar{h}t_c$.

The number of variables for the optimization problem in (9) is at most $\left(2\bar{d}+1\right)|S|$. Each problem is a positive semidefinite diagonal QP on a simplex, which can be solved with polynomial complexity.

V. Numerical Evaluation

In this section, we evaluate the scaled gradient projection algorithm, i.e., **SGP** proposed in Section IV by simulation. We implement several baseline algorithms and compare the performance of those against SGP over different networks and parameter settings. We also compare with a non-scaled version of SGP to show the improved convergence speed by using scaling matrices. A flow-level evaluator is available at [24].

Network	**Parameters**							
Topology	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert S\rvert$	$\lvert R\rvert$	**Link**	$\bar{d}_{ij}$	**Comp**	$\bar{s}_i$
Connected-ER	20	40	15	5	Queue	10	Queue	12
Balanced-tree	15	14	20	5	Queue	20	Queue	15
Fog	19	30	30	5	Queue	20	Queue	17
Abilene	11	14	10	3	Queue	15	Queue	10
LHC	16	31	30	5	Queue	15	Queue	15
GEANT	22	33	40	7	Queue	20	Queue	20
SW	100	320	120	10	(both)	20	(both)	20
Other Parameters	$M = 5$, $r_{\min} = 0.5$, $r_{\max} = 1.5$							

TABLE II: Simulated Network Scenarios

We summarize the simulation scenarios in Table II. Algorithms are evaluated in the following network topologies: **Connected-ER** is a connectivity-guaranteed Erdős–Rényi graph, generated by creating links uniformly at random with probability $p = 0.1$ on a linear network concatenating all nodes. **Balanced-tree** is a complete binary tree. **Fog** is a sample topology for fog-computing, where nodes on the same layer are linearly linked in a balance tree [25]. **Abilene** is the topology of the predecessor of *Internet2 Network* [26]. **GEANT** is a pan-European data network for the research and education community [26]. **SW** (small-world) is a ring-like graph with additional short-range and long-range edges [27].

Table II also summarizes the number of nodes $|V|$ and edges $|E|$, as well as the number of tasks $|S|$ in each network. We

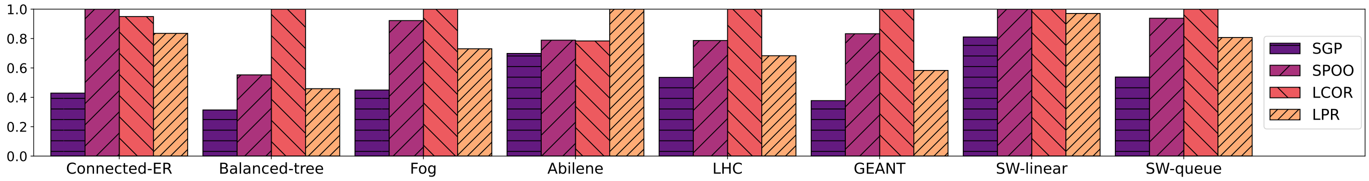
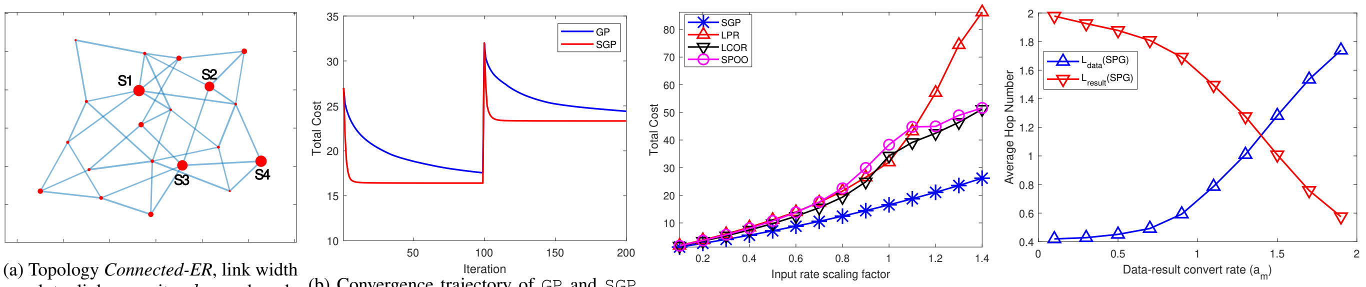


Fig. 4: Normalized total cost for network scenarios in Table II



(a) Topology *Connected-ER*, link width equal to link capacity d_{ij} and node size equal to computation capacity s_i

(b) Convergence trajectory of GP and SGP subject to server failure at S1

(c) Total cost versus scaled input rate

(d) L_{data}, L_{result} and their ratio versus a_m

Fig. 5: Topology and results in scenario *Connected-ER*

set a_m to be exponential with mean value 0.5 and truncated into interval $[0.1, 5]$, considering that most computations have a_m smaller that 1, but special types like video rendering have relatively larger a_m. Each task is uniformly assigned at random with one computation type and one destination node, along with $|R|$ random active data sources (i.e., the nodes i for which $r_i(d,m) > 0$). The input rate $r_i(d,m)$ of each active data source is chosen u.a.r. in $[r_{\min}, r_{\max}]$. **Link** is the type of link cost $D_{ij}(\cdot)$, where *Linear* denotes a linear link cost with unit cost d_{ij}, i.e. $D_{ij}(F_{ij}) = d_{ij}F_{ij}$, and *Queue* denotes a queueing delay with link capacity d_{ij}, i.e. $D_{ij}(F_{ij}) = \frac{F_{ij}}{d_{ij}-F_{ij}}$. **Comp** is the type of computation cost $C_i(G_i)$, where *Linear* denotes a weighted sum of linear cost for each type, i.e. $C_i(G_i) = s_i \sum_m w_{im} g_i^m$, and *Queue* denotes a queueing delay-like computation cost with capacity s_i, i.e. $C_i(G_i) = \frac{\sum_m w_{im} g_i^m}{s_i - \sum_m w_{im} g_i^m}$, where the weights w_{im} are u.a.r. drawn from $[1, 5]$. Parameters d_{ij} are u.a.r. drawn from $[0, 2\bar{d}_{ij}]$. Parameters s_i are exponential random variables with mean $\bar{s}_i$ for *Queue*, or uniform in $[0, 2\bar{s}_i]$ for *Linear*.

We implement the following baseline algorithms:

GP (Gradient Projection) is a non-scaled version of SGP, where the scaling matrices is simply chosen as follows,

$$M_i^-(d,m)^t = \frac{t_i^-(d,m)}{\beta} \times \text{diag}\{1, \cdots, 1, 0, 1, \cdots, 1\},$$

$$M_i^+(d,m)^t = \frac{t_i^+(d,m)}{\beta} \times \text{diag}\{1, \cdots, 1, 0, 1, \cdots, 1\},$$

where the only "0" entry on the diagonal of $M_i^-(d,m)^t$ is in the j-th position where $j = \arg\min_k \delta_{ik}^-(d,m)^t$, and similarly for $M_i^+(d,m)^t$. Note that GP and SGP are both supposed to converge to global optimum of (2), but with different convergence speeds.

SPOO (Shortest Path Optimal Offloading) fixes the routing variables to the shortest path (measured with marginal cost at $F_{ij} = 0$, accounting for the propagation delay without queueing effect), and studies the optimal offloading along these paths. Namely, SPOO only optimize T over offloading variables $\phi_{i0}^-(d,m) \in [0,1]$. It sets $\phi_{ij}^- = 1 - \phi_{i0}^-$ and $\phi_{ij}^+ = 1$ for (i,j) on the shortest path, and sets $\phi_{ij}^- = \phi_{ij}^+ = 0$ for (i,j) not on the shortest path. A similar strategy is considered in [12] with linear-topology and partial offloading.

LCOR (Local Computation Optimal Routing) computes all exogenous input flows at the their data sources, and optimally routes the result to destinations using scaled gradient projection in [28]. That is, LCOR only optimizes T over result routing variables $\phi_{ij}^+(d,m)$. It sets all $\phi_{i0}^-(d,m) = 1$ and $\phi_{ij}^-(d,m) = 0$. Note that we focus on the scenarios where such pure-local computation is feasible, i.e., the computation costs are finite if we set all $\phi_{i0}^-(d,m) = 1$.

LPR (Linear Program Rounded) is the joint routing and offloading method by [9], which does not consider partial offloading, congestible links and result flow. To adapt linear link costs in [9] to our schemes, we use the marginal cost at zero flow. To ensure sufficient communication capacity for the result flow, we assign a saturation factor of 0.7 for queueing delay costs, i.e., the data flow could not exceed 0.7 of the real capacity. Shortest path routing is used for result flow.

Fig.4 compares the total cost T of different algorithms in steady state over networks in Table II (we omit GP as it has the same steady state performance with SGP), where the bar heights of each scenario are normalized according to the worst performing algorithm. We test both linear cost and queueing delay with other parameters fixed in topology SW, labeled as SW-linear and SW-queue. Our proposed algorithm SGP significantly outperforms other baselines in all simulated scenarios, with as much as 50% improvement over LPR, which also jointly optimizes routing and task offloading but does not consider partial offloading and congestible links. The difference of case SW-linear and SW-queue suggests that our proposed algorithm promises a considerable improvement over SOTA especially when the networks are congestible. Note that LCOR and SPOO reflects the optimal objective for routing and offloading subproblems, respectively. The gain of jointly optimizing over both strategies could be inferred by comparing

SGP against LCOR and SPOO. For example, LCOR performs very poorly in topology *Balanced-tree*, because routing cannot be optimized in a tree topology.

We also perform refined experiments in Connected-ER, with the network topology and capacity shown in Fig.5a. There are 4 major servers as labeled, and we assume server S1 fails (communication and computation capability disabled, stop performing as data source or destination) at the 100-th iteration. We compare the convergence speed of GP and SGP in Fig.5b subject to S1 failure. SGP takes many fewer iterations to converge and adapt to topology change, showing the advantages of the sophisticatedly designed scaling matrices.

Fig.5c shows the change of total cost where all exogenous input rates $r_i(d,m)$ are scaled by a same factor, with other parameters fixed. The performance advantage of SGP quicly grows as the network becomes more congested, especially against LPR.

To further illustrate why SGP outperforms baselines significantly with congestion-dependent cost, we define L_{data} and L_{result} as the average travel distance (hop number) of data blocks from input to computation, and that of result blocks from generation to being delivered, respectively.

In Fig. 5d, we compare L_{data}, L_{result} for SGP over different a_m with other parameters fixed. The trajectories suggest that the average computation offloading distance grows with a_m. i.e., SGP tends to offload tasks generating larger result nearer to destination. When $a_m \gg 1$, the cost for transmitting results dominates the total cost, therefore the optimal strategy yields shorter result transmission distance. In contrast when a_m is small, the optimal strategy offloads tasks near data sources or servers, since the cost of transmitting results is low. This phenomenon demonstrates the underlying optimality of our proposed method, reaching a "balance" among the cost for data forwarding, result forwarding and computation, and therefore optimizes the total cost.

VI. Conclusion

We propose a novel joint routing and computation offloading model incorporating the result flow, partial offloading and multi-hop routing for both data and result. To the best of our knowledge, this is also the first flow model analysis of computation offloading with congestion-dependent link cost and arbitrary network topology. We propose a non-convex total cost minimization problem and optimally solve it by providing sufficient optimality conditions. We devise a fully distributed and scalable algorithm that reaches the global optimal.

References

[1] Ericsson. Ericsson mobility report (2021, Nov.). [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report

[2] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133–1145, 2020.

[3] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in internet of things," *IEEE access*, vol. 5, pp. 16 441–16 458, 2017.

[4] K. Zhu, W. Zhi, X. Chen, and L. Zhang, "Socially motivated data caching in ultra-dense small cell networks," *IEEE Network*, vol. 31, no. 4, pp. 42–48, 2017.

[5] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[6] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, 2016.

[7] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3512–3524, 2018.

[8] J. Zhang, A. Sinha, J. Llorca, A. Tulino, and E. Modiano, "Optimal control of distributed computing networks with mixed-cast traffic flows," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1880–1888.

[9] 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*, vol. 19, no. 4, 2020.

[10] H. Al-Shatri, S. Müller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[11] Q. Luo, W. Shi, and P. Fan, "Qoe-driven computation offloading: Performance analysis and adaptive method," in *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2021, pp. 1–5.

[12] X. He, R. Jin, and H. Dai, "Multi-hop task offloading with on-the-fly computation for multi-uav remote edge computing," *IEEE Transactions on Communications*, 2021.

[13] C. Funai, C. Tapparello, and W. Heinzelman, "Computational offloading for energy constrained devices in multi-hop cooperative networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 60–73, 2019.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017.

[15] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "Qos-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, 2019.

[16] B. Xiang, J. Elias, F. Martignon, and E. Di Nitto, "Joint planning of network slicing and mobile edge computing: Models and algorithms," *arXiv preprint arXiv:2005.07301*, 2020.

[17] S. Ioannidis and E. Yeh, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

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

[19] B. Liu, K. Poularakis, L. Tassiulas, and T. Jiang, "Joint caching and routing in congestible networks of arbitrary topology," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 105–10 118, 2019.

[20] Z. Chen, Q. Ma, L. Gao, and X. Chen, "Edgeconomics: Price competition and selfish computation offloading in multi-server edge computing networks," in *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*.

[21] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE transactions on communications*, vol. 25, 1977.

[22] J. Zhang, Y. Liu, and E. Yeh, "Result and congestion aware optimal routing and partial offloading in collaborative edge computing," 2022. [Online]. Available: https://arxiv.org/abs/2205.00714

[23] Y. Xi and E. M. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 4081–4106, 2008.

[24] J. Zhang, "Joint-Routing-and-Computation-2022." [Online]. Available: https://github.com/JinkunZhang/Joint-Routing-and-Computation-2022

[25] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation, caching and forwarding in data-centric computing networks," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 111–120.

[26] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, vol. 2011, pp. 1–6, 2011.

[27] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 163–170.

[28] D. Bertsekas, E. Gafni, and R. Gallager, "Second derivative algorithms for minimum delay distributed routing in networks," *IEEE Transactions on Communications*, vol. 32, no. 8, pp. 911–919, 1984.