MICS: Min-cost Joint 3C Scheduling with Asymmetric Routing

Yuanhao Wu, Edmund Yeh
Northeastern University, Boston, MA, USA
wu.yuanh@northeastern.edu, eyeh@ece.neu.edu

***Abstract*—The rapid rise of computation- & data-intensive applications, such as machine learning training and inference, poses significant challenges for joint communication, computation, and caching (3C) scheduling over the network. In this work, we study a cache-enabled computation network, where each node can forward packets, cache data, and compute tasks. Each task produces non-cacheable results and requires two types of input data: (i) cacheable data orchestrated by the cloud server and (ii) non-cacheable, user-specific data uploaded together with the request from the user. Existing joint 3C scheduling approaches either assume restrictive network topologies or neglect the size of user-specific input data, resulting in the widely used symmetric routing assumption, where computation results always return to the user following the reverse path of the corresponding requests. We propose our joint 3C framework, *MICS*, a distributed framework that incorporates user-specific data size into the model and jointly schedules network operations to minimize convex flow and computation costs over arbitrary network topologies. By breaking the symmetric routing, MICS significantly reduces system costs. In *MICS*, a dual subgradient-based control plane ensures ergodic convergence of the 3C variables under the Slater condition, while the data plane executes practical 3C operations. Simulations demonstrate up to a 32.1% improvement in average request satisfaction time compared to baseline methods.**

I. Introduction

The rapid growth of computation- and data-intensive applications poses significant challenges for efficiently allocating the limited computation and network resources. To address these challenges, computation offloading has been widely explored in various domains, including scientific computing [1] and machine learning [2]. To further reduce the latency, caching techniques have been incorporated to enable data reuse. In particular, pull-based network architectures, such as *Named Data Networking* (NDN) [3], have been adopted for data-intensive applications [4], leveraging name-based in-network caching to enhance performance.

Optimally utilizing processing, storage, and bandwidth resources in the network through intelligent communication, computation, and caching (3C) scheduling is, therefore, a fundamental challenge for achieving efficient data-intensive computation. Toward this direction, numerous joint optimization frameworks have also been proposed recently.

Works in [5], [6] study the joint 3C scheduling algorithm in a wireless edge network, but both works are restricted to the specific tree-topology edge network and can only cache at the wireless access points or users. The work in [7] designs the joint 3C algorithm for throughput optimality across arbitrary network topologies, but can only centrally schedule routing, computing, and caching operations. DECO [8], developed from VIP [9] jointly and distributively optimizes 3C scheduling for throughput optimality with an arbitrary multi-hop topology. Works in [7] and [8] both achieve throughput optimality according to the Lyapunov-drift theory, but do not minimize network costs. The LOAM framework in [10] distributively minimizes the nonlinear forwarding, caching, and computation costs with elastic caching capabilities, but it neglects the size of user-specific input data.

To the best of our knowledge, this is the first paper that distributively minimizes the nonlinear convex costs associated with traffic flow and computation workloads over a cache-enabled arbitrary topology network with a general 2-input-1-output (*2I1O*) computation model. In the 2I1O model, computation tasks output non-cacheable computation results to the requesting node and input two types of data to start: (a) cacheable network-stored data (also referred to as *content*) from cloud storage or cache nodes and (b) non-cacheable user-specific uploaded data accompanying computation requests from requester nodes. For example, the content may represent machine learning models or public datasets, while user-specific data could include task parameters or uploaded camera images in VR applications.

This work is based on the pull-based computation scheduling architecture, *NDN*. Take DECO [8] as an example, the computation tasks are remotely called in the following way in NDN. As illustrated by the symmetric routing case in Figure 1, a computation remote call begins with a user client sending a *Computation Interest Packet* (CIP). The description of the computation request, including the function name, input data names, and estimated computation cost, together with the user-specific input data, is carried by *Computation Interest Packets* (CIPs). These CIPs target clouds that storing input content as the final destinations, but can also be opportunistically accepted and executed by any node receiving them. If a node decides to compute the task locally, it will send *Content Data Interest Packets* (DIPs) to retrieve the network-stored data (content) as part of the computation input. DIPs can be satisfied by the cloud node storing the corresponding content, or by the nodes that cache it. The returned *Content Data Packets* (DPs) will go through the reverse path of the corresponding DIPs. The forwarding nodes can cache DPs during forwarding for possible reuse. After receiving all input data, the computation execution node starts the computation. When the computation finishes, the computation results return to the requester in *Computation Data Packets* (CDPs), following the reverse path of CIPs. Since CIPs are assumed to be user-specific, unlike network-stored data, the returned CDPs are assumed to be noncacheable since results are user specific.

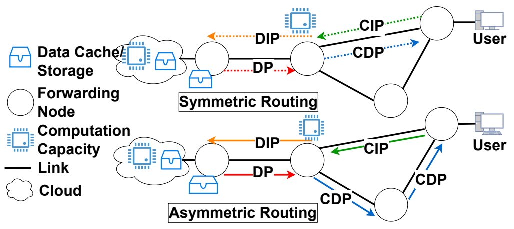


Fig. 1: The computation network with symmetric and asymmetric routing cases.

We extend DECO's model by considering nonnegligible user-specific data and breaking the symmetric routing assumption, which is widely adopted in existing joint 3C scheduling algorithms [8], [10]–[12]. The CIPs and CDPs in our model can be routed along different paths, as shown in the asymmetric routing case in Figure 1. This work is the first to theoretically demonstrate that when the sizes of the user-specific input data and the computation results are nonnegligible, breaking the symmetric routing rule helps further minimize network costs compared to the symmetric case.

Our algorithm, named "**MI**n-cost joint **C**ommunication, **C**omputation, and **C**aching **S**cheduling with asymmetric routing (MICS)" introduces a two-layer network structure, similar to DECO, consisting of a *control plane* and an *actual plane*. A distributive dual-subgradient algorithm runs in the control plane. The dual-subgradient method converges to the dual optimal solutions with suitable step sizes in each iteration. By appropriate weighted averaging of the primal variables generated in dual-subgradient iterations over the iteration indices, we get the *ergodic sequence*, which converges to the feasible optimal primal solutions when the Slater condition holds [13]. In the actual plane, we perform random forwarding, caching, and computation scheduling operations following the values of *ergodic sequence*.

We summarize our contributions as follows:

- We design a distributive joint 3C scheduling algorithm for arbitrary multi-hop network topologies to minimize the aggregated continuously differentiable and monotone increasing convex cost functions associated with link flows and computation workloads, under the 2I1O scenario.
- We propose breaking the symmetric routing rules for computation requests and results, and show that this could further reduce network costs.
- We prove that the *ergodic sequence* of primal variables generated via our MICS algorithm converges to the optimal primal solutions if the Slater condition holds.
- In the numerical evaluation, our joint 3C scheduling implementation outperforms several baseline policies over different network topologies by up to 32.1% decrease in request satisfaction time.

II. Model

A. Network Model

Consider a time-slotted joint 3C scheduling system over a multi-hop wireline network. The network topology is represented as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the 1-indexed set of N nodes, and $\mathcal{L}$ is the set of L directed links. There are two kinds of nodes in our model, the cloud and the non-cloud nodes. Each node $n \in \mathcal{N}$ can forward packets, cache received data, and execute computation tasks. The network resources have multiple capacity restrictions, where $L_n \geq 0$ is the cache capacity (in bits) at node n, $C_n \geq 0$ is the computation capacity in instructions per slot (IPS) at node n, and $C_{ab} > 0$ is the link capacity (in bits per slot) of the link $(a, b) \in \mathcal{L}$.

MICS performs forwarding and caching operations over K types of network stored *contents*, which forms a 1-indexed content set $\mathcal{K}$. We assume the size of content k is z_k in bits, and at least one node $src(k) \in \mathcal{N}$ in the network serves as the data source for the content. Such data sources are the destination of the corresponding requests. Nodes can also cache content to satisfy data requests opportunistically. Let $\mathcal{M}$ be the 1-indexed set of function types. Each computation request can be identified by the tuple $(m, k) \in \mathcal{M} \times \mathcal{K}$ with computation function m, and input content k. A (m, k) request admitted into the network at node $\theta \in \mathcal{N}$, will be further categorized and identified with the triplet (m, k, θ).

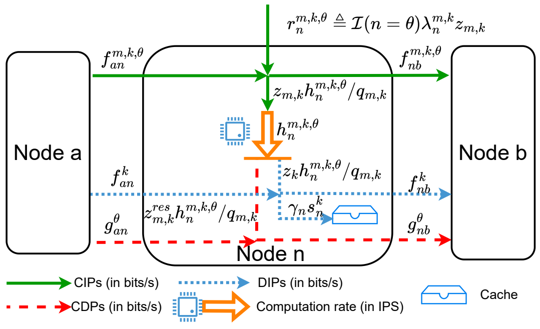

$r_n^{m,k,\theta} \triangleq \mathcal{I}(n = \theta)\lambda_n^{m,k} z_{m,k}$
$f_{an}^{m,k,\theta}$
$f_{nb}^{m,k,\theta}$
$z_{m,k} h_n^{m,k,\theta} / q_{m,k}$
$h_n^{m,k,\theta}$
Node a
Node b
f_{an}^k
$z_k h_n^{m,k,\theta} / q_{m,k}$
f_{nb}^k
$z_{m,k}^{res} h_n^{m,k,\theta} / q_{m,k}$
$\gamma_n s_n^k$
g_{an}^θ
g_{nb}^θ
Node n
CIPs (in bits/s)
DIPs (in bits/s)
Cache
CDPs (in bits/s)
Computation rate (in IPS)

Fig. 2: Flow Dynamics with Nodes $a \rightarrow n \rightarrow b$.

B. Flow Dynamics

In this section, we present the flow dynamics of our time-slotted computation network model, inspired by pull-based *Remote Procedure Call* (RPC) protocols [14]. Figure 2 shows the flow dynamics at node n, which is connected to the neighboring nodes a and b. The exogenous request arrival rates for task (m, k) at node θ are $\lambda_\theta^{m,k}$ (in requests per slot). The CIPs will be admitted together with the necessary computation parameters and user-specific input data. Assume the size of the uploaded data for each (m, k) task is $z_{m,k}$ bits. The time-averaged exogenous arrival rates of CIPs (m, k, θ) in bits at node n is $r_n^{m,k,\theta} \triangleq \mathcal{I}(n = \theta) z_{m,k} \lambda_n^{m,k}$, where $\mathcal{I}$ is the indicator function.

After being admitted into the network, if not computed locally, the CIPs will be forwarded to the next hop together with the necessary computation parameters and user-specific input data. The transmission rate of CIP flows for tasks (m,k,θ) through the link (a,b) is $f_{ab}^{m,k,\theta} \geq 0$ (in bits per slot). In the network, at least one computation cloud stores content k and has considerable computation resources. Such clouds are the destinations for the corresponding CIPs. A CIP reaching the destination will no longer be forwarded. Along the forwarding path to the clouds, nodes receiving CIPs can also opportunistically accept and compute the tasks.

If a node n decides to compute the tasks with type (m,k,θ), the corresponding assigned computation rate at node n is $h_n^{m,k,\theta} \geq 0$ in instructions per slot (IPS). The sum of the assigned computation rates at node n should be restricted by the local computation capacity C_n, i.e.,

$$\sum_{m\in\mathcal{M},k\in\mathcal{K},\theta\in\mathcal{N}} h_n^{m,k,\theta} \leq C_n \tag{1}$$

Assume that each task with type (m,k) needs $q_{m,k}$ computation resources (in instructions) to finish, and then the service rate in bits for CIPs of type (m,k,θ) at node n is $z_{m,k}h_n^{m,k,\theta}/q_{m,k}$. Due to the CIP (m,k,θ) flow conservation at each node n, we have the constraint $d_n^{m,k,\theta} \triangleq$

$$r_n^{m,k,\theta} + \sum_{a\in\mathcal{N}} f_{an}^{m,k,\theta} - \sum_{b\in\mathcal{N}} f_{nb}^{m,k,\theta} - \frac{z_{m,k}h_n^{m,k,\theta}}{q_{m,k}} \leq 0. \tag{2}$$

It means that the difference between the CIP arrival rates and the service rates is non-positive.

As Figure 1 shows, CIP flows at node n will be converted into both CDP and DIP flows if the computation is scheduled locally. The computation result size of task (m,k) is $z_{m,k}^{res}$ bits. If node $n=\theta$ (i.e., the requester node), the CDPs of type θ at node n are are not forwarded further and eliminated. If $n\neq\theta$, the transmission rate of CDPs of type θ through the link (n,b) is $g_{nb}^{\theta} \geq 0$ (in bits per slot). Due to CDP flow conservation, for any CDP of type θ at any node n, if $n\neq\theta$, we have $d_n^{\theta} \triangleq$

$$\sum_{m\in\mathcal{M},k\in\mathcal{K}} z_{m,k}^{res}h_n^{m,k,\theta}/q_{m,k} + \sum_{a\in\mathcal{N}} g_{an}^{\theta} - \sum_{b\in\mathcal{N}} g_{nb}^{\theta} \leq 0. \tag{3}$$

DIPs retrieve cacheable content, which is returned as DPs to complete the computation inputs. For the DIPs and DPs, we keep the symmetric routing feature, requiring the DPs to go through the reverse path of the corresponding DIPs, since we assume the content Interest Packet sizes are negligible. The DIPs at node n for content k can be served by either a local cache or forwarded to next hops. The overall cache status at node n can be represented by an indicator vector $\boldsymbol{s}_n = [s_n^1, s_n^2, ..., s_n^K]^T$. A cache vector $\boldsymbol{s}_n$ is *feasible* if

$$\boldsymbol{s}_n \in \{0,1\}^K, \boldsymbol{z}^T\boldsymbol{s}_n \leq L_n, \forall n\in\mathcal{N}, \forall n\in\mathcal{N}, \tag{4}$$

where $\boldsymbol{z} = [z_1, ..., z_K]^T$. The restrictions indicate (i) any element $s_n^k = 1$ if the content k is cached by the node n during slot t; otherwise, $s_n^k = 0$; (ii) cached data cannot exceed the cache space limit. The speed of data read and duplication after cache hits is γ_n (in bits/slot) at node n. Our model allows the cached content at each node to dynamically adapt for opportunistic cache hits. We define the probability of content k being cached at node n: $y_n^k \triangleq \sum_{\boldsymbol{s}_n:\boldsymbol{s}_n\in\{0,1\}^K \wedge \sum_{k\in\mathcal{K}} s_n^k z_k \leq L_n} \mathcal{I}(s_n^k = 1)P(\boldsymbol{s}_n)$ Therefore, the vector $\boldsymbol{y}_n \triangleq [y_n^1, ..., y_n^K]^T$ is inside the convex hull of cache status vectors at node n such that

$$\boldsymbol{y}_n \in Co(\boldsymbol{s}_n). \tag{5}$$

If forwarded, the transmission rate of DIPs for content $k\in\mathcal{K}$ through the link (n,b) is $f_{nb}^k \geq 0$ (in bits per slot). Due to DIP flow conservation, for any type k at any node $n \neq src(k)$, we have the constraint $d_n^k \triangleq$

$$\sum_{m\in\mathcal{M},\theta\in\mathcal{N}} \frac{z_k h_n^{m,k,\theta}}{q_{m,k}} + \sum_{a\in\mathcal{N}} f_{an}^k - \sum_{b\in\mathcal{N}} f_{nb}^k - \gamma_n y_n^k \leq 0. \tag{6}$$

The assigned forwarding rates, in bits, for the traffic flows should also satisfy the link capacity constraints. For any link $(a,b)\in\mathcal{L}$, the transmission rates follow the inequality

$$\sum_{k\in\mathcal{K}} f_{ba}^k + \sum_{m,k,\theta} f_{ab}^{m,k,\theta} + \sum_{\theta} g_{ab}^{\theta} \leq C_{ab}. \tag{7}$$

Here, we use the reverse node pair (b,a) for DIP flow summation instead of (a,b) since, as mentioned above, the size of the Interests for DIPs is neglected compared to the returned content, and we reserve the link bandwidth for the returned content when we forward DIPs.

C. Problem Formulation

For each directed link $(a,b)\in\mathcal{L}$, we define the flow cost function $D_{ab}(F_{ab})$. $D_{ab}(F_{ab})$ is a continuously differentiable, monotone increasing, and strictly convex function associated with the total flow through link (a,b). Define $F_{ab} \triangleq \sum_{k\in\mathcal{K}} f_{ba}^k + \sum_{m\in\mathcal{M},k\in\mathcal{K},\theta\in\mathcal{N}} f_{ab}^{m,k,\theta} + \sum_{\theta\in\mathcal{N}} g_{ab}^{\theta}$ to be the total flows in bits per slot through link (a,b).

For each node n, we define the computation workload cost function $G_n(h_n)$. $G_n(h_n)$ is a continuously differentiable, monotone increasing, and strictly convex function associated with the aggregated computation workloads h_n at node n, where $h_n \triangleq \sum_{m\in\mathcal{M},k\in\mathcal{K},\theta\in\mathcal{N}} h_n^{m,k,\theta}$ is the total workloads assigned at node n in instructions per second.

Cost functions D_{ab} and G_n can take various forms depending on the control objectives. A commonly used cost function represents the time-averaged waiting queue size in an M/M/1 queue, expressed as $D(F) = F/(C-F)$ [15], where F denotes the total scheduled traffic flow or computation workload, and C represents the link or computation capacity.

The primal optimization problem (PP) is cast as

$$\min_{\mathbf{f},\mathbf{g},\mathbf{h},\mathbf{y}} \sum_{(a,b)\in\mathcal{L}} D_{ab}(F_{ab}) + \sum_{n\in\mathcal{N}} G_n(h_n) \tag{PP}$$

$$\begin{aligned} \text{sbj. to}\quad & \text{(a) nonnegative orthant constraints} && \text{(nc)}\\ & f_{ab}^{m,k,\theta} \geq 0, f_{ab}^k \geq 0, g_{ab}^{\theta} \geq 0, h_n^{m,k,\theta} \geq 0, \\ & \forall m\in\mathcal{M}, k\in\mathcal{K}.(a,b)\in\mathcal{L}, \theta\in\mathcal{N}, \\ & \text{(b) capacity constraints}(1),(4),(5),(7), && \text{(cc)}\\ & \text{(c) flow conservation constraints}(2),(3),(6), && \text{(fc)} \end{aligned}$$

where $\mathbf{f}, \mathbf{g}, \mathbf{h}, \mathbf{y}$ are vectors of variables, defined as $\mathbf{f} \triangleq (f_{ab}^{m,k,\theta})_{a,b,\theta\in\mathcal{N},m\in\mathcal{M},k\in\mathcal{K}} \oplus (f_{ab}^{k})_{a,b\in\mathcal{N},k\in\mathcal{K}}$, $\mathbf{g} \triangleq (g_{ab}^{\theta})_{a,b,\theta\in\mathcal{N}}$, $\mathbf{h} \triangleq (h_{n}^{m,k,\theta})_{n,\theta\in\mathcal{N},m\in\mathcal{M},k\in\mathcal{K}}$, $\mathbf{y} \triangleq (y_n^k)_{n\in\mathcal{N},k\in\mathcal{K}}$, and $\oplus$ denotes vector concatenation

III. Algorithm

A. Asymmetric Routing

When we formulate Problem PP, we allow asymmetric routing. We now formally establish its benefits.

Theorem 1: Asymmetric routing can potentially reduce the optimal cost values, and also enlarge the feasible region of problem PP compared to the symmetric case.

Proof: Please refer to [16]. ■

B. Dual Subgradient Method

In the following sections, we will use a dual subgradient method to distributively solve the problem PP in the control plane of the computation network. We first write the corresponding dual problem (DP).

$$\max_{\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}} \quad q(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}), \quad \text{sbj. to} \quad \boldsymbol{u},\boldsymbol{v},\boldsymbol{w} \geq 0, \tag{DP}$$

where $\boldsymbol{u} \triangleq (u_n^{m,k,\theta})_{n,\theta\in\mathcal{N},m\in\mathcal{M},k\in\mathcal{K}}$, $\boldsymbol{v} \triangleq (v_n^{\theta})_{n,\theta\in\mathcal{N}}$, $\boldsymbol{w} \triangleq (w_n^{k})_{n\in\mathcal{N},k\in\mathcal{K}}$. The dual function value $q(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w})$ is obtained via solving the Lagrange function minimization problem LMP$(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w})$ given the dual variables. That is

$$\min_{\mathbf{f},\mathbf{h},\mathbf{g},\mathbf{y}} \sum_{(a,b)\in\mathcal{L}} D_{ab}(F_{ab}) + \sum_{n\in\mathcal{N}} G_n(h_n) + \sum_{n\in\mathcal{N},k\in\mathcal{K}} w_n^k d_n^k + \sum_{n,\theta\in\mathcal{N},m\in\mathcal{M},k\in\mathcal{K}} u_n^{m,k,\theta} d_n^{m,k,\theta} + \sum_{n,\theta\in\mathcal{N}} v_n^{\theta} d_n^{\theta} \tag{LMP}$$

subject to (a) nonnegative orthant constraints (nc) and (b) capacity constraints (cc). Here $v_n^\theta \equiv 0$ if $n = \theta$ and $w_n^k \equiv 0$ if $n \in \text{src}(k)$. $d_n^{m,k,\theta}, d_n^\theta, d_n^k$ come from definitions in (2), (3), and (6) but can be positive as we relax the constraints.

Algorithm 1 Dual Projected Subgradient Method

```
1: Initialization: set $\boldsymbol{u}(0) = \mathbf{0}$, $\boldsymbol{v}(0) = \mathbf{0}$, $\boldsymbol{w}(0) = \mathbf{0}$.
2: General step: for any $t = 0, 1, ...$ execute following steps:
3: (a) set a harmonic series step size $\gamma(t) = \frac{\alpha}{\beta+t}$, $\alpha, \beta > 0$;
4: (b) for $\forall a, b, n, \theta \in \mathcal{N}, k \in \mathcal{K}, m \in \mathcal{M}$, set $f_{ab}^{m,k,\theta}(t)$,
   $g_{ab}^{\theta}(t)$, $f_{ab}^{k}(t)$, $h_n^{m,k,\theta}(t)$, $\boldsymbol{y}_n(t)$ according to the solutions
   in section III-C;
5: (c) set $\boldsymbol{u}(t+1), \boldsymbol{v}(t+1), \boldsymbol{w}(t+1)$ as followings:
6: at any node $n \in \mathcal{N}$, for any $\theta \in \mathcal{N}, m \in \mathcal{M}, k \in \mathcal{K}$, set
   $u_n^{m,k,\theta}(t+1) = \left(u_n^{m,k,\theta}(t) + \gamma(t) d_n^{m,k,\theta}(t)\right)^+$
7: at any node $n \in \mathcal{N}$, for any $\theta \in \mathcal{N}$, set $v_n^\theta(t+1) =$
   $\left(v_n^\theta(t) + \gamma(t) d_n^\theta(t)\right)^+$
8: at any node $n \in \mathcal{N}$, for any $n \in \mathcal{N}, k \in \mathcal{K}$, set $w_n^k(t+1) =$
   $\left(w_n^k(t) + \gamma(t) d_n^k(t)\right)^+$
9: Here, $(x)^+ \triangleq \max\{x, 0\}$.
```

We will run Algorithm 1, the dual subgradient method [13], distributively over the network nodes to solve the optimization problem. The dual subgradient method includes updates for both the primal and dual variables. In Algorithm 1 part (a), we set the step sizes over the network based on the iteration indices t. In part (b), for each iteration t, the Lagrangian function will be distributively minimized over the network nodes to solve the LMP problem and update primal variables with these solutions. After updating primal variables in part (b), Algorithm 1 part (c) updates the dual variables distributively across network nodes using a projected subgradient method. Ignoring the overhead from synchronizing iteration indices across the network, the per-iteration computational complexity of Algorithm 1 at node n is $O(MKN^2) + O(KL_n)$, if the dynamic programming method is used to solve the caching knapsack problem, which will be further discussed in the following section.

C. Solve LMP problem.

Forwarding Solution: For ease of exposition, we first define $\delta_{ab}^{m,k,\theta}$ as the difference of the duality variables for node pair (a, b):

$$\delta_{ab}^{m,k,\theta} \triangleq \{u_a^{m,k,\theta} - u_b^{m,k,\theta}\}, \forall m \in \mathcal{M}, k \in \mathcal{K}, \theta \in \mathcal{N}$$
$$\delta_{ab}^{0,0,\theta} \triangleq \{v_a^{\theta} - v_b^{\theta}\}, \forall \theta \in \mathcal{N}, \delta_{ab}^{0,k,0} \triangleq \{w_b^k - w_a^k\}, \forall k \in \mathcal{K}.$$

Notice that since we reserve bandwidth for the returned content, $\delta_{ab}^{0,k,0}$ is the dual variable difference for DIP flows in the reverse direction of link (a, b). Next, define the maximum dual variable difference $\delta_{ab}^{m^*,k^*,\theta^*}$ for link (a, b)

$$\delta_{ab}^{m^*,k^*,\theta^*} = \Delta_{ab} \triangleq \max_{(m,k,\theta)\in\Phi} \left\{\delta_{ab}^{m,k,\theta}\right\}, \tag{8}$$

where $\Phi = \mathcal{M} \cup \{0\} \times \mathcal{K} \cup \{0\} \times \mathcal{N} \cup \{0\} / \{(0,0,0)\}$ and the maximum difference tuple is determined with ties broken arbitrarily. Then, at any node $n \in \mathcal{N}$, run algorithm 2.

Algorithm 2 Forwarding assignment given $\boldsymbol{u}(t), \boldsymbol{v}(t), \boldsymbol{w}(t)$

```
1: For any $b$ s.t. $(n, b) \in \mathcal{L}$, get $(m^*, k^*, \theta^*)$ and $\Delta_{ab}$ as (8)
2: $D'_{nb}$ is the derivative of the flow cost function
3: if $D'_{nb}(0) - \Delta_{nb} \geq 0$ then
       $x_{ab}^{m,k,\theta}(t) = 0, \forall (m, k, \theta) \in \Phi$
4: else Denote $D'^{-1}_{nb}$ as the inverse function of the first-order
   derivative of $D_{nb}$, and $\mathcal{P}_X$ as the projection operation to
   the set X. Then, $x_{nb}^{m,k,\theta}(t) =$
       $\begin{cases} \mathcal{P}_{[0,C_{nb}]}\left(D'^{-1}_{nb}(\Delta_{nb})\right), & \text{if}(m,k,\theta) = (m^*,k^*,\theta^*) \\ 0, & \text{otherwise} \end{cases}$
5: end if
6: For any $m \in \mathcal{M}, k \in \mathcal{K}, \theta \in \mathcal{N}$, set $f_{nb}^{m,k,\theta}(t) =$
   $x_{nb}^{m,k,\theta}(t)$, $g_{nb}^{\theta}(t) = x_{nb}^{0,0,\theta}(t)$, $f_{bn}^{k}(t) = x_{nb}^{0,k,0}(t)$.
```

Caching Solution: We will solve the caching subproblem at each node n

$$\max \sum_{k\in\mathcal{K}} w_n^k(t) y_n^k(t), \text{ sbj. to } \boldsymbol{y}_n(t) \in Co(\boldsymbol{s}_n)$$
$$\boldsymbol{s}_n \in \{0,1\}^K, \boldsymbol{z}^T \boldsymbol{s}_n \leq L_n, \forall n \in \mathcal{N}, \forall n \in \mathcal{N}$$

This is a linear programming problem over a convex polyhedron. Thus, the optimal solution must lie at the extreme points (vertices). Also, because the feasible set is a closed convex hull $Co(\boldsymbol{s}_n)$, the extreme points of the convex hull must form a subset of the feasible cache vectors $\{\boldsymbol{s}_n \mid \boldsymbol{s}_n \in \{0,1\}^K, \boldsymbol{z}^T\boldsymbol{s}_n \leq L_n\}$. This means that, although $\boldsymbol{y}$ is a vector with continuous real number elements, an optimal solution can always be found at a vertex of the convex hull, which must be a feasible $\boldsymbol{s}_n$. Therefore, this caching maximization problem can be converted into a knapsack problem with capacity L_n and K contents, where content k has value $w_n^k(t)$ and size z_k. This can be solved in pseudo-polynomial time with dynamic programming. In a simple case, with equal-sized content, $\forall k \in \mathcal{K}$, we cache the contents with the largest $w_n^k(t)$ values

Computation Solution: At each node $n \in \mathcal{N}$, we define

$$\delta_n^{m,k,\theta} \triangleq \frac{u_n^{m,k,\theta} z_{m,k} - v_n^\theta z_{m,k}^{res} - w_n^k z_k}{q_{m,k}}$$
$$\delta_n^{m^*,k^*,\theta^*} = \Delta_n^* \triangleq \max_{m\in\mathcal{M},k\in\mathcal{K},\theta\in\mathcal{N}} \delta_n^{m,k,\theta},$$

where the ties are broken arbitrarily. Then on each node n at iteration t, set

$$h_n^{m,k,\theta}(t) = \begin{cases} h^*(t), & \text{if}(m,k,\theta) = (m^*,k^*,\theta^*). \\ 0, & \text{otherwise} \end{cases}$$

where

$$h^*(t) = \begin{cases} 0, & \text{if } G_n'(0) - \Delta_n^{m^*,k^*,\theta^*}(t) \geq 0, \\ \mathcal{P}_{[0,C_n]}\left(G_n'^{-1}\left(\Delta_n^{m^*,k^*,\theta^*}\right)\right), & \text{otherwise.} \end{cases}$$

Here, G_n' is the first derivative of the function G_n with respect to the total computation workload h_n, and $G_n'^{-1}$ is the inverse function of G_n'.

Theorem 2: The above forwarding, caching, and computation solutions solve the LMP problem.

Proof: Please refer to [16]. ■

D. Ergodic Sequence

Define $\boldsymbol{z}(t)$ as the vector consisting of all primal variables at iteration t. Then, the ergodic sequence for the primal variables is set as follows. Let $\alpha \geq 0$ be a non-negative constant and $\bar{\boldsymbol{z}}(1) = \bar{\boldsymbol{z}}(0) = \boldsymbol{z}(0)$ and if $t \geq 2$

$$\bar{\boldsymbol{z}}(t) = \frac{\sum_{i=0}^{t-2}(i+1)^\alpha \bar{\boldsymbol{z}}(t-1) + t^\alpha \boldsymbol{z}(t-1)}{\sum_{i=0}^{t-1}(i+1)^\alpha}.$$

E. Primal Convergence

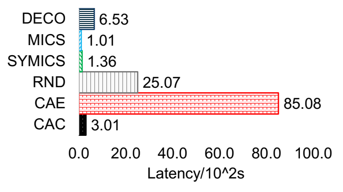
To express the convergence analysis clearly, first define Z^* as the set of feasible optimal solutions of Problem PP.

Theorem 3 (Convergence): Suppose the Slater condition holds for the programming problem PP. Then

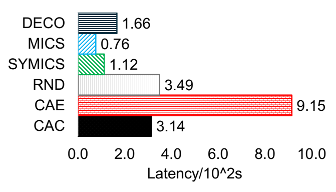
$$\min_{\boldsymbol{v}\in Z^*} \|\boldsymbol{v} - \bar{\boldsymbol{z}}(t)\|_2 \to 0, \text{ as } t \to \infty,$$

where Z^* is a nonempty, closed, and convex set.

Proof: Refer to [16] for proof. ■



(a) LHC: $\lambda = 30$



(b) GEANT: $\lambda = 10$

Fig. 3: Simulations on LHC, and GEANT. With arrival rates 30 and 10 respectively, the diagrams show the per-request latency achieved by MICS and the baseline policies.

F. 3C Scheduling in the Actual Plane

Although the ergodic sequence converges, the constraints can be violated after a finite number of iterations. Here, we propose a practical implementation for scheduling operations in the actual network. We perform our optimization algorithm with one iteration per slot and get the updated ergodic sequence. At each node n, whenever a CIP, CDP, or DIP is received, the forwarder at the node will perform the forwarding and computation scheduling operations randomly according to the probability derived from the current ergodic sequence values $\bar{\boldsymbol{z}}(t)$. Use $\bar{\cdot}$ denoting the ergodic sequence variables. When a CIP(m,k,θ) arrives, it may be forwarded to the neighboring node b with probability proportional to the weight $\left(\bar{f}_{nb}^{m,k,\theta}(t) - \bar{f}_{bn}^{m,k,\theta}(t)\right)^+$, or be scheduled for local computation with probability proportional to the weight $z_{m,k}\bar{h}_n^{m,k,\theta}(t)$. If all weights mentioned above are zero, we will schedule forwarding direction b with probability proportional to $\sum_\theta \left(\bar{f}_{nb}^{m,k,\theta}(t) - \bar{f}_{bn}^{m,k,\theta}(t)\right)^+$. If all weights above are zero, we will try to use the In the worst case, if both methods fail, we forward the CIPs along the shortest path. CDPs and DIPs are forwarded in a similar way.

For caching control, in each iteration t, treat the continuous cache variables $\bar{y}_n^k$ as the cache score. When the content is received at node n in the actual plane, do cache replacement to maximize the total cache score for the contents in the cache space. This has no theoretical guarantee, but is simple enough and works well in our simulation.

IV. Performance Evaluation

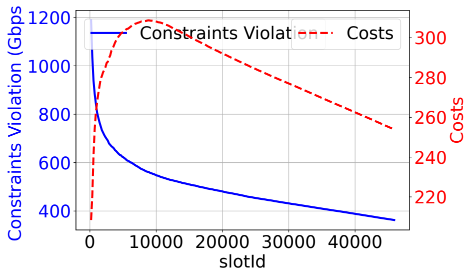


Fig. 4: GEANT: Costs & Constraints Violation over iterations.

We numerically evaluate our MICS algorithm across the LHC and GEANT topologies [16] using a task-level simulator.

The clouds and data sources are PRD, MIT, FNL, and UFL in LHC, and nodes $10-21$ in GEANT. The Requester nodes are WSC, VND, NBR, and UCSD in LHC, and nodes $0-9$ in GEANT. For all topologies, the link capacity is 100 Gbps, the cache capacity is 10 GB per node, the user-specific data is 1 GB, the content is 2 GB, the computation result is 0.5 GB each, and the cache read-out speed is 20 GB/s. Each non-cloud node has computation capacity (IPS) 2×10^8 in LHC and 4×10^8 in GEANT, respectively. Each cloud node has 4×10^9 IPS, and each task costs 10^8 instructions.

We evaluate 5 baseline policies, as shown in Figure 3. In RND, every forwarding node schedules computation with probability 0.5. In CAE, all tasks are computed at the edge. In CAC, all tasks are computed in the cloud. RND, CAE, and CAC use LFU as their cache eviction policy and shortest path as their forwarding policy. In SYMICS, we run the same control algorithm as MICS but force symmetric routing in the actual data plane. In DECO, MICS and SYMICS, the slot length is 0.2 seconds. We set the step size $\gamma(t) = \frac{500}{500+t}$ for MICS and the ergodic average parameter $\alpha = 3$.

For all the tests, we have 50 different tasks and contents, respectively. Each requester generates computation requests according to a Poisson Process at a given arrival rate λ. Indices of the function and content are randomly picked independently following the $\text{Zipf}(1)$ distribution to generate new computation requests. The tests generate new computation requests during a 20-minute active period and wait for the return of all the results. After all results are returned, we calculate the computation satisfaction time per request. In MICS and SYMICS, we run our algorithm in the control plane for $3.6 * 10^4$ iterations before the active period to preheat.

In Figure 3, we run LHC test with an arrival rate 30 requests/s per requester node and GEANT test with an arrival rate 10 requests/s per requester node. We measure the computation satisfaction time (latency) per request over the whole network. Such latency is defined as the time between when the requester generates a computation request and when the computation result is returned to the requester. We show the MICS delay in seconds and the percentage reduction in delay compared to other policies. In numerical results, MICS outperforms all the compared baseline policies by at least 25.7% decrease in latency in the LHC test and at least 32.1% decrease in latency in the GEANT test. In Figure 4, we show the convergence status of MICS in the GEANT test with an arrival rate 10. We can see that the constraint violations continue to decrease as the cost approaches certain values.

V. Conclusion

This work introduces a pull-based computation framework, MICS, with a distributed control policy that jointly optimizes the forwarding, caching, and computation scheduling under the 2I1O computation scenario. We design a joint 3C optimization algorithm in the control plane based on a dual subgradient method to minimize the convex forwarding and computation costs. We prove the primal convergence of the ergodic sequence of forwarding, caching, and computation scheduling variables. We demonstrate the significance of asymmetric routing in reducing costs compared to the symmetric case. We also design a realistic actual plane implementation following the result of the control plane, and show that MICS outperforms several baseline policies in the numerical evaluation concerning the average computation request satisfaction time and decreases the latency by up to 32.1% compared to all the other policies.

References

[1] S. A. Makhlouf and B. Yagoubi, "Data-aware scheduling strategy for scientific workflow applications in iaas cloud computing," 2019.

[2] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Toward an intelligent edge: Wireless communication meets machine learning," *IEEE communications magazine*, vol. 58, no. 1, pp. 19–25, 2020.

[3] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[4] Y. Wu, F. V. Mutlu, Y. Liu, E. Yeh, R. Liu, C. Iordache, J. Balcas, H. Newman, R. Sirvinskas, M. Lo *et al.*, "N-dise: Ndn-based data distribution for large-scale data-intensive science," in *Proceedings of the 9th ACM Conference on Information-Centric Networking*, 2022, pp. 103–113.

[5] Z. You, Q. Li, A. Pandharipande, X. Ge, R. Liu, and M. Shuai, "Joint 3c scheduling with coordinated multi-point transmission," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2023, pp. 1–6.

[6] W. Wen, Y. Cui, T. Q. Quek, F.-C. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7879–7894, 2020.

[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] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation scheduling, caching, and communication in data-intensive computing networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1058–1072, 2021.

[9] E. Yeh, T. Ho, Y. Cui, M. Burd, R. Liu, and D. Leong, "Vip: A framework for joint dynamic forwarding and caching in named data networks," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 117–126.

[10] J. Zhang and E. Yeh, "Loam: Low-latency communication, caching and computation in data-intensive computing networks," in *2025 23rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2025, pp. 1–8.

[11] J. Zhang, Y. Liu, and E. Yeh, "Optimal congestion-aware routing and offloading in collaborative edge computing," in *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 2022, pp. 121–128.

[12] J. Zhang and E. Yeh, "Delay-optimal service chain forwarding and offloading in collaborative edge computing," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 3931–3936.

[13] E. Gustavsson, M. Patriksson, and A.-B. Strömberg, "Primal convergence from dual subgradient methods for convex optimization," *Mathematical Programming*, vol. 150, pp. 365–390, 2015.

[14] M. Król, K. Habak, D. Oran, D. Kutscher, and I. Psaras, "Rice: Remote method invocation in icn," in *Proceedings of the 5th ACM Conference on Information-Centric Networking*, 2018, pp. 1–11.

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

[16] Y. Wu and E. Yeh, "Joint 3c scheduling in data-intensive computing networks with asymmetric routing," 2024. [Online]. Available: https://drive.google.com/file/d/14TeyfbYra6C9YBN5aCGJmllW4u2HD3XZ/view?usp=sharing