Joint Data Compression and Caching: Approaching Optimality with Guarantees

Jian Li
College of Information and Computer Sciences
University of Massachusetts
Amherst, MA 01003, USA
jianli@cs.umass.edu

Faheem Zafari*
Department of Electrical and Electronic Engineering,
Imperial College London,
London SW72AZ, U.K.
faheem16@imperial.ac.uk

Don Towsley
College of Information and Computer Sciences
University of Massachusetts
Amherst, MA 01003, USA
towsley@cs.umass.edu

Kin K. Leung
Department of Electrical and Electronic Engineering,
Imperial College London,
London SW72AZ, U.K.
kin.leung@imperial.ac.uk

Ananthram Swami
U.S. Army Research Laboratory
Adelphi, MD 20783 USA
ananthram.swami.civ@mail.mil

ABSTRACT

We consider the problem of optimally compressing and caching data across a communication network. Given the data generated at edge nodes and a routing path, our goal is to determine the optimal data compression ratios and caching decisions across the network in order to minimize average latency, which can be shown to be equivalent to maximizing *the compression and caching gain* under an energy consumption constraint. We show that this problem is NP-hard in general and the hardness is caused by the caching decision subproblem, while the compression sub-problem is polynomial-time solvable. We then propose an approximation algorithm that achieves a $(1 - 1/e)$-approximation solution to the optimum in strongly polynomial time. We show that our proposed algorithm achieve the near-optimal performance in synthetic-based evaluations. In this paper, we consider a tree-structured network as an illustrative example, but our results easily extend to general network topology at the expense of more complicated notations.

ACM Reference Format:
Jian Li, Faheem Zafari, Don Towsley, Kin K. Leung, and Ananthram Swami. 2018. Joint Data Compression and Caching: Approaching Optimality with Guarantees. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3184407.3184410

1 INTRODUCTION

In recent years, with the ever increasing prevalence of edge computing enabled mobile devices and applications, such as social media, weather reports, emails notifications, etc., the demand for data communication has significantly increased. As bandwidth and the power supply associated with mobile devices are limited, efficient data communication is critical.

In this paper, we consider a network of nodes, each capable of compressing data and caching a constant amount of data. A set of nodes generates real time data and a sink node collects the data from these nodes through fixed paths to serve requests for these data. However, the requests need not reach nodes that generated the data, i.e. request forwarding stops upon reaching a node on the path that has cached the requested data. Upon finding the data, it is sent along the reverse path to the sink node to serve the requests.

While each node can cache data to serve future requests so as to reduce access latency and bandwidth requirement, it incurs additional caching costs [9]. Furthermore, data compression reduces the transmission cost at the expense of computation cost [4, 26]. Thus, there is an energy consumption tradeoff among data compression, transmission, and caching to reduce latency. Since bandwidth and energy required for network operation is expensive [26], it is critical to efficiently compress, transmit and cache the data to reduce latency. This raises the following question, what is the right balance between compression and caching to minimize total communication latency for limited energy consumption?

Our primary goal is to minimize average network latency (delay) due to data transfer across the network, subject to an energy consumption constraint on compression and caching of the data. This problem is naturally abstracted and motivated by many real world applications, including wireless sensor networks (WSNs) [9], peer-to-peer networks [10], content distribution networks (CDNs) [6, 13, 23], Information Centric Networks (ICNs) [18] and so on. For example, in a hierarchical WSN, the sensors generate data, which can be compressed and forwarded to the sink node through fixed paths to serve requests generated from outside the network. These requests can be served from the intermediate nodes along the path that cache the data; if, however, data are not cached on any node along the path, the request can subsequently be forwarded to the edge sensor that generates the requested data. Similarly, in an ICN, requests for data can be served locally from intermediate caches

*Co-primary authors with equal contribution

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184410

placed between the server and origin. Both applications can be mapped into the problem we consider here.

For these and many other applications, it is natural to assume that edge nodes in the network generate data which is then compressed and transmitted by all the nodes along the path to the sink node. The sink node receives and serves requests generated outside the communication network. The intermediate nodes along the path can cache data to serve requests. However, compression, transmission, and caching consume energy, while the node power supply is usually limited. To address this challenge, our main goal is to design a lightweight and efficient algorithm with provable performance guarantees that minimizes average latency. We make the following contributions:

- We propose a formal mathematical framework for joint data compression and cache optimization. Specifically, we formulate the problem of finding optimal data compression ratios and caching locations that minimize average delay in serving requests subject to an energy constraint.
- We analyze the complexity of the problem and show that it is NP-hard in general. The hardness is caused by data allocation to the caches.
- We propose polynomial time solvable algorithms for the formulated problem. Because the original optimization problem is NP-hard and non-convex, we relax the constraints and show that the relaxed problem can be transformed into an equivalent convex optimization problem that can be solved in polynomial time. We then show that combining this solution with greedy caching allocation achieves a solution with 1/2-approximation to the optimum. Moreover, we construct a polynomial-time $(1 - 1/e)$ approximation algorithm for the problem.
- We conduct extensive simulations using synthetic network topologies and compare our proposed algorithm with benchmark techniques. Our results show that the proposed algorithm achieves near-optimal performance, and significantly outperforms benchmarks Genetic Algorithm [12], Bonmin [5], and NOMAD [21] by obtaining a feasible solution in less time for various network topologies.

The rest of the paper is organized as follows: We discuss related work in Section 2 and present our mathematical formulation in Section 3. Our main results are presented in Section 4. Numerical evaluation of our algorithms against benchmarks is given in Section 5 and finally we conclude the paper in Section 6.

2 RELATED WORK

Optimizing energy consumption has been widely studied in the literature with a primary focus on clustering [33], routing [25] and MAC protocols [15]. With the proliferation of smart sensors [26], in-network data processing, such as data aggregation, has been widely used as a way to reduce system energy cost by lowering data volume for transmission. Yu et al. [35] proposed an efficient algorithm for data compression in a tree structured networks. Nazemi et al. [26] further presented a distributed algorithm to obtain the optimal compression ratio at each node in a tree structured network so as to minimize the overall energy consumption.

However, none of these works considered caching costs. As caches have been widely deployed in many modern data communication networks, they can be used to enhance system performance by making data available close to end users, which in turn reduces the communication costs [9] and latencies.

A number of authors have studied optimization problems for cache allocation [2, 3, 6, 16, 23, 27–29, 31]. Ioannidis, Li and Shanmugam et. al [16, 22, 31] showed that it is NP-hard to determine the optimal data caching location, and an $(1 - 1/e)$ approximation algorithm was obtained through the pipage rounding algorithm [1, 8]. Beyond cache placement, [13] and [17] have jointly optimized routing and caching under a single hop bipartite graph and general graph, respectively. However, none of the existing work considered data compression and the corresponding costs for caching and compression.

The recent paper by Zafari et al. [36] is closest to the problem we tackle here. The differences between our work and [36] are mainly from two perspectives. First, the mathematical formulations (objectives) are quite different. Zafari et al. [36] considered energy tradeoffs among communication, compression, and caching in communication network, while we focus on maximizing the overall compression and caching gain by characterizing the tradeoff between compression and caching costs with an overall energy consumption constraint. This difference requires different techniques to handle the problem. Second, the methodologies are different. [36] aimed to provide a solution to the non-convex mixed integer programming problem (MINLP) with an ϵ-global[1] optimality guarantee. Since MINLP is NP-hard in general, the proposed algorithm V-SBB in [36] is complex and slow to converge to an ϵ-global optimal solution. Furthermore, it is difficult to generalize V-SBB to larger network topologies as the algorithm relies on symbolically reformulating the original non-convex MINLP problem that results in extra constraints and variables. Instead, in this paper, we focus on developing an approximation algorithm to optimize the gain defined above. In doing so, we first allow the caching decision variables to be continuous, approximate the objective function and then convert the problem into a convex one. Finally, we propose a master-slave based algorithm to efficiently solve the approximated relaxed problem, and show that the rounded solutions are feasible to the original problem with performance guarantee, and our algorithm is more efficient than that in [36] and can be applied to a larger problem size.

Note that we focus on minimizing the latency and ignore throughput issues, since we do not model congestion. Combing these two issues together and proposing efficient approximation algorithms is an interesting problem, which is out of the scope of this paper.

3 MODEL

We represent the network as a directed graph $G = (V, E)$. For simplicity, we consider a tree, with $N = |V|$ nodes, as shown in Figure 1. Node $v \in V$ is capable of storing S_v amount of data. Let $\mathcal{K} \subseteq V$ with $K = |\mathcal{K}|$ be the set of leaf nodes, i.e., $\mathcal{K} = \{1, 2, \cdots, K\}$. Time is partitioned into periods of equal length $T > 0$ and data generated

[1] ϵ-global optimality means that the obtained solution is within ϵ tolerance of the global optimal solution i.e., achieved cost/optimal cost $\leq 1 + \epsilon$. The value of ϵ depends on the requirement of different problems. Usually it is very small such as 0.0001.

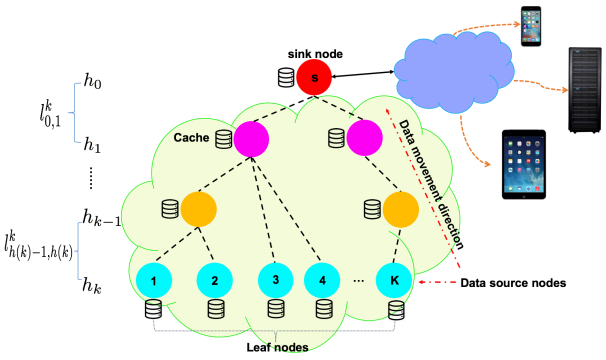

Figure 1: Tree-Structured Network Model.

Table 1: Summary of Notations

Notation	Description
$G(V,E)$	Network graph with $\|V\| = N$ nodes
$\mathcal{K}$	Set of leaf nodes with $\|\mathcal{K}\| = K$
S_v	Cache capacity at node $v \in V$
h_i^k	The i-th node on the path between leaf node k and sink node
$\delta_{k,i}$	Compression ratio for data generated by leaf node k at i^{th} node on path from $k \in \mathcal{K}$ to sink
l_{ij}	Latency of edge $(i,j) \in E$
ε_{vR}	per-bit reception cost of node $v \in V$
ε_{vT}	per-bit transmission cost of node $v \in V$
ε_{vC}	per-bit compression cost of node $v \in V$
y_k	Number of data (bits) generated at node $k \in \mathcal{K}$
$b_{k,i}$	Variable indicating whether i^{th} node on path from k to sink caches the data from leaf node $k \in \mathcal{K}$
w_{ca}	Caching power efficiency
R_k	Request rate for data from node $k \in \mathcal{K}$
W	Global Energy constraint
T	Time duration for which data are cached
δ_v	Reduction rate at node v
C_v	Set of leaf nodes that are children of node v
s.t.	Subject to

in each period are independent. Without loss of generality (w.l.o.g.), we consider one particular period in the remainder of the paper. We assume that only leaf nodes $k \in \mathcal{K}$ can generate data, and all other nodes in the tree receive and compress data from their children nodes, and transmit and/or cache the compressed data to their parent nodes during time T. In Section 3.5, we discuss how these assumptions can be relaxed. For ease of exposition, the parameters used throughout this paper are summarized in Table 1.

Our objective is to determine the optimal data compression ratio and caching locations across the network to minimize average latency under an energy constraint.

3.1 Compression and Caching Costs

Let y_k be the amount of data generated by leaf node $k \in \mathcal{K}$. Data generated at the leaf nodes are transmitted up the tree to the sink node s, which serves requests for the data generated in the network. Let $h(k)$ be the depth of node k in the tree. W.l.o.g., we assume that the sink node is located at level $h(s) = 0$. We represent the unique path from node k to the sink node by $\mathcal{H}^k$ of length $h(k)$, a sequence $\{h_0^k, h_1^k, \cdots, h_{h(k)}^k\}$ of nodes $h_j^k \in V$ such that $(h_j^k, h_{j+1}^k) \in E$, where $h_0^k \triangleq s$ (i.e., the sink node) and $h_{h(k)}^k \triangleq k$ (i.e., the node itself).

We denote the per-bit reception, transmission, and compression costs of node $v \in V$ as $\varepsilon_{vR}, \varepsilon_{vT}$, and ε_{vC}, respectively. Each node h_i^k along the path $\mathcal{H}^k$ compresses the data generated by leaf node k at a *data reduction rate*[2] $\delta_{k,i}$, where $0 < \delta_{k,i} \leq 1, \forall i,k$. The higher the value of $\delta_{k,i}$, the lower the compression will be, and vice versa. The higher the degree of data compression, the larger will be the amount of energy consumed by compression (computation). Similarly, caching data closer to the sink node can reduce the transmission cost for serving the request, however, each node only has a finite storage capacity. We study the tradeoff among the energy consumed at each node for transmitting, compressing and caching data to minimize the average delay (which will be defined in (4)) in serving a request.

We consider an energy-proportional model [9] for caching, i.e., $w_{ca}\delta_v y_v T$ units of energy is consumed if the received data y_v is cached for a duration of T where w_{ca} represents the power efficiency of caching, which strongly depends on the storage hardware technology. w_{ca} is assumed to be identical for all the nodes.

Data produced by every leaf node k is received, transmitted, and possibly compressed by all nodes in the path from the leaf node k to the root node. On the first request, the energy consumed for this processing of the data from leaf node k is

$$E_k^{\mathrm{C}} = \sum_{i=0}^{h(k)} y_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m}, \tag{1}$$

where $\prod_{m=i}^{j} \delta_{k,m} := 1$ if $i \geq j$ and $f(\delta_v) = \varepsilon_{vR} + \varepsilon_{vT}\delta_v + \varepsilon_{vC} l_v(\delta_v)$ is the sum of per-bit reception, transmission and compression cost at node v per unit time. We take $l_v(\delta_v) = 1/\delta_v - 1$ which was used in [26, 36] to capture compression costs.

Let E_k^{R} be the total energy consumed in responding to the subsequent $(R_k - 1)$ requests for the data originally generated by leaf node k. We have

$$E_k^{\mathrm{R}} = \sum_{i=0}^{h(k)} y_k (R_k - 1) \Bigg\{ f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \Bigg(1 - \sum_{j=0}^{i-1} b_{k,j}\Bigg) + \Bigg(\prod_{m=i}^{h(k)} \delta_{k,m}\Bigg) b_{k,i} \Bigg(\frac{w_{ca} T}{R_k - 1} + \varepsilon_{kT}\Bigg)\Bigg\}, \tag{2}$$

where $b_{k,j} = 1$ if node j caches data generated by k, otherwise $b_{k,j} = 0$. The first term captures the energy cost for reception, transmission, and compression up the tree from node $v_{k,i-1}$ to $v_{k,0}$ and the second term captures the energy cost for storage and transmission by node $v_{k,i}$. A detailed explanation of (1) and (2) with a toy example is provided in [24].

To consider data generated by all leaf nodes, the total energy consumed in the network is

$$E^{\mathrm{total}}(\boldsymbol{\delta}, \boldsymbol{b}) \triangleq \sum_{k \in \mathcal{K}} \left(E_k^{\mathrm{C}} + E_k^{\mathrm{R}}\right)$$

[2] defined as the ratio of the volume of the output data to the volume of input data at any node

$$= \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} - \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k (R_k - 1) \cdot f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \sum_{j=0}^{i-1} b_{k,j} + \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k (R_k - 1) \cdot \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) b_{k,i} \left(\frac{w_{ca} T}{R_k - 1} + \varepsilon_{kT}\right)$$

$$= \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k \Bigg\{ R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) b_{k,i} (w_{ca} T + (R_k - 1)\varepsilon_{kT}) \Bigg\} - \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k (R_k - 1) f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \sum_{j=0}^{i-1} b_{k,j}$$

$$\leq \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k \Bigg\{ R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) b_{k,i} (w_{ca} T + (R_k - 1)\varepsilon_{kT}) \Bigg\} \triangleq \tilde{E}^{\text{total}}(\boldsymbol{\delta}, \boldsymbol{b}), \tag{3}$$

where $\boldsymbol{\delta} = \{\delta_{k,i}, \forall k \in \mathcal{K}, i = 0, \cdots, h(k)\}$ and $\boldsymbol{b} = \{b_{k,i}, \forall k \in \mathcal{K}, i = 0, \cdots, h(k)\}$.

Note that $\tilde{E}^{\text{total}}(\boldsymbol{\delta}, \boldsymbol{b})$ is an upper bound of $E^{\text{total}}(\boldsymbol{\delta}, \boldsymbol{b})$, which is tight when there is no caching in the network. In the following optimization, we use $\tilde{E}^{\text{total}}(\boldsymbol{\delta}, \boldsymbol{b})$ for energy constraint.

3.2 Latency Performance

W.l.o.g., we consider the path $\{h_0^k, h_1^k, \cdots, h_{h(k)}^k\}$. A request for data generated by leaf node k is forwarded along this path from the root node s until it reaches the node that has cached the requested data. Upon finding the requested data, it is propagated along the reverse direction of the path, i.e., carrying the requested data to the sink node where the request originated. To capture the average latency due to data transfer at any particular link, we associate each link with a cost $l_{i,j}$ for $(i,j) \in E$, representing the latency of transmitting the data across the link (i,j). Denote the latency associated with path $\{h_0^k, h_1^k, \cdots, h_{h(k)}^k\}$ as $\{l_{0,1}^k, l_{1,2}^k, \cdots, l_{h(k)-1,h(k)}^k\}$.

Then the overall latency for all the paths is

$$L(\boldsymbol{\delta}, \boldsymbol{b}) = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \prod_{j=0}^{i} (1 - b_{k,j}). \tag{4}$$

3.3 Optimization

Our objective is to determine the optimal compression ratio $\boldsymbol{\delta} = \{\delta_{k,i}, \forall k \in \mathcal{K}, i = 0, \cdots, h(k)\}$ and data caching location $\boldsymbol{b} = \{b_{k,i}, \forall k \in \mathcal{K}, i = 0, \cdots, h(k)\}$ to minimize the expected total latency subject to the energy constraint. That is,

$$\min \quad L(\boldsymbol{\delta}, \boldsymbol{b}) \tag{5a}$$

$$\text{s.t.} \quad \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k \Bigg\{ R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) b_{k,i} \cdot (w_{ca} T + (R_k - 1)\varepsilon_{kT}) \Bigg\} \leq W, \tag{5b}$$

$$b_{k,i} \in \{0,1\}, \forall k \in \mathcal{K}, i = 0, \cdots, h(k), \tag{5c}$$

$$\sum_{k\in C_v} b_{k,h(v)} y_k \prod_{j=h(k)}^{h(v)} \delta_{k,j} \leq S_v, \forall\, v \in V, \tag{5d}$$

$$\sum_{i=0}^{h(k)} b_{k,i} \leq 1, \forall k \in \mathcal{K}. \tag{5e}$$

Now suppose that there is no compression or caching, then all the requests need to be served from leaf nodes. The corresponding total latency L^u is given as

$$L^u = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} y_k l_{i,i+1}^k R_k. \tag{6}$$

Clearly, L^u is an upper bound on the expected total latency.

Then the *compression and caching gain* is

$$G(\boldsymbol{\delta}, \boldsymbol{b}) = L^u - L(\boldsymbol{\delta}, \boldsymbol{b}) = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j=0}^{i} (1 - b_{k,j})\right). \tag{7}$$

An equivalent *optimization problem* to (5) is to maximize the above gain, given as follows

$$\max \quad G(\boldsymbol{\delta}, \boldsymbol{b}) \qquad \text{s.t.} \quad \text{Constraints in (5).} \tag{8}$$

The objective in (8) is to maximize the expected compression and caching gain. Constraint (5b) ensures that the total energy consumption in the network as given in (3) is limited. Constraint (5c) constrains our caching decision variables to be binary. Constraint (5d) ensures that each cache v stores no more than S_v amount of data. Constraint (5e) ensures that at most one copy of the generated data can be cached at any node along the path between the leaf and the sink node. Each node potentially compresses data from different leaf nodes differently; the coupling occurs due to the storage and energy constraints.

3.4 Complexity Analysis

There are two decision variables in (8), i.e., the compression ratio and the caching decision variables. In the following, we show the impact of these variables on the hardness of our problem, i.e., we consider two cases, (i) given the caching decisions variables $\boldsymbol{b}$; (ii) given the compression ratio $\boldsymbol{\delta}$.

3.4.1 Given Caching Decisions: For given caching decision variables $\boldsymbol{b}$, the optimization problem in (8) turns into a geometric programming problem over the compression ratio $\boldsymbol{\delta}$ that can be solved in polynomial time.

Theorem 3.1. *Given fixed caching decisions $\boldsymbol{b}$, the optimization problem in (8) is polynomial-time solvable.*

Proof. Once $\boldsymbol{b}$ is given, (8) becomes a geometric programming problem in $\boldsymbol{\delta}$; we will show in Section 4.2 that it can be transformed into a convex optimization problem, which can be solved in polynomial time. □

3.4.2 Given Compression Ratios: Given compression ratios $\boldsymbol{\delta}$, the optimization problem in (8) is only over the caching decision variables $\boldsymbol{b}$. Hence, we obtain an integer programming problem, which is NP-hard.

Theorem 3.2. *Given a fixed compression ratio $\boldsymbol{\delta}$, the optimization problem in (8) is NP-hard.*

Proof. We prove the hardness by reduction from the classical job-shop problem which is NP-hard [19].

We can reduce the job-shop problem to our problem in (8) with fixed compression ratios $\boldsymbol{\delta}$ as follows. Consider each node $v \in V$ in our model to be a machine M_i. Denote the set of machines as $\mathcal{M} = \{M_1, M_2 \cdots M_{|V|}\}$. The caching decision constitutes the set of jobs $\mathcal{J} = \{J_1, J_2\}$, where J_1 means that the data is cached and J_2 means otherwise. Let $\mathcal{X}$ be the set of all sequential job assignments to different machines so that every machine performs every job only once. The elements $x \in \mathcal{X}$ can be written as $2 \times |V|$ matrices, where column v order-wise lists the sequential jobs that the machine M_v will perform. There is a cost function C that captures the cost (i.e., latency) for any machine to perform a particular job. Our objective in the optimization problem (8) is to find assignments of job $x \in \mathcal{X}$ to minimize the latency or maximize the gain, which is equivalent to the classical job-shop problem. Since job-shop problem is NP-hard [19], our problem in (8) with given compression ratios $\boldsymbol{\delta}$ is also NP-hard. □

Therefore, given the results in Theorems 3.1 and 3.2, we know that our optimization problem is NP-hard in general.

Corollary 3.3. *The optimization problem defined in (8) is NP-hard.*

3.5 Relaxation of Assumptions

We made several assumptions in the above for the sake of model simplicity. In the following, we discuss how these assumptions can be relaxed.

First, the network is assumed to be structured as a tree, however, we can easily relax this assumption by incorporating routing into our joint optimization problem. We take the tree structure as our motivating example since it is a simple and representative topology that captures the key parameters in the optimization formulation without introducing more complexity for a general network topology.

Second, while we only allow leaf nodes to generate data, our model can be extended to allow intermediate nodes to generate data at the cost of added complexity, i.e., the number of decision variables will be increased to represent the caching decision and compression ratio for the data produced at the intermediate nodes. Furthermore, rather than having a constant R_k requests for data generated at the leaf node k, we can generalize our approach to the case where R_k for various leaf nodes are drawn from a distribution such as the Zipf distribution [9].

Third, in our model, we assume that the requests for the data that are generated and valid for a time period T are known. But our solutions can be applied to an online setting with predicted user requests.

4 APPROXIMATION ALGORITHM

Since our optimization problem (8) is NP-hard, we focus on developing efficient approximation algorithms. In particular, we develop a polynomial-time solvable algorithm that produces compression ratios and cache decisions with a constant approximation of the minimum average latency. In the following, we first derive several properties that allow us to develop such an approximation algorithm. Then we discuss how to obtain a constant approximation solution in polynomial time.

4.1 Properties of the Problem Formulation

In this section, we show that (8) is a submodular maximization problem under matroid constraints. To begin, we first review the concepts of submodular functions and matroids.

Definition 4.1. (Submodular function [30]) If Ω is a finite set, a submodular function is a set function $f: 2^{\Omega} \rightarrow \mathbb{R}$, where 2^{Ω} denotes the power set of Ω, which satisfies one of the following equivalent conditions:

(1) For every $X, Y \subseteq \Omega$ with $X \subseteq Y$ and every $x \in \Omega \setminus Y$, we have $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$;
(2) For every $S, T \subseteq \Omega$, we have $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$;
(3) For every $X \subseteq \Omega$ and $x, y \in \Omega \setminus X$, we have $f(X \cup \{x\}) + f(X \cup \{y\}) \geq f(X \cup \{x, y\}) + f(X)$.

Definition 4.2. (Monotone sub-modular function [20]) A submodular function f is monotone if for every $T \subseteq S$, we have $f(T) \geq f(S)$.

Definition 4.3. (Matroid [32]) A finite matroid M is a pair $(E, \mathcal{I})$, where E is a finite set and $\mathcal{I}$ is a family of subsets of E (called the independent sets) with the following properties:

(1) The empty set is independent, i.e., $\emptyset \in \mathcal{I}$;
(2) Every subset of an independent set is independent, i.e., for each $A \subset B \subset E$, if $B \in \mathcal{I}$ then $A \in \mathcal{I}$;
(3) If A and B are two independent set of $\mathcal{I}$ and A has more elements than B, then there exists $x \in A \setminus B$ such that $B \cup \{x\}$ is in $\mathcal{I}$.

Given the above concepts, we easily obtain the following result

Theorem 4.4. *The objective function in (8) is monotone and submodular, and the constraints in (8) are matroid.*

The proof is simply to verify that the objective function and constraints in (8) satisfy Definitions 4.1, 4.2 and 4.3. We skip the details due to space limitations.

Corollary 4.5. *Since (8) is a sub-modular maximization problem under matroid constraints, a solution with $1/2$ approximation from the optimum can be constructed by a greedy algorithm*[3].

Now we are ready to develop a polynomial-time solvable approximation algorithm with improved approximation ratio when compared to the greedy algorithm. Since the optimization problem in (8) is a non-convex mixed integer non-linear programing problem (MINLP), we first relax the integer variables and transform it into a

[3]Start with caching all data at the leaf nodes, then compute the optimal compression ratio, and then iteratively add the data to caches by selecting feasible caching decisions at each step that leads to the largest increase in the compression and caching gain.

convex optimization problem, which can be solved in polynomial time. Then we round the achieved solutions to ones that satisfy the original integer constraints, if there are any fractional solutions.

4.2 Convex Relaxation

We first relax the integer variables $b_{k,i} \in \{0,1\}$ to $\tilde{b}_{k,i} \in [0,1]$ for $\forall k \in \mathcal{K}$ and $i = 0, \cdots, h(k)$, in (4), (5), (7) and (8). Let μ be the joint distribution over $\boldsymbol{b}$, and let $\mathbb{P}_\mu(\cdot)$ and $\mathbb{E}_\mu(\cdot)$ be the corresponding probability and expectation with respect to μ, i.e.,

$$\tilde{b}_{k,i} = \mathbb{P}_\mu[b_{k,i} = 1] = \mathbb{E}_\mu[b_{k,i}]. \tag{9}$$

Then the relaxed expected latency and gain are given as

$$\begin{aligned} L(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \prod_{j=0}^{i} (1-\tilde{b}_{k,j}), \\ G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) &= L^u - L(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \\ &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j=0}^{i} (1-\tilde{b}_{k,j})\right). \end{aligned} \tag{10}$$

Therefore, the relaxed optimization problem is

$$\begin{aligned} \max \quad & G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)} y_k \left\{ R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) \tilde{b}_{k,i} \right. \\ & \qquad \left. \cdot (w_{ca} T + (R_k - 1)\varepsilon_{kT}) \right\} \le W, \\ & \tilde{b}_{k,i} \in [0,1], \forall k \in \mathcal{K}, i = 0, \cdots, h(k), \\ & \sum_{k\in C_v} \tilde{b}_{k,h(v)} y_k \prod_{j=h(k)}^{h(v)} \delta_{k,j} \le S_v, \forall\, v \in V, \\ & \sum_{i=0}^{h(k)} \tilde{b}_{k,i} \le 1, \forall k \in \mathcal{K}. \end{aligned} \tag{11}$$

Theorem 4.6. *Suppose that $(\boldsymbol{\delta}^*, \boldsymbol{b}^*)$ and $(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*)$ are the optimal solutions to (8) and (11), respectively, then*

$$G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \ge G(\boldsymbol{\delta}^*, \boldsymbol{b}^*). \tag{12}$$

Proof. The results hold since (11) maximizes the same objective function over a larger domain due to relaxation of integer variables $\boldsymbol{b}$ and energy constraint in (3). □

However, (11) is not a convex optimization problem. Since $e^x \approx 1 + x$ for $x \to 0$ and $\log(1-x) \approx -x$ for $x \to 0$, we obtain an approximation for (10). The approximated expected total latency and *approximated compression and caching gain* are given as follows

$$\begin{aligned} L(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \prod_{j=0}^{i} (1-\tilde{b}_{k,j}) \\ &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k e^{\sum_{j=0}^{i} \log(1-\tilde{b}_{k,j})} \\ &\overset{(a)}{\approx} \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \left(1 - \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}\right) \\ &\triangleq \tilde{L}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}), \end{aligned}$$

$$\tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) = L^u - \tilde{L}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \cdot \left(1 - \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}\right)\right), \tag{13}$$

where (a) is based on the two approximate properties discussed above.

Then, the *relaxed approximated optimization problem* is given as

$$\begin{aligned} \max \quad & \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) && \text{(14a)} \\ \text{s.t.} \quad & \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)} y_k \left\{ R_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m} + \left(\prod_{m=i}^{h(k)} \delta_{k,m}\right) \tilde{b}_{k,i} \right. \\ & \qquad \left. \cdot (w_{ca} T + (R_k - 1)\varepsilon_{kT}) \right\} \le W, && \text{(14b)} \\ & \tilde{b}_{k,i} \in [0,1], \forall k \in \mathcal{K}, i = 0, \cdots, h(k), && \text{(14c)} \\ & \sum_{k\in C_v} \tilde{b}_{k,h(v)} y_k \prod_{j=h(k)}^{h(v)} \delta_{k,j} \le S_v, \forall\, v \in V, && \text{(14d)} \\ & \sum_{i=0}^{h(k)} \tilde{b}_{k,i} \le 1, \forall k \in \mathcal{K}. && \text{(14e)} \end{aligned}$$

However, $\tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ is not concave. In the following, we transform it into a convex term through Boyd's method (Section 4.5 [7]) to deal with posynomial terms in (14a), (14b) and (14d).

4.2.1 Transformation of the Objective Function. Given our approximated objective function

$$\tilde{L}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \triangleq \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \left(1 - \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}\right), \tag{15}$$

we define two new variables as follows

$$\begin{aligned} \log(\tilde{b}_{k,j}) &\triangleq u_{k,j}, \quad i.e., \quad \tilde{b}_{k,j} = e^{u_{k,j}}, \\ \log \delta_{k,m} &\triangleq \tau_{k,m}, \quad i.e., \quad \delta_{k,m} = e^{\tau_{k,m}}. \end{aligned} \tag{16}$$

Then the approximated objective function can be transformed into

$$\tilde{L}(\boldsymbol{\tau}, \boldsymbol{u}) \triangleq \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \sum_{m=i+1}^{h(k)} e^{\tau_{k,m} + \log(y_k R_k l_{i,i+1}^k)} \left(1 - \min\left\{1, \sum_{j=0}^{i} e^{u_{k,j}}\right\}\right). \tag{17}$$

Therefore, we can transform $\tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ into

$$\begin{aligned} &\tilde{G}(\boldsymbol{\tau}, \boldsymbol{u}) = L^u - \tilde{L}(\boldsymbol{\tau}, \boldsymbol{u}) \\ &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} e^{\log(R_k y_k l_{i,i+1}^k)} \left(1 - \sum_{m=i+1}^{h(k)} e^{\tau_{k,m}} \left(1 - \min\left\{1, \sum_{j=0}^{i} e^{u_{k,j}}\right\}\right)\right). \end{aligned} \tag{18}$$

Next we need to transform the constraints following Boyd's method.

4.2.2 Transformation of the Constraints.

Constraint (14b): We take the left hand side of the constraint and transform it. To simplify, we divide the equation into multiple parts,

$$\underbrace{\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k y_k f(\delta_{k,i}) \prod_{m=i+1}^{h(k)} \delta_{k,m}}_{\text{Part 1}} + \underbrace{\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k w_{ca} T \tilde{b}_{k,i} \prod_{m=i}^{h(k)} \delta_{k,m}}_{\text{Part 2}} + \underbrace{\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} y_k \varepsilon_{kT}(R_k-1)\tilde{b}_{k,i} \prod_{m=i}^{h(k)} \delta_{k,m}}_{\text{Part 3}}. \tag{19}$$

Part 1: From (16), i.e., $\tau_{k,i} = \log \delta_{k,i}$, we have

$$\begin{aligned}\textbf{Part } 1 &= \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k y_k(\varepsilon_{kR}-\varepsilon_{kC}+\delta_{k,i}\varepsilon_{kT}+\frac{\varepsilon_{kC}}{\delta_{k,i}}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \\ &= \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k y_k(\varepsilon_{kR}-\varepsilon_{kC}+e^{\tau_{k,i}}\varepsilon_{kT}+\frac{\varepsilon_{kC}}{e^{\tau_{k,i}}}) \prod_{m=i+1}^{h(k)} \delta_{k,m} \\ &= \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k y_k(\varepsilon_{kR}-\varepsilon_{kc}+\varepsilon_{kT}e^{\tau_{k,i}}+\varepsilon_{kc}e^{-\tau_{k,i}}) e^{\sum_{m=i+1}^{h(k)}\tau_{k,m}}.\end{aligned} \tag{20}$$

Part 2: From (16), i.e., $\tilde{b}_{k,j} = e^{u_{k,j}}$, we have

$$\textbf{Part } 2 = \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log(y_k w_{ca} T)+u_{k,i}}. \tag{21}$$

Part 3: Similarly, we have

$$\textbf{Part } 3 = \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log\,(y_k(R_k-1)\varepsilon_{kT})+u_{k,i}}. \tag{22}$$

Combining (20), (21) and (22), Constraint (14b) becomes

$$\begin{aligned}&\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k(y_k\varepsilon_{kR}-y_k\varepsilon_{kc}+y_k\varepsilon_{kT}e^{\tau_{k,i}}+y_k\varepsilon_{kc}e^{-\tau_{k,i}}) e^{\sum_{m=i+1}^{h(k)}\tau_{k,m}} \\ &\quad+\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log(y_k w_{ca}T)+u_{k,i}} \\ &\quad+\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log\,(y_k(R_k-1)\varepsilon_{kT})+u_{k,i}} \le W,\end{aligned} \tag{23}$$

which is convex in $\boldsymbol{\tau}$ and $\boldsymbol{u}$ on the left hand side, respectively.
Constraint (14d): Similarly, we have

$$\sum_{k\in C_v} e^{\sum_{j=h(k)}^{h(v)}\tau_{k,j}+\log y_k+u_{k,h(v)}} \le S_v, \tag{24}$$

which is convex in $\boldsymbol{\tau}$ and $\boldsymbol{u}$ on the left hand side, respectively.

4.2.3 Optimization Problem in Convex Form.

Following the transformation given in (18), (23) and (24), we obtain the convex form for the optimization problem, i.e.,

$$\begin{aligned}\max\quad & \tilde{G}(\boldsymbol{\tau},\boldsymbol{u}) \\ \text{s.t.}\quad & \sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} R_k y_k(\varepsilon_{kR}-\varepsilon_{kc}+\varepsilon_{kT}e^{\tau_{k,i}}+\varepsilon_{kc}e^{-\tau_{k,i}}) e^{\sum_{m=i+1}^{h(k)}\tau_{k,m}} \\ &\quad+\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log(y_k w_{ca}T)+u_{k,i}} \\ &\quad+\sum_{k\in\mathcal{K}}\sum_{i=0}^{h(k)} e^{\sum_{m=i}^{h(k)}\tau_{k,m}+\log\,(y_k(R_k-1)\varepsilon_{kT})+u_{k,i}} \le W, \\ & e^{u_{k,i}} \in [0,1], \forall k\in\mathcal{K}, i=0,\cdots,h(k), \\ & \sum_{k\in C_v} e^{\sum_{j=h(k)}^{h(v)}\tau_{k,j}+\log y_k+u_{k,h(v)}} \le S_v, \\ & \sum_{i=0}^{h(k)} e^{u_{k,i}} \le 1, \forall k\in\mathcal{K}.\end{aligned} \tag{25}$$

Theorem 4.7. *The optimization problem given in (25) is convex in $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively.*

Proof. It can be easily checked that the objective function in (25) satisfies the second order condition [7] for $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively. We omit the details due to space constraints. □

Remark 1. *Note that the optimization problem given in (25) is convex in $\boldsymbol{\tau}$ for a given $\boldsymbol{u}$, and vice versa. In the following, we will present an efficient master-slave algorithm to solve the convex optimization problem in $\boldsymbol{\tau}$ and $\boldsymbol{u}$, respectively.*

4.3 Efficient Algorithms

Theorem 4.8. *The optimization problems given in (14a) and (25) are equivalent.*

Proof. This is clear from the way we convexified the problem. □

Note that after the convex relaxation and transformation, the optimization problem in (25) is point-wise convex in $\boldsymbol{\tau}$ and $\boldsymbol{u}$. We focus on designing a polynomial-time solvable algorithm.
Algorithm: We consider the master-slave algorithm shown in Algorithm 1, i.e., given a fixed $\boldsymbol{\tau}_0$, we solve (25) to obtain $\boldsymbol{u}_0$, and then given $\boldsymbol{u}_0$, we solve (25) to obtain $\boldsymbol{\tau}_1$. We repeat the above process until that the values of $\boldsymbol{\tau}$ and $\boldsymbol{u}$ converge[4][5]. We denote this as the optimal solution of (25) as $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$[6].

Given the optimal solution to (25) as $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$, then from Theorem 4.8, we know there exists $(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$, which is the optimal

[4] If the difference between the current value and the previous one is within a tolerance, we say the value converges.
[5] Since our objective function is a function of the variables $\boldsymbol{u}$ and $\boldsymbol{\tau}$, once these variables converge, the value of the objective function must converge. As we are interested in the objective value, in Algorithm 1, we write the convergence criteria with respect to the objective function value, where ϵ equals to 0.001.
[6] Note that our master-slave algorithm is very efficient to solve this convex optimization problem, we can obtain a solution within one or two iterations.

Algorithm 1 Master-Slave Algorithm

Input: R_k, y_k, W, $\boldsymbol{l}$, obj_0
Output: $\boldsymbol{b}, \boldsymbol{\delta}$, obj_f
Step 1: Initialize $\boldsymbol{u}$
Step 2: $\boldsymbol{\tau} \longleftarrow$ Random($\boldsymbol{lb}, \boldsymbol{ub}$) ▷ Generate random $\boldsymbol{\tau}$ between lower bound $\boldsymbol{lb}$ and upper bound $\boldsymbol{ub}$
while $\text{obj}_\chi - \text{obj}_{\chi-1} \geq \epsilon$ **do**
  Step 3: $\boldsymbol{u}_\chi \longleftarrow$ Convex(master, $\boldsymbol{\tau}_\chi$) ▷ Solve the master optimization problem for $\boldsymbol{u}_\chi$
  Step 4: $\boldsymbol{\tau}_\chi \longleftarrow$ Convex(slave, $\boldsymbol{u}_\chi$) ▷ Solve the slave optimization problem for $\boldsymbol{\tau}_\chi$
  Step 5: $(\boldsymbol{b}_\chi, \boldsymbol{\delta}_{\boldsymbol{\chi}}, \text{obj}_\chi) \longleftarrow$ Rounding($\boldsymbol{u}_\chi, \boldsymbol{\tau}_\chi$) ▷ Round the values of $\boldsymbol{u}_\chi$, remap $\boldsymbol{u}_\chi, \boldsymbol{\tau}_\chi$ to $\boldsymbol{b}_\chi$ and $\boldsymbol{\delta}_\chi$ and obtain the new objective function value

solution to (14a) such that $\tilde{G}(\boldsymbol{\tau}^*, \boldsymbol{u}^*) = \tilde{G}(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$ and $G(\boldsymbol{\tau}^*, \boldsymbol{u}^*) = G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$.

Theorem 4.9. *Denote the optimal solutions to (11) and (25) as* $(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*)$ *and* $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$*, respectively. Then, we have*

$$\left(1 - \frac{1}{e}\right) G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \leq G(\boldsymbol{\tau}^*, \boldsymbol{u}^*) \leq G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*). \tag{26}$$

Proof. Consider any $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ that satisfies the constraints in (11) and (14a).

First, we show that $G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \leq \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$, as follows

$$\begin{aligned}
G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) &\overset{(a)}{=} \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \mathbb{E}\left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j=0}^{i} (1 - b_{k,j})\right) \\
&= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \\
&\qquad \cdot \prod_{m=i+1}^{h(k)} \delta_{k,m} \mathbb{E}\left[\prod_{j=0}^{i} (1 - b_{k,j})\right] \\
&= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \\
&\qquad \cdot \prod_{m=i+1}^{h(k)} \delta_{k,m} \mathbb{E}\left[1 - \min\left\{1, \sum_{j=0}^{i} b_{k,j}\right\}\right] \\
&\overset{(b)}{\leq} \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \\
&\qquad \cdot \prod_{m=i+1}^{h(k)} \delta_{k,m} \left(1 - \min\left\{1, \mathbb{E}\left[\sum_{j=0}^{i} b_{k,j}\right]\right\}\right) \\
&= \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}),
\end{aligned} \tag{27}$$

where the expectation $\mathbb{E}$ in (a) is taken over $\boldsymbol{b}$ due to the linear relaxation, and (b) holds true due to the concavity of the min operator.

Next, we show that $G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \geq \left(1 - \frac{1}{e}\right) \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$, as follows

$$\begin{aligned}
G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) &= \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j=0}^{i} (1 - \tilde{b}_{k,j})\right) \\
&\geq \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - \prod_{j=0}^{i} (1 - \tilde{b}_{k,j})\right) \\
&\overset{(a)}{\geq} \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \left(1 - (1 - 1/i)^i\right) \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\} \\
&\overset{(b)}{\geq} \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\},
\end{aligned} \tag{28}$$

where (a) holds true since [11, 14]

$$1 - \prod_{j=0}^{i} (1 - \tilde{b}_{k,j}) \geq \left(1 - (1 - 1/i)^i\right) \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}, \tag{29}$$

and (b) holds true since $(1 - 1/i)^i \leq 1/e$. Also we have

$$\begin{aligned}
&\left(1 - \frac{1}{e}\right) \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) = \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \left(1 - \frac{1}{e}\right) \tilde{L}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \\
&= \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} \\
&\qquad \cdot \prod_{m=i+1}^{h(k)} \delta_{k,m} y_k R_k l_{i,i+1}^k \left(1 - \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}\right) \\
&= \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l_{i,i+1}^k - \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} y_k R_k l_{i,i+1}^k \\
&\qquad \cdot \left(1 - \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\}\right) \\
&= \left(1 - \frac{1}{e}\right) \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} y_k R_k l_{i,i+1}^k \min\left\{1, \sum_{j=0}^{i} \tilde{b}_{k,j}\right\},
\end{aligned} \tag{30}$$

then from (28) and (30), we immediately have

$$G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \geq \left(1 - \frac{1}{e}\right) \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}), \tag{31}$$

therefore, for any $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ that satisfies the constraints in (11) and (14a), we have

$$\left(1 - \frac{1}{e}\right) \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \leq G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) \leq \tilde{G}(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}). \tag{32}$$

Now, since $(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*)$ is optimal to (11), then

$$G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**}) \leq G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*). \tag{33}$$

Similarly, since $(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$ is optimal to (14a),

$$G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \leq \tilde{G}(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \leq \tilde{G}(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**}) \leq \frac{e}{e-1} G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**}), \tag{34}$$

where the first and third inequality hold due to (32).

Therefore, we have

$$\left(1 - \frac{1}{e}\right) G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \leq G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**}) \leq G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*), \tag{35}$$

i.e.,

$$\left(1 - \frac{1}{e}\right) G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*) \leq G(\boldsymbol{\tau}^*, \boldsymbol{u}^*) \leq G(\tilde{\boldsymbol{\delta}}^*, \tilde{\boldsymbol{b}}^*). \tag{36}$$

□

Since (25) is a convex optimization problem, $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$ can be obtained in strongly polynomial time.

4.4 Rounding

To provide a constant approximation solution to (8), the optimal solution $(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$ needs to be rounded.

Property: W.l.o.g., we consider a feasible solution $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ and assume that there are two fractional solutions $\tilde{b}_{k,j}$ and $\tilde{b}_{k,l}$. We define

$$\begin{aligned} \epsilon_1 &= \min\{\tilde{b}_{k,j}, 1-\tilde{b}_{k,l}\}, \\ \epsilon_2 &= \min\{1-\tilde{b}_{k,j}, \tilde{b}_{k,l}\}, \end{aligned} \tag{37}$$

and set

$$\begin{aligned} \tilde{\boldsymbol{b}}'(1) &= (\tilde{\boldsymbol{b}}_{-(j,l)}, \tilde{b}_{k,j}-\epsilon_1, \tilde{b}_{k,l}+\epsilon_1), \\ \tilde{\boldsymbol{b}}'(2) &= (\tilde{\boldsymbol{b}}_{-(j,l)}, \tilde{b}_{k,j}+\epsilon_2, \tilde{b}_{k,l}-\epsilon_2), \end{aligned} \tag{38}$$

where $\tilde{\boldsymbol{b}}_{-(j,l)}$ means all other components in $\tilde{\boldsymbol{b}}$ remain the same besides $\tilde{b}_{k,j}$ and $\tilde{b}_{k,l}$. Set $\tilde{\boldsymbol{b}} = \tilde{\boldsymbol{b}}'(1)$, if $G(\tilde{\boldsymbol{b}}'(1)) > G(\tilde{\boldsymbol{b}}'(2))$, otherwise set $\tilde{\boldsymbol{b}} = \tilde{\boldsymbol{b}}'(2)$.

Remark 2. *From the above rounding steps (37) and (38), it is clear that $\tilde{\boldsymbol{b}}'$ has smaller number of fractional components than $\tilde{\boldsymbol{b}}$. Since the number of components in $\tilde{\boldsymbol{b}}$ is finite, the rounding steps terminate in a finite number of steps. Also, it is clear that $\tilde{\boldsymbol{b}}'$ satisfies the second and the fourth constraints in (11) and (14a) for $\forall\epsilon \in [-\epsilon_1, \epsilon_2]$ or $\forall\epsilon \in [-\epsilon_2, \epsilon_1]$.*

Now suppose that $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}')$ is the rounded solution. Then following an argument similar to that in [1], we have

Lemma 4.10. *For $k \in \mathcal{K}$, if $\sum_{j=1}^{h(k)} b_{k,j}$ is an integer, then $\sum_{j=1}^{h(k)} b'_{k,j}$ is also an integer; if $\sum_{j=1}^{h(k)} b_{k,j}$ is a fraction, then $\left\lfloor \sum_{j=1}^{h(k)} b_{k,j} \right\rfloor \le \sum_{j=1}^{h(k)} b'_{k,j} \le \left\lfloor \sum_{j=1}^{h(k)} b_{k,j} \right\rfloor + 1$.*

We refer the interested reader to [1] for more details.

Now since the energy constraint is integer, given that $(w_{ca}T + (R_k - 1)\varepsilon_{kT}) \le 1$, Lemma (4.10) implies that $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}')$ satisfies the constraints in (11). Therefore, after the rounding, we obtain a feasible solution obeying the constraints in (11).

Theorem 4.11. *We consider a feasible solution $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ and assume that there are two fractional solutions $\tilde{b}_{k,j}$ and $\tilde{b}_{k,l}$. W.l.o.g., we assume that $\tilde{\boldsymbol{b}}' = (\tilde{\boldsymbol{b}}_{-(j,l)}, \tilde{b}_{k,j}-\epsilon, \tilde{b}_{k,l}+\epsilon)$ following rounding steps (37) and (38), then $G(\cdot)$ is convex in ϵ.*

Proof. Recall that

$$G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}) = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l^k_{i,i+1} \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j=0}^{i} (1-\tilde{b}_{k,j})\right),$$

then

$$G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}', \epsilon) = \sum_{k\in\mathcal{K}} \sum_{i=0}^{h(k)-1} R_k y_k l^k_{i,i+1} \left(1 - \prod_{m=i+1}^{h(k)} \delta_{k,m} \prod_{j'\neq j,l}^{i} (1-\tilde{b}_{k,j'}) \cdot (1-\tilde{b}_{k,j}+\epsilon)(1-\tilde{b}_{k,l}-\epsilon)\right),$$

by the second order condition, it is obvious that $G(\cdot)$ is convex in ϵ. This property is called ϵ-convexity property in [1]. □

Corollary 4.12. *Since $G(\cdot)$ is convex in ϵ, it should achieve its maximum at the endpoint of $[-\epsilon_1, \epsilon_2]$ or $\epsilon \in [-\epsilon_2, \epsilon_1]$. Therefore, following the above rounding steps (37) and (38), we have $G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}') \ge G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$.*

Proof. $G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}') \ge G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ follows directly from the convexity of $G(\cdot)$ in ϵ and the rounding steps in (37) and (38). □

Rounding Scheme: Now for any solution $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$ that satisfies the constraints in (11) and (14a), where $\tilde{\boldsymbol{b}}$ contains fractional terms. There always exists a way to transfer mass between any two fractional variables $\tilde{b}_{k,j}$ and $\tilde{b}_{k,l}$ such that

- (i) at least one of them becomes 0 or 1;
- (ii) the resultant solution $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}')$ is feasible, i.e., $(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}')$ satisfy the constraints in (11) and (14a);
- (iii) the gain satisfies $G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}}') \ge G(\boldsymbol{\delta}, \tilde{\boldsymbol{b}})$.

Then we can obtain an integral solution with the following iterative algorithm:

(1) Given the optimal solution $(\boldsymbol{\tau}^*, \boldsymbol{u}^*)$ to (25) , we first obtain the optimal solution $(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**})$ through the convexity mapping defined in (16).
(2) If there are fractional solutions in $\tilde{\boldsymbol{b}}^{**}$, the number of fractional solutions must be at least two since the capacities are integer. W.l.o.g., consider two fractional solutions $\tilde{b}^{**}_{k,j}$ and $\tilde{b}^{**}_{k,l}$, for $j, l \in \{1, \cdots, h(k)\}$ and $j \neq l$.
(3) Following the above properties (i) (ii) and (iii) to transform at least one of them into 0 or 1 and the resultant gain G is increased.
(4) Repeat steps 2 and 3 until there are no fractional solutions in $\tilde{\boldsymbol{b}}^{**}$.

Denote the resultant solution as $(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**\prime})$ which satisfies the constraints in (8). Note that each step can round at least one fractional solution to an integer one, the above iterative algorithm can terminate at most in $|\mathcal{K}| \times \sum_{k\in\mathcal{K}} |h(k)|$ steps. As each rounding step increases the gain, we have

$$\begin{aligned} G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**\prime}) \ge G(\boldsymbol{\delta}^{**}, \tilde{\boldsymbol{b}}^{**}) &\overset{(a)}{\ge} \left(1-\frac{1}{e}\right) G(\tilde{\boldsymbol{\delta}}^{*}, \tilde{\boldsymbol{b}}^{*}) \\ &\overset{(b)}{\ge} \left(1-\frac{1}{e}\right) G(\boldsymbol{\delta}^{*}, \boldsymbol{b}^{*}), \end{aligned} \tag{39}$$

where (a) holds from Theorem 4.9 and (b) holds from Theorem 4.6. Therefore, we have obtained a $(1-1/e)$-approximation solution to the original optimization problem (8).

5 PERFORMANCE EVALUATION

We evaluate the performance of our proposed algorithm against benchmarks over synthetic data-based network topologies.

5.1 Benchmarks

To compare our proposed solution technique with existing ones, we solve the original non-convex mixed integer non-linear optimization (MINLP) in (7) using conventional online solvers, including

Table 2: Characteristics of the Online Solvers

Solver	Characteristics
Bonmin [5]	A deterministic approach based on Branch-and-Cut method that solves relaxation problem with Interior Point Optimization tool (IPOPT), as well as mixed integer problem with Coin or Branch and Cut (CBC).
NOMAD [21]	A stochastic approach based on Mesh Adaptive Direct Search Algorithm (MADS) that guarantees local optimality. It can be used to solve non-convex MINLP.
GA [12]	A meta-heuristic stochastic approach that can be tuned to solve global optimization problems.

Bonmin [5], NOMAD [21] and Genetic Algorithm (GA) [12], which have all been designed to solve classes of MINLP problems. The characteristics of these solvers are given in Table 2.

Note that GA is a stochastic approach whose performance greatly varies from one simulation run to other. In order to reduce the variance, we run the algorithm 10 times and provide the average, maximum and minimum time along with objective function value obtained using GA. For sake of comparison, we also ran our algorithm 10 times. For our proposed algorithm, we use Algorithm 1 to solve the approximate relaxed convex problem and then use the rounding scheme discussed in Section 4.4 to obtain a feasible solution to the original problem. We compare the performance of our proposed algorithm with these benchmarks with respect to average latency as well as the complexity (measured in units of time).

Table 3: Parameters Used in Simulations

Parameter	Value	Parameter	Value
y_k	100	ε_{vR}	50×10^{-9} J
R_k	1000	ε_{vT}	200×10^{-9} J
w_{ca}	1.88×10^{-6}	ε_{cR}	80×10^{-9} J
T	10s	l	0.6
S_v	120	W	200

5.2 Synthetic Evaluation

5.2.1 Simulation Setting. We consider binary tree networks with 7, 15, 31 and 63 nodes, respectively. We assume that each leaf node generates $y_k = 100$ data items[7], which will be requested $R_k = 1000$ times during a time period $T = 10s$. $S_v = 120$ is the storage capacity of each node. For simplicity, we assume that the latency along each link in the network is identical and take $l = 0.6$. Our simulation parameters are provided in Table 3, which are typical values used in the literature [15, 26, 34]. We implement Bonmin, NOMAD and Algorithm 1 in Matlab using OPTI-Toolbox and Matlab's built-in GA algorithm on a Windows 7 64 bits, 3.40 GHz Intel Core-i7 Processor with 16 GB memory.

5.2.2 Evaluation Results. The performance of these algorithms with respect to the obtained value of the objective function and the time needed to obtain it, are given in Table 4.

On the one hand, we observe that neither Bonmin or NOMAD provide feasible solution with the constraint in (5e). We then further relax this constraint for Bonmin and NOMAD. Hence, the results provided in Table 4 for Bonmin and NOMAD are solved without constraint (5e). Again, we notice that even after relaxing the constraint, Bonmin and NOMAD still exhibit poor performance, i.e. they either provide an infeasible solution or do not converge to a feasible solution. This is mainly due to the hardness of the original non-convex MINLP (7). Hence, it is important to provide an efficient approximation algorithm to solve it.

On the other hand, we observe that both our proposed algorithm and GA provide encouraging results. We run both GA and our algorithm 10 times and report their average as the obtained solutions and run time in Table 4. Tables 5 and 6 provide detailed results for the two algorithms. It is clear that our proposed algorithm significantly outperforms these conventional online solvers both in terms of run time and the obtained objective function value.

In particular, for the 63 node network, GA provides a solution faster than ours. However, GA is not robust and reliable for larger networks. We characterize the robustness of GA, as shown in Table 6, where the maximal (Max.), minimal (Min.) and average values of the objective function are presented as well as the corresponding time to obtain them for 7, 15, 31 and 63 nodes binary tree networks. We notice that for the 63 nodes network, only 4 out of 10 runs converge to a feasible solution using GA. Therefore, GA cannot always guarantee a feasible solution though it may complete in less time. Table 5 provides a detailed overview of our algorithm. The maximal, minimal and average values in terms of time, obtained solution and number of iterations are given. Our master-slave algorithm converges to a solution in small number of iterations.

Also note that our proposed approach always achieves a feasible solution within the $(1-1/e)$ approximation of the optimal solution[8]. Therefore, our proposed algorithm can efficiently solve the problem, i.e., provides a feasible solution in a reasonable time and is robust to network topologies changes.

We also characterize the impact of the number of requests on the caching and compression gain, shown in Figure 2. We observe that as the number of requests increases, the gain increases, as reflected in the objective function (7). Note that the objective function (7) is monotonically increasing in the number of requests R_k for all $k \in \mathcal{K}$ provided that $\boldsymbol{\delta}$ and $\boldsymbol{b}$ are fixed.

Remark 3. *Throughout the evaluations, we notice that the compression ratio at a leaf node is much smaller than the ratio at the root node. For example, in the* 63 *node network, the compression ratio*[9] *at a leaf node is* 0.01 *while it is* 0.37 *at the root node. This captures the tradeoff between the costs of compression, communication and caching in our optimization framework. Similar observations can be made in other networks and hence are omitted here.*

5.2.3 Heterogeneous Networks. In the previous section, we consider binary tree networks under homogeneous settings, i.e., the

[7]Note that this can be equivalently taken as 100 sensors generating data

[8]Note that the original optimization problem (7) is non-convex MINLP, which is NP-hard. Bonmin, NOMAD and GA all claim to solve MINLP with a ϵ-optimal solution. However, GA and NOMAD are stochastic approaches, they cannot guarantee ϵ-global optimality. Hence, we compare our solution with these of Bonmin, NOMAD and GA to verify the approximation ratio.

[9]Defined as the ratio of the volume of the output data to that of the input data at a node. The higher the compression ratio is, the lower is the data compression.

Table 4: Comparison Among Selected Algorithms Using Synthetic Data for Various Network Topologies

Nodes	Proposed		GA		Nomad		Bonmin	
	Obj. Value	Time(s)	Obj. Value	Time(s)	Obj. Value	Time(s)	Obj. Value	Time(s)
7	480000	3.30	479820	273.29	Infeasible	9.45	Infeasible	1.01
15	1440000	6.33	1440000	15.12	1439900	16.18	Infeasible	> 4000
31	3840000	29.28	3839000	3501.10	Non-Convergence	98.90	Non-Convergence	1232.31
63	9599900	538.17	8792100	158.56	Non-Convergence	966.16	Non-Convergence	2.04

Table 5: Detailed Results for Our Proposed Algorithm

Node	Time			Obj. Value			Iterations		
	Max.	Min.	Average	Max.	Min.	Average	Max.	Min.	Average
7	3.5848	3.12	3.30	480000	480000	480000	4	4	4
15	7.23	6.00	6.33	1440000	1440000	1440000	2	2	2
31	30.99	28.57	29.28	3840000	3839900	3840000	2	2	2
63	553.94	531.61	538.17	9600000	9599900	9599900	3	3	3

Table 6: Robustness of GA Algorithm

Node	Time (s)			Objective Value			Convergence (%)
	Max.	Min.	Average	Max.	Min.	Average	
7	369.43	161.76	273.29	479880	479750	479820	100
15	18.15	12.56	15.12	1440000	1440000	1440000	100
31	4446.70	2552.40	3501.10	3839100	3838900	3839000	100
63	413.28	24.41	158.56	9599100	8041500	8792100	40

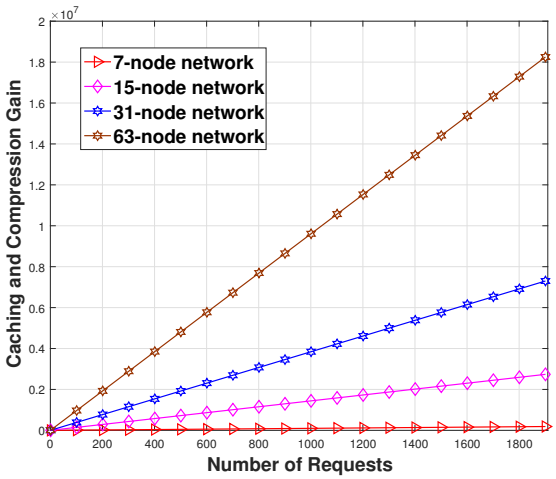

Figure 2: Impact of number of requests on performance.

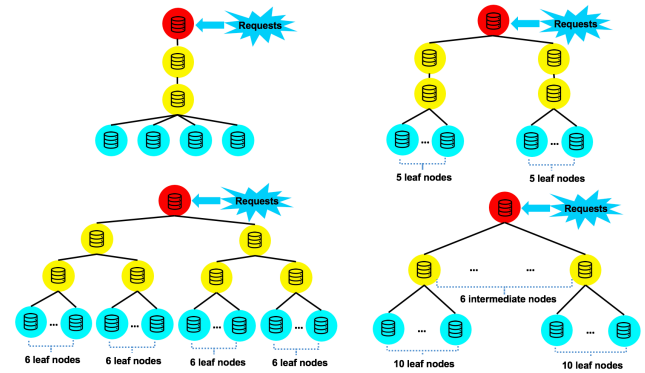

Figure 3: Heterogeneous Tree Networks used in Simulations

value of different parameters are identical for all nodes in the network, as given in Table 3. In this section, we generalize the simulation setting from two perspectives: (i) First, we consider heterogeneous parameter values across the network. For example, for the node cache capacity S_v, we assume that $S_v = 100+rand(1, 20)$, where $rand(i, j)$ assigns a random number between i and j. Similarly, we assign a random number to ε_{vR}, ε_{vT} and ε_{cR} on each node; (ii) Second, instead of considering binary tree, we consider more general network topologies with 7, 15, 31 and 67 nodes, as shown in Figure 3.

The performance of these algorithms with respect to the obtained value of the objective function and the time needed to obtain it, are given in Table 7. Again, we observe that neither Bonmin nor NOMAD can effectively solve the original problem in (7), which shows the hardness of the problem. Hence, it is important to provide an efficient approximation algorithm to solve it.

Similarly, we also observe that both our proposed algorithm and GA provide encouraging results. We run both GA and our algorithm 10 times and report their average as the obtained solutions and run time in Table 7. We also obtain detailed results for both algorithms, where trends similar to Tables 5 and 6 are observed. These are omitted here due to space constraints, and are available in [24]. It is clear that our proposed algorithm significantly outperforms these conventional online solvers both in terms of run time and the obtained objective function value. Furthermore, again we notice that GA cannot always guarantee a feasible solution.

We also characterize the impact of the number of requests on the caching and compression gain. Similar to Figure 2, we observe that as the number of requests increases, the gain increases, hence the plot is omitted, which is available in [24].

Table 7: Comparison Among Selected Algorithms Using Synthetic Data for Various Network Topologies II

Nodes	Proposed		GA		Nomad		Bonmin	
	Obj. Value	Time(s)	Obj. Value	Time(s)	Obj. Value	Time(s)	Obj. Value	Time(s)
7	720000	5.5261	720000	6.14	720000	78.99	Infeasible	45.37
15	1799900	5.87	1800000	20.08	Non-Convergence	37.27	Infeasible	1151.05
31	4319500	33.81	4318700	66.68	Non-Convergence	179.21	Non-Convergence	32293.37
67	7197900	115.17	6531200	399.39	Non-Convergence	1037	Non-Convergence	> 40000

6 CONCLUSION

We considered the problem of optimally compressing and caching data across a communication network, with the goal of minimizing the total latency under an energy constraint. We reformulated this as a problem of maximizing compression and caching gain. This problem is NP-hard. We then proposed an efficient approximation algorithm that can achieve a $(1 - 1/e)$ approximation solution to the optimum in strongly polynomial time. Finally, we evaluated the performance of our proposed algorithm through extensive synthetic simulations, and made a comparison with benchmarks. We observed that our proposed algorithm can achieve near-optimal solution and outperform the benchmarks.

ACKNOWLEDGMENTS

This work was supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon. Faheem Zafari also acknowledges the financial support by EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1), and Department of Electrical and Electronics Engineering, Imperial College London.

REFERENCES

[1] Alexander A Ageev and Maxim I Sviridenko. 2004. Pipage Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee. *Journal of Combinatorial Optimization* 8, 3 (2004), 307–328.

[2] David Applegate, Aaron Archer, Vijay Gopalakrishnan, Seungjoon Lee, and KK Ramakrishnan. 2016. Optimal Content Placement for a Large-Scale VoD System. *IEEE/ACM Transactions on Networking* 24, 4 (2016), 2114–2127.

[3] Ivan Baev, Rajmohan Rajaraman, and Chaitanya Swamy. 2008. Approximation Algorithms for Data Placement Problems. *SIAM J. Comput.* 38, 4 (2008), 1411–1429.

[4] Kenneth C Barr and Krste Asanović. 2006. Energy-aware Lossless Data Compression. *ACM Transactions on Computer Systems* (2006).

[5] Pierre Bonami et al. 2008. An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs. *Disc. Opt.* 5, 2 (2008), 186–204.

[6] Sem Borst, Varun Gupta, and Anwar Walid. 2010. Distributed Caching Algorithms for Content Distribution Networks. In *Proc. IEEE INFOCOM*. 1–9.

[7] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[8] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2007. Maximizing a Submodular Set Function Subject to a Matroid Constraint. In *IPCO*, Vol. 7. Springer, 182–196.

[9] Nakjung Choi, Kyle Guan, Daniel C Kilper, and Gary Atkinson. 2012. In-network Caching Effect on Optimal Energy Consumption in Content-Centric Networking. In *Proc. IEEE ICC*.

[10] Edith Cohen and Scott Shenker. 2002. Replication Strategies in Unstructured Peer-to-Peer Networks. In *ACM SIGCOMM CCR*, Vol. 32. 177–190.

[11] G Cornnejols, M Fisher, and G Nemhauser. 1977. Location of Bank Accounts of Optimize Float: An Analytic Study of Exact and Approximate Algorithm. *Management Science* 23 (1977), 789–810.

[12] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[13] Mostafa Dehghan, Anand Seetharam, Bo Jiang, Ting He, Theodoros Salonidis, Jim Kurose, Don Towsley, and Ramesh Sitaraman. 2015. On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks. In *Proc. IEEE INFOCOM*. 936–944.

[14] Michel X. Goemans and David P. Williamson. 1994. NEW 3/4-APPROXIMATION ALGORITHMS FOR THE MAXIMUM SATISFIABILITY PROBLEM. *SIAM Journal on Discrete Mathematics* 7, 4 (1994).

[15] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. 2000. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *System sciences*.

[16] Stratis Ioannidis and Edmund Yeh. 2016. Adaptive Caching Networks with Optimality Guarantees. In *Proc. ACM SIGMETRICS*. 113–124.

[17] Stratis Ioannidis and Edmund Yeh. 2017. Jointly Optimal Routing and Caching for Arbitrary Network Topologies. *arXiv preprint arXiv:1708.05999* (2017).

[18] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. 2009. Networking Named Content. In *Proc. ACM CoNEXT*. 1–12.

[19] Anant Singh Jain and Sheik Meeran. 1999. Deterministic Job-Shop Scheduling: Past, Present and Future. *European journal of operational research* 113, 2 (1999).

[20] Andreas Krause and Daniel Golovin. 2014. Submodular Function Maximization. http://www.cs.cmu.edu/afs/.cs.cmu.edu/Web/People/dgolovin/papers/submodular_survey12.pdf. (2014).

[21] Sébastien Le Digabel. 2011. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm. *ACM TOMS* 37, 4 (2011), 44.

[22] Jian Li, Truong Khoa Phan, Wei Koong Chai, Daphne Tuncer, George Pavlou, David Griffin, and Miguel Rio. 2018. DR-Cache: Distributed Resilient Caching with Latency Guarantees. In *Proc. IEEE INFOCOM*.

[23] Jian Li, Srinivas Shakkottai, John C.S. Lui, and Vijay Subramanian. 2017. Accurate Learning or Fast Mixing? Dynamic Adaptability of Caching Algorithms. *arXiv preprint arXiv:1701.02214* (2017).

[24] Jian Li, Faheem Zafari, Don Towsley, Kin K. Leung, and Aanathram Swami. 2018. Joint Data Compression and Caching: Approaching Optimality with Guarantees. *Arxiv preprint arXiv:1801.02099* (2018).

[25] A. Manjeshwar and D. P. Agrawal. 2001. TEEN: a Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *IPDPS*.

[26] Sepideh Nazemi, Kin K Leung, and Ananthram Swami. 2016. QoI-aware Tradeoff Between Communication and Computation in Wireless Ad-hoc Networks. In *Proc. IEEE PIMRC*.

[27] Nitish K. Panigrahy, Jian Li, and Don Towsley. 2017. Hit Rate vs. Hit Probability Based Cache Utility Maximization. In *Proc. ACM MAMA*.

[28] Nitish K. Panigrahy, Jian Li, and Don Towsley. 2017. Network Cache Design under Stationary Requests: Challenges, Algorithms and Experiments. *Arxiv preprint arXiv:1712.07307* (2017).

[29] Nitish K. Panigrahy, Jian Li, Faheem Zafari, Don Towsley, and Paul Yu. 2017. What, When and Where to Cache: A Unified Optimization Approach. *Arxiv preprint arXiv:1711.03941* (2017).

[30] Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Vol. 24. Springer Science & Business Media.

[31] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. 2013. Femtocaching: Wireless Content Delivery through Distributed Caching Helpers. *IEEE Transactions on Information Theory* 59, 12 (2013), 8402–8413.

[32] Dominic JA Welsh. 2010. *Matroid Theory*. Courier Corporation.

[33] Mao Ye, Chengfa Li, Guihai Chen, and Jie Wu. 2005. EECS: an Energy Efficient Clustering Scheme in Wireless Sensor Networks. In *IEEE IPCCC*.

[34] Wei Ye, John Heidemann, and Deborah Estrin. 2002. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *IEEE INFOCOM*.

[35] Yang Yu, Bhaskar Krishnamachari, and Viktor K Prasanna. 2008. Data Gathering with Tunable Compression in Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems* 19, 2 (2008), 276–287.

[36] Faheem Zafari, Jian Li, Kin K. Leung, Don Towsley, and Aanathram Swami. 2017. Optimal Energy Tradeoff among Communication, Computation and Caching with QoI-Guarantee. *Arxiv preprint arXiv:1712.03565* (2017).