Journal of Parallel and Distributed Computing 157 (2021) 168–178

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Joint optimization of cache placement and request routing in unreliable networks

Youmei Song [a], Tianyu Wo [a,*], Renyu Yang [b,*], Qi Shen [a], Jie Xu [b]

[a] *National Key Laboratory of Software Development Environment, Beihang University, China*
[b] *School of Computing, University of Leeds, UK*

ARTICLE INFO

Article history:
Received 11 June 2020
Received in revised form 21 March 2021
Accepted 19 June 2021
Available online 8 July 2021

Keywords:
Edge caching
Content placement
Request scheduling
Network unreliability
Optimization

ABSTRACT

Edge caching is a prevailing media delivery technology where data is hosted at the edge nodes with computing and storage capability in close proximity to users, in order to expand the backhaul network capacity and enhance users' quality of experience (QoE). The existing work in this area often neglects the fact that large-scale distributed cache networks are not particularly reliable and many edge nodes are prone to failure. In this paper we investigate and develop a novel, cooperative caching mechanism for content placement and request routing. We aim to minimize the content access delay and achieve the optimization in polynomial time, taking into account failures in an unreliable network environment with limited edge storage and bandwidth. We introduce two optimization algorithms: 1) a primal-dual algorithm that is based on the Lagrangian dual decomposition and subgradient method, and 2) a greedy-based approximation algorithm with a proven approximation ratio. Numerical results show that the proposed algorithms outperform other comparative approaches in synthetic and real network environments, and the approximation algorithm is particularly suitable for networking scenarios with sparse node connectivity and resources in short supply.

1. Introduction

Due to the proliferation of social-media service providers (Facebook, YouTube, or Twitter), content delivery is experiencing unprecedented growth worldwide. According to a study by Cisco in 2017 [7], the global mobile traffic will reach 77 exabytes per month by 2022. Undoubtedly, the *data tsunami* will fill up the holistic network capacity and devastate the users' quality of experience (QoE). Edge caching has been becoming the main-streaming technology to store content closer to end users via the use of caching servers. A downward trend in the storage space price [1] also substantially advances the edge caching and expands the holistic network capacity. Technically it is achieved by placing contents in edge nodes, e.g., small base stations (SBSs) with computing and storage capability in close proximity to users [30,8,42]. Numerous content requests can be directed to these intermediate SBSs instead of remote servers [37,49], thereby significantly reducing the backhaul traffic and transmission delays.

While promising, the insufficient cache capacity within edge nodes impedes the further enhancement of caching performance and reduction of service latency [44,5]. In reality, when a large number of different data flows are generated, massive contents tend to be missing in cache nodes with a low hit probability. The effective bandwidth for content delivery is the data rate over the backhaul network. To make the best use of existing cached contents, the missed requests at a given edge node are routed to other edge nodes, by leveraging multi-hops delivery path [27]. Meanwhile, cooperative caching mechanisms are proposed to manage the increasing number of SBSs [11] as a cache pool, thereby increasing the chances of obtaining contents from the edge [23]. Hence, at the core of the cooperative caching management is the joint mechanism for content *cache placement* and *request routing*; content placement decides how to locate the contents over the caching nodes, either servers or edge nodes, while request routing determines how to efficiently route the content request to the most suitable nodes. Although a plethora of algorithms [21,12,20] aim to improve the caching performance, they fail to take into account the unreliable networks, making it difficult to apply them into real-world intricate network conditions.

Essentially, network failures become the norm rather than the exception within large-scale networking systems [10,25,14,45] particularly in geo-distributed multiple joint cloud environments [39,

[*] Corresponding authors.
E-mail addresses: songym@act.buaa.edu.cn (Y. Song), woty@act.buaa.edu.cn (T. Wo), R.Yang1@leeds.ac.uk (R. Yang), shenqi@act.buaa.edu.cn (Q. Shen), J.Xu@leeds.ac.uk (J. Xu).

https://doi.org/10.1016/j.jpdc.2021.06.006
0743-7315/© 2021 Elsevier Inc. All rights reserved.

43]. Providers thus face pressure to provision uninterrupted reliable services while reducing operational costs due to significant software and hardware failures [17,2,46]. The distributed cache networks where many small servers deployed at edge nodes are more prone to failure caused by power supply shutdowns, hardware failures [32,13]. Due to further indexing and transmission distance, network failure events may lead to the failure of cooperative caching to effectively support delay-sensitive services [3,16]. Besides, a high backhaul fetching cost might still be incurred even some storage spaces have been allocated to the contents. Hence, it is imperative to devise an effective caching mechanism for data placement and request routing, particularly considering the manifestation of network unreliability, so as to avoid repeated content retrieval and reduce the delivery cost.

To minimize the delay of content access, this paper presents a novel mechanism for jointly addressing the data placement and request routing problem in unreliable edge computing network environments with limited storage and bandwidth capacity. We formulate the problem as an integer linear programming (ILP) under constraints and prove it generally NP-hard (§3). We provide two distinct algorithmic schemes, adaptive to different scenarios with different resource constraints: (i) we firstly develop a primal-dual algorithm by using the Lagrangian relaxation and hierarchical primal-dual decomposition method so that the problem can be decomposed into multiple smaller sub-problems. Due to the objective function is not differentiable over the whole feasible region, the subgradient method is adopted (§4). (ii) Since the objective function is proved to be monotone submodular subject to uniform matroid constraints, we then advocate a greedy algorithm to achieve a provably better approximation ratio. The primal-dual algorithm achieves higher caching benefits but higher computing complexity while the greedy algorithm moderates the computing complexity through compromising caching efficiency (§5). The work is evaluated under both real network and synthetic network environment; the result show that our proposed algorithms can yield better caching performance. The content retrieval latency is reduced by up to 22%. The greedy algorithm is more suitable for network scenarios with strict resource constraints compared to the primal-dual algorithm. Particularly, this paper makes the following contributions:

- It identifies and investigates the data caching and request routing problem for unreliable and dense caching networks with limited bandwidth and storage capacity.
- It advocates two-fold algorithmic solutions, including a primal-dual algorithm combining Lagrangian dual decomposition method with sub-gradient method, and a greedy caching algorithm with lower computation complexity but a provable approximation ratio guaranteed.
- It outperforms other comparative approaches in terms of the caching performance and content retrieval latency in practice and synthetic networks. The greedy algorithm yields almost the same performance as the primal-dual algorithm in hit rate when cache resources are strongly limited.

The paper is organized as follows: §2 discusses the related work. §3 outlines solution overview and conducts the complexity analysis. The Primal-dual algorithm and the greedy algorithm are detailed in §4 and §5, respectively. Experimental results are presented in §6 before we conclude the paper in §7.

2. Related work

Edge caching, an efficient way to improve users' QoE, is broadly exploited and advanced by both academia and industry. Content placement [38,26] and request routing [35,4] are two critical mechanisms for enhancing users' QoE, particularly in cache-enabled networks:

Content placement and request routing. [31] proposes a cache-aware and social-based routing scheme for Named Data Networking (NDN) with Quality of Service (QoS) guaranteed. The social relationship among caching nodes is considered in the event of forward. [27] presents a cache-aware routing optimization mechanism for maximizing the offloading traffic over software-defined networking (SDN)-enabled 5G networks. The proposed system can route the content to near-optimal caching positions and reduce the involved computational complexity. [40] focuses on a multi-path routing problem to maximize the total bandwidth of edge-disjoint paths. To solve the problem, they designed a heuristic edge delivery algorithm based on global optimization. [47] introduces edge computing into vehicle networks. Given the interconnection among edge vehicle networks, they propose a vehicle-adaptive caching scheme that guarantees cache diversity and improves the packet delivery rate. [48] presents an information-centric networking architecture where different content caching and request forwarding schemes are adopted. [34] proposes a distributed caching method where the small base stations store contents to assist macro base stations to serve user demands. Such a distributed caching problem is proved NP-hard and solved by an approximation algorithm with a constant factor. [29] proposes an optimized content placement solution in hierarchical networks. The requests will reach the content server if none of the caches on the cache path have stored the content replicas. The NP-hard problem is solved in polynomial time when caches are installed only on a single hierarchy path. [20] introduces a collaborative caching scheme, exploiting the traffic diversity depending upon whether the content is coded. a fully polynomial-time approximation algorithm can significantly reduce the operational cost, i.e., delay, compared against non-collaborative approaches. However, the aforementioned algorithms only focus on the optimization of content placement, ignoring the negative impact of the transfer path on the cache performance.

Joint optimization. Apart from the individual optimization, the procedure of content placement and request routing should be devised hand-in-hand [8,18,41]. Typically, the outcome of the content placement policy will be fed into the request routing policy design; meanwhile, the routing result also has a direct impact on the popularity and influence the effectiveness of the original placement. [19] exploits cooperative content caching and one-hop delivery policies for the heterogeneous cellular networks where user devices and SBSs are leveraged as cache nodes. The content delivery problem is formulated as an unbalanced assignment problem and solved by the Hungarian algorithm. [9] considers the in-network caching and routing pathways to remote servers within a reliable heterogeneous network environment. The problem is proved NP-complete and solved by a greedy-based algorithm with guaranteed performance. However, the main limitation is such approaches overlook the bandwidth requirements on the delivery paths, which hinder a wider applicability within bandwidth-scarce environments. Additionally, [32] aims to cache replications of the same data in multiple caches to optimize the LRU caching in an unreliable network environment. However, it only considers single-hop routing in its distributed caching system, and thus cannot be directly applied into the path routing with multi-hops. [6] employs a modularity-based information caching approach by leveraging information reachability to achieve reliable communication of information under adverse network conditions. [22] maximizes the expected caching gain through a distributed resilient caching algorithm (DR-Cache) considering network failures. However, the bandwidth constraints within different routing links are not well discussed.

This work, as a departure from them, considers the uncertainties including node unreliability and link bandwidth constraints.

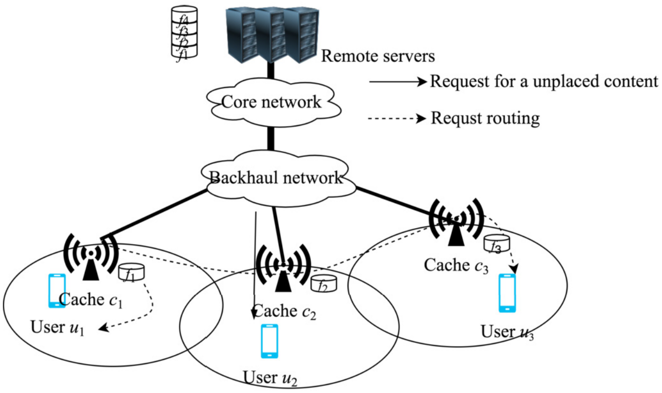


Fig. 1. Illustration of the unreliable network environment consisting of remote servers and edge nodes.

Table 1
Key notations.

$\mathcal{N}$	Set of N SBSs
$\mathcal{U}$	Set of U users
$\mathcal{F}$	Set of F contents
C	Cache capacity
B	Bandwidth capacity
$\lambda_{u,f}$	Probability that user u requests content f
s_n	Stability score of SBS n
$d_{i,j}$	Transmission edge delay between SBS i and j
$\mathcal{Q}_{u,f}$	User u obtain content f through a set of L cached paths
d_0	Content access delay without cache
D	The total expected delay for all requests
$R_{\mathbf{x},\mathbf{q}}$	Caching gain
$q^l_{u,f}$	User u obtain content f through cache path l (1) or not(0)
$x_{i,f}$	Content f is placed at cache-node i

We devise two distinct caching solutions to balance the computation complexity and caching effectiveness and satisfy the diverse requirements in different scenarios.

3. Our approach

3.1. Overview

A motivating example. Fig. 1 depicts a motivating example where an unreliable edge network manifests, and each node simultaneously underpins both caching and forwarding functionalities, with a given failure probability. A request routed along a path will fail if any of the intermediate nodes on the path crashes. We assume that all contents have the same size, normalized to 1. Remote servers store all the contents $f_1 - f_4$, while each cache node only stores one content specified by the storage capacity constraints. For example, given a path with three nodes (c_1, c_2, c_3) and suppose node c_3 caches the content f_4. Since both remoter server and edge cache c_3 store the content f_3 requested by the user u_1, the dispatching algorithm will prioritize to direct the user u_1's request to cache c_3. However, if the propagation is interrupted due to a failure of c_2, the request will directly fail since c_3 is unreachable. In this case, a system without node reliability considered has to perform a further operation of index to ensure the request be directed to remote servers, which is extremely cost-ineffective. Obviously, a better way is to route the demands to the remote server directly instead of c_3. Therefore, it is imperative for content delivery vendors to consider the node stability information when making a decision of content caching in the SBSs.

Problem scope and assumptions. We assume a general network architecture with $\mathcal{N}$, a set of SBSs that can provision internet access for $\mathcal{U}$ a set of users. Each SBS has a finite cache capacity, denoted by C. Users generate requests for a set of $\mathcal{F} = \{f_1, f_2, ..., f_F\}$ of F files with unit size independently at random, according to a popularity distribution. We assume each request arrives independently following Poisson process with the average rate λ_u and the popularity of content follows Zipf distribution with a shape parameter α. These assumptions are similar to those used in related work [37,22]. Then the probability that user u requests content f is $\lambda_{u,f} = \lambda_u \cdot p_f$, where p_f is given by $p_i = (1/f^\alpha)/\sum_{i=1}^{F}(1/i^\alpha)$. Throughout this paper, we will use the terms content and file interchangeably.

Note that routing path between user and SBS is referred to as cached path while routing path between user and remote server is referred to as uncached path. User u can obtain content f through a set of cached paths, which are denoted by $\mathcal{Q}_{u,f} = \{q^1_{u,f}, ..., q^L_{u,f}\}$. $1 - \sum_{l \in \mathcal{Q}_{u,f}} q^l_{u,f}$ is denoted as request from user u for file f is directed through uncached path. We use probability s_n to capture the resilience of each cache node, which is a dependent variable of every transmission edge delay $d_{i,j}(s_j)$. $d(\cdot)$ is continuous and decreasing convex function. As shown in [14,13], there are several factors affecting failure probability of a node. However, deriving an exact model to predict the failure probability beyond the scope of this work.

We assume there is a Caching Management Entity(CME) in the network, which acts as a centralized controller. Information, such as the network topology, the stability of each cache node, the users' locations et al., is reported to the CME to make caching and routing decisions. Table 1 summaries the notations used in the rest of the paper.

3.2. Problem formulation

Basic definition. In our model, we first introduce the content placement variable $x_{i,f}$ where

$$x_{i,f} = \begin{cases} 1 & \text{if cache node } i \text{ stores file } f \text{ ,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The content placement decision matrix is $\mathbf{x} = \{x_{i,f} | i \in \mathcal{N}, f \in \mathcal{F}\}$. Then considering the storage capacity constraint on each node, we have

$$\sum_{f=1}^{F} x_{i,f} = C_i, \forall i \in \mathcal{N} \tag{2}$$

When a file is placed on a cache node, the caching decision would not change in the event of a cache hit or miss. The argument in [24] and [9] shows that static caching can achieve minimum expected delay under a fixed routing policy. In the case, given the caching policy, we denote the request routing decision by $q^l_{u,f}$ where

$$q^l_{u,f} = \begin{cases} 1 & \text{if path } l \text{ transmits } f \text{ to } u, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

We say that a request is well routed if there is a set of paths $\mathcal{Q}_{u,f}$, which contain requested content in the path destination. Beyond that, the bandwidth capacity constraint on each path is also an important factor for fulfilling users' requests. Each request only has one communication link to cache node or remote server. We set the allocated bandwidth to each link as one unite. Then we use B_l, the certain bandwidth capacity of each link, to denote the largest number of content requests that path l can fulfill at the same time.

Content access delay. We consider the case where the request by user u for file f is routed through a set of cached paths $\mathcal{Q}_{u,f}$, the corresponding content access delay is denoted as

$$D_{u,f} = \lambda_{u,f}\Big(\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f} \sum_{k=1}^{|l|-1} d_{k,k+1}\Big). \tag{4}$$

Then, the total expected delay for all request is defined as

$$\begin{aligned} D = &\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f}\Big(\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f} \sum_{k=1}^{|l|-1} d_{k,k+1} \\ &+ (1-\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f})d_0\Big) \end{aligned} \tag{5}$$

and we consider a case where there is no cache node assistance. In the case, it takes d_0 for each request to access the content. Then we have the caching gain denoted by

$$\begin{aligned} R(\mathbf{x},\mathbf{p}) = &\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f}\Big(d_0 - \big(\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f} \sum_{k=1}^{|l|-1} d_{k,k+1} \\ &+ (1-\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f})d_0\big)\Big) \end{aligned} \tag{6}$$

The objective function can be further expressed as:

$$R(\mathbf{x},\mathbf{q}) = \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f} \sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f}\Big(d_0 - \sum_{k=1}^{|l|-1} d_{k,k+1}\Big) \tag{7}$$

Optimization goal. The joint effort to cache content replicas and route requests aims to minimize the sum content access delay, or equivalently, maximizes the caching gain. It can be formulated as follows:

$$\begin{aligned} max\ R(\mathbf{x},\mathbf{q}) = &\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f}\cdot \\ &\sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f}\Big(d_0 - \sum_{k=1}^{|l|-1} d_{k,k+1}\Big) && (8a)\\ s.t.\ & q^{l}_{u,f} \le x_{i,f}, \forall i\in\mathcal{N}_l, \forall f\in\mathcal{F}, \forall l\in\mathcal{Q}_{u,f} && (8b)\\ & \sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f} \le 1, \forall i\in\mathcal{N}, \forall f\in\mathcal{F} && (8c)\\ & \sum_{f=1}^{F} x_{i,f} \le C_i, \forall i\in\mathcal{N} && (8d)\\ & \sum_{f=1}^{F}\sum_{u=1}^{U} \lambda_{u,f} q^{l}_{u,f} \le B_l, \forall l\in\mathcal{Q}_{u,f} && (8e)\\ & x_{i,f} = \{0,1\}, \forall i\in\mathcal{N}, \forall f\in\mathcal{F} && (8f)\\ & q^{l}_{u,f} = \{0,1\}, \forall i\in\mathcal{N}, \forall f\in\mathcal{F}, \forall l\in\mathcal{Q}_{u,f} && (8g) \end{aligned}$$

Constraint (8b) ensures that request from user u is only routed through the path l when the destination of the path hosts the required content f, since content f is placed to SBS i only when $x_{i,f} = 1$. Constraint (8c) indicates that users access content over at most one path at each time. Constraint (8d) guarantees that at any time, allocated resources at a SBS do not exceed its storage capacity. Constraint (8e) guarantees that at any time, allocated resources at each path do not exceed its bandwidth capacity.

Model generalization and applicability. While this paper focuses on addressing a joint optimization problem to minimize the overall request latency with the caching capacity and bandwidth constraints, the formulated problem and solutions presented in the following sections are readily applicable to other real-world problem-solving in such areas as wireless networks, SDN, and even in latency-sensitive job scheduling and streaming management in cloud and edge computing. Without loss of generality, the generic model can be depicted as a problem where the objective is to minimize the system cost or maximize the system gain over two variables. Meanwhile, the constraints, e.g., (8a–8e) can be easily extended and customized to underpin scenario-specific requirements. For example, in a SDN network, the placement of network functions and routing selection are important factors for network load balancing. The variables in the objective could be redefined and tweaked to involve the placement and routing of network function chains. Constraints such as CPU and bandwidth capacity or other co-location affinity/anti-affinity can be supplemented in the similar form of (8d), (8e).

3.3. Complexity analysis

In this section, we character the complexity based on storage and bandwidth capacity constraints.

Storage constraints only. Consider a special case where we only have the storage constraints; in other words, bandwidth is sufficient to accommodate all possible data transfer. Then optimization problem will be transformed as:

$$\begin{aligned} max\ R(\mathbf{x},\mathbf{q}) = &\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f} \sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f}\Big(d_0 - \\ &\sum_{k=1}^{|l|-1} d_{k,k+1}\Big) && (9a)\\ s.t.\ &(8b)-(8d),(8f),(8g) && (9b) \end{aligned}$$

We deduce the Problem (9) into the 2-disjoint set cover(2*DSC*) problem. Given a bipartite graph $\mathcal{G} = \{\mathcal{A},\mathcal{B},E\}$ with edges E connecting the set of two disjoint vertex sets A and B. Define subset $\mathcal{N}_b \subseteq \mathcal{A}$ the of neighbors of node b. Then we have $\bigcup_{b\in\mathcal{B}}\mathcal{N}_b = \mathcal{A}$ clearly. 2*DSC* determines whether there exist two disjoint sets $\mathcal{B}_1,\mathcal{B}_2 \subset \mathcal{B}$, such that $|\mathcal{B}_1|+|\mathcal{B}_2| = |\mathcal{B}|$ and $\mathcal{A} = \bigcup_{b\in\mathcal{B}_1}\mathcal{N}_b = \bigcup_{b\in\mathcal{B}_2}\mathcal{N}_b$.

The process of reduction from 2*DSC* to our problem is give as follows. The SBSs set and user set are set to $\mathcal{A}$ and $\mathcal{B}$, respectively. The storage capacity of each node is set to 1. File library is set to $\mathcal{F} = \{f_1, f_2\}$. We set the requests for all users one unit, i.e., $\lambda_{b,f} = 1, \forall b\in\mathcal{B}, f\in\{f_1,f_2\}$. Since we set that there is only one delivery path from each user to each cache node, $q^{l}_{u,f}$ is rewritten as $q^{i,j}_{u,f}$ denoting node j can satisfy the request generated by user u submitted to node i. We set $d_0 = d^{i,j}_{u,f} = 1$ if only if $e_{i,j}\in E$, otherwise $d^{i,f}_{u,f} = 0$.

If there exists a solution to *2DSC*, then we can cache file f_1 at all SBSs in set B_1 and f_2 at remaining SBSs since the storage capacity of each cache node is set as 1. In this case, the solution can serve all the requests. On the other side, if there exists a solution to our problem, B_1 caching f_1 and B_2 caching f_2 is a feasible solution to cover all requests. Thus, there exist 2 disjoint set covers.

Bandwidth constraints only. Consider a special case where we only have the bandwidth constraints and each node stored all the requested contents. The optimization problem will be transformed to:

$$\begin{aligned} max\ R(\mathbf{x},\mathbf{q}) = &\sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} \lambda_{u,f} \sum_{l\in\mathcal{Q}_{u,f}} q^{l}_{u,f}\Big(d_0 - \\ &\sum_{k=1}^{|l|-1} d_{k,k+1}\Big) && (10a)\\ s.t.\ &(8\text{c, e, g}) && (10b) \end{aligned}$$

This special case can be interpreted as the classical knapsack problem where the bandwidth capacity equal to the knapsack, and one content represents the item. The content size is the item weight and the caching gain is the value of the item.

Hence, proving NP-hardness for these special cases shows that the problem is NP-hard in the general case.

4. Primal-dual algorithm

Since the problem is an integer linear program, it can be solved using standard convex optimization techniques. We decide to investigate the use of Lagrangian relaxation and decomposition method. Lagrangian relaxation method is generally an efficient bounding technique and decomposition can decompose the original large problem into distributively solvable subproblems [28]. Note that the decomposition action can not break the optimality of the solution. Consider the problem architecture, there are two variables, content placement decision, and request routing decision, triggering two possible Lagrangian relaxations: $R_\eta(\mathbf{x},\mathbf{q})$, resulting from the dualization of constraint (8f) by using dual variable η, and $R_\gamma(\mathbf{x},\mathbf{q})$, resulting from the dualization of constraint (8e) by using dual variable γ. We firstly incorporate the restriction (8b) into objective (8a) by variable $\boldsymbol{\eta}$. Then the problem (8) is rewritten as:

$$
\begin{aligned}
&min\ R(\mathbf{x},\mathbf{q})_\eta = \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\lambda_{u,f}\sum_{l\in\mathcal{Q}_{u,f}} q^l_{u,f}(d_0 - \\
&\quad \sum_{k=1}^{|l|-1} d_{k,k+1}) + \sum_{u\in\mathcal{U}}\sum_{\in\mathcal{F}}\lambda_{u,f}\sum_{l\in\mathcal{Q}_{u,f}}\eta^l_{u,f}\cdot \\
&\quad (x_{j,f} - q^l_{u,f}) && (11a)\\
&s.t.\ \eta^l_{u,f} \geq 0, \forall i\in\mathcal{N}, \forall f\in\mathcal{F}, \forall l\in\mathcal{Q}_{u,f} && (11b)\\
&\quad (8c)-(8g) && (11c)
\end{aligned}
$$

Apparently $R(\mathbf{x},\mathbf{q})_\eta$ is the upper bound of $R(\mathbf{x},\mathbf{q})$ when $\eta \geq 0$. According to [28], the problem (11) is called master problem. After the first Lagrangian relaxation, we can separate decompose the original problem into subproblem P_1 only involving content placement and subproblem P_2 only involving request routing, and solve them independently.

The subproblem P_1 can be formulated as:

$$
\begin{aligned}
&max \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\sum_{l\in\mathcal{Q}_{u,f}}\lambda_{u,f}\eta^l_{u,f}x_{j,f} && (12a)\\
&s.t.\ (8d),(8f) && (12b)
\end{aligned}
$$

The subproblem P_1 can be categorized as a knapsack problem. We can obtain the solution of caching decision variables by selecting the storage capacity most popular contents. Specifically, each cache node i orders the popularity of files in decreasing order and makes the x_i's corresponding to capacity largest coefficients to 1, all others are 0. In relation to routing decision, the problem P_2 is not readily available until we take the second Lagrangian relaxation. P_2 is expressed as:

$$
\begin{aligned}
&max \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\sum_{l\in\mathbf{Q}_{u,f}}\lambda_{u,f}q^l_{u,f}(d_0 - \sum_{k=1}^{|l|-1} d_{k,k+1} \\
&\quad -\eta^l_{u,f}) && (13a)\\
&s.t.\ (8c),(8e),(8g) && (13b)
\end{aligned}
$$

Then we incorporate constraint (8e) into the objective function by associating the Lagrangian multiplier $\boldsymbol{\gamma}$. The problem P_2 is rewritten as

$$
\begin{aligned}
&min\ max \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\sum_{l\in\mathcal{Q}_{u,f}}\lambda_{u,f}q^l_{u,f}(d_0 - \\
&\quad \sum_{k=1}^{|l|-1} d_{k,k+1} - \eta^l_{u,f} - \gamma_l) + \sum_{l\in\mathcal{Q}_{u,f}}\gamma_l B_l && (14a)\\
&s.t.\ (8c),(8g) && (14b)
\end{aligned}
$$

To obtain the request routing decision, we can simply set

$$
q'^l_{u,f} = \begin{cases} 1 & \text{if } w^l_{u,f} \text{ is max,} \\ 0 & \text{otherwise,} \end{cases} \tag{15}
$$

where, $w^l_{u,f}$ is the coefficient of each routing decision variable $q^l_{u,f}$

$$
w^l_{u,f} = \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\sum_{l\in\mathcal{Q}_{u,f}}\lambda_{u,f}(d_0 - \sum_{k=1}^{|l|-1} d_{k,k+1} - \eta^l_{u,f} - \gamma_l) \tag{16a}
$$

If we want to obtain content placement decisions and request routing decisions, the value of two Lagrangian multipliers, $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, is required. We employ the subgradient method to update them. Since the two Lagrangian multipliers are solved in the same way, let's take the $\boldsymbol{\gamma}$ as an example and explain it in detail. To simplify the expression, we replace $q^l_{u,f}$ as $q^i_{u,f}$, where i is nodes corresponding to path $q^l_{u,f}$. In each updating iteration t, $\lambda^l_{u,f}$ is updated by the rule

$$
\gamma_i(t+1) = max\{0, \lambda^i_{u,f}(t) - \theta_i(t)g_i(t)\} \tag{17}
$$

where $g_i(t) = B_i - \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}}\lambda_{u,f}q^l_{u,f}(t)$ is the subgradient direction and $\theta(t) = \beta(t)(ub(t) - lb(t))/(\|\mathbf{g}(t)\|^2)$ is step size at $\gamma_i(t)$, respectively. $\beta \in [0,2]$ is a scalar of step size, $ub(t), lb(t)$ is the upper bound and lower bound at iteration t, respectively. β always starts with 2. Half the value of β if the upper bound stays the same after some number of iterations. The process of iteration continues until one of the conditions of termination criterion TC_γ happens. TC_γ is listed as follows:

1. $t > 1000$;
2. $\beta \leq 0.005$;
3. $(ub - lb)/ub < 0.01$;
4. ub unchanged by 20 consecutive iterations.

In this case, we can obtain the request routing decisions when there are infinite gradient iterations. The problem P_2 has been solved. The summary procedure of the solution is given in Algorithm 1.

After the execution of Algorithm 1, we have obtain the value of $R(\mathbf{x},\mathbf{q})$ given $\boldsymbol{\eta}$. For the first master problem, we repeat the above progress to update $\boldsymbol{\eta}$ until dual variable η_t converge to the dual optimal value η^*. Then optimal value of $\mathbf{x}^*$ can be obtained since the problem is convex. The summary of the primal-dual algorithm for the joint problem is given in Algorithm 2.

In Algorithm 2, there are t iterations to converge to the dual primal value. Each iteration involves the sum of computation complexity of the problem P_1 and P_2. Solving the problem P_1 takes $\mathcal{O}(N\cdot F\cdot U)$ time since P_1 can be decomposed into N one-dimensional knapsack problems. Solving problem P_2 takes $\mathcal{O}(N\cdot F\cdot U\cdot t)$ time since there are t iterations to update Lagrangian multiplier $\boldsymbol{\gamma}$. Hence the total running time is $\mathcal{O}((N\cdot F\cdot U + N\cdot F\cdot U\cdot t)\cdot t)$.

Algorithm 1: Pseudocode for Algorithm 1.

```
Input: $\lambda_{u,f}, \eta^l_{u,f}, d_0, B_i, x_{i,f}*, d_{k,k+1}$.
Output: $q^l_{u,f}*$.
Initialization: $ub = +\infty$, lower bound $lb = -\infty$, $t = 0$, and $\gamma_i(0)$ to some
  positive value.
while not the termination criterion do
    Compute $q^l_{u,f}(t)$ by (14) and obtain the objective function value $v_\gamma(P_2)$
      according to (14a);
    Obtain a feasible solution $\mathbf{q}_f$;
    Calculated $v(P_2)$ according to (13a);
    $lb \leftarrow max\{lb, v(P_2)\}$, $ub \leftarrow min\{ub, v_\gamma(P_2)\}$;
    $\gamma_i(t+1) \leftarrow max\{0, \lambda^i_{u,f}(t) - \theta_i(t)g_i(t)$
    $q^l_{u,f}* \leftarrow q^l_{u,f}(t)$;
    Test termination criterion;
end
```

Algorithm 2: Pseudocode for Algorithm 2.

```
Input: $\lambda_{u,f}, d_0, B_i, d_{k,k+1}, C_i$.
Output: $x^*_{i,f}$.
Initialization: $ub = +\infty$, lower bound $lb = -\infty$, $t = 0$.
while not the termination criterion do
    Compute $\mathbf{q}^{i*}_{u,f}(t)$ by using Algorithm 1;
    Update (8a) denoted by $R(\mathbf{x}, \mathbf{q})_\eta$;
    Obtain a feasible solution $\mathbf{q}_f$ and $\mathbf{x}_f$;
    Update (11a) denoted by $R(\mathbf{x}, \mathbf{q})$;
    $lb \leftarrow max\{lb, R(\mathbf{x}, \mathbf{q})\}$; $ub \leftarrow min\{ub, R(\mathbf{x}, \mathbf{q})_\eta\}$;
    Update $\eta_i(t+1)$ according to (17) with setting the gradient direction as
      $\lambda_{u,f}(x_{i,f}(t) - q^l_{u,f}(t))$;
    $x^*_{u,f} \leftarrow x_{u,f}(t)^*$;
    Test termination criterion $TC$;
end
s
```

5. Greedy algorithm

In this section, we will show that the problem (8) belongs to the classical problems of maximizing a monotone submodular function subject to partition matroid constraints. Then we provide a solution with lower computation complexity compared to the primal-dual algorithm. This algorithm guarantees a 1.5-approximate result for the maximization problem. Before that, the definition of matroids and submodular function are provided firstly.

5.1. Preliminaries

Matroids: A matroid is a pair of $\mathscr{M} = (\mathcal{S}, \mathcal{I})$, where $\mathcal{S}$ is a finite set and $\mathcal{I} \subseteq 2^{\mathcal{S}}$ is a collection of subsets of $\mathcal{S}$ with following properties:

1). $\phi \subseteq \mathcal{I}$,
2). $\mathcal{I}$ is download closed, i.e. if $B \in \mathcal{I}$ and $A \subseteq B$, then $A \in \mathcal{I}$,
3). if $A, B \in \mathcal{I}$, and $|A| < |B|$, then $\exists e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

Submodular functions: Let $\mathcal{S}$ be a finite set. A set function $f : 2^{\mathcal{S}} \to \mathcal{R}$ is submodular if for every $A \subseteq B \subseteq \mathcal{S}$ and every $e \in S \setminus B$ we have

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \tag{18}$$

The *submodular functions* can be an expression of diminishing returns: as the set becomes larger, the benefit of adding a new element to the set will decrease.

Monotone: The function f is monotone increasing if $f(A) \leq f(B)$, $A \subseteq B \subseteq \mathcal{S}$.

5.2. Greedy algorithm

For our problem, let the ground set $\mathcal{S}$ denote the set of possible files that could be placed in all cache nodes $\mathcal{N}$. And let A denote the set of content placement management. We have $|A_i| \leq C_i$, where C_i is the storage capacity size of node i.

Lemma 1. *The storage capacity constraints in (8d) can form a partition matroid $\mathcal{M} = (\mathcal{S}, \mathcal{I})$.*

Proof. We divide the ground set $\mathcal{S}$ into N disjoint sets $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2, ..., \mathcal{S}_N$, which means the possible contents placed in N cache nodes. Recall that A is the set of contents stored in cache nodes, $a_{i,f} = 1$ if and only if $s_{i,f} \in A$. Thus, the storage capacity constraints of cache nodes can be expressed as

$$\mathcal{I} = \{A \subseteq \mathcal{S} : |A \cap S_i| \leq C_i, \forall i = 1, 2, .., N\} \tag{19}$$

Note that $(\mathcal{S}, \mathcal{I})$ defines a matroid. □

Lemma 2. *The objective function (8a) is monotone and submodular.*

Proof. Let us consider two cache placement sets A, B, such that $A \subseteq B$, and a new content placement element $e_{i,f}$, where $e_{i,f} \in S \setminus B$, meaning content f being placed in node i. $R(A)$ is denoted as margin value of delay on content placement set A, which is the change in the content access delay after adding this element to the set. Since the sum of monotone submodular functions is also monotone submodular, it suffices to prove that for each request, the set function $R_{u,f}(A)$ is monotone submodular. Then $R_{u,f}(A)$ can be simplified as,

$$R_{u,f}(A) = max_{l \in \mathcal{Q}_{u,f}}\{0, d_0 - \sum_{k=1}^{|l|-1} d_{k,k+1}\} \tag{20}$$

For express conveniently, let $d^i_{u,f}$ denote delay request experiencing through routing path $q^l_{u,f}$, where node i is the destination corresponding to the path. □

Monotone: Obviously, $R_{u,f}(A \cup \{e_{i,f}\}) \leq R_{u,f}(A)$ because adding a new file to the cache list diverts traffic from remote servers onto cache nodes. Since $R_{u,f}$ is monotonic function, we have $R_{u,f}(A) \geq R_{n,f}(B)$.

Submodular: We discuss the following three cases to prove $R_{u,f}$ is submodular:

1). If $R_{u,f}(e) > R_{u,f}(B)$, according the definition of A and B, then the margin value on two sets is $R_{u,f}(e)$.

$$\begin{aligned} &R_{u,f}(B, e_{i,f}) - R_{u,f}(B) = \\ &\quad R_{u,f}(A, e_{i,f}) - R_{u,f}(A) \end{aligned} \tag{21}$$

2). If $R_{u,f}(e) < R_{u,f}(A)$, adding the cache placement element $e_{i,f}$ wouldn't bring any caching benefit. Then the margin value of them is zero.

3). If $R_{u,f}(A) < R_{u,f}(e) < R_{u,f}(B)$, according the previous discussion, the margin value on set A and B is $R_{u,f}(e)$ and 0, respectively.

In summary, according to the analysis of the above three cases, the margin value on set A always greater or equal to the margin value on set B. The objective function R is submodular function on set S.

The greedy algorithm is a popular method to solve the problem of maximizing a submodular function subject to a partition matroid, which can yield performance at most $\frac{1}{2}$ times worse than optimal. Since our joint problem involves two subproblems, the greedy algorithm can be extended for both decisions on content placement and request routing. We have proved that the problem function in (8) is submodular function of content placement. However, for given content placement of each node, we can't make the optimal request routing decisions according to complexity analysis. The best we can get is $\frac{1}{2}$-approximate [15] solution to request routing problem.

At content placement subproblem, we iteratively choose the element with margin value being at most $\frac{1}{2}$ times worse than that of the optimal choice. Once the content placement is given, we decide to request a routing policy in the same way. According to [15], the locally greedy algorithm guarantees a tight factor-$(\alpha+1)$ result for the submodular function maximization problem over partition matroids. Since greedy algorithm yields a $\frac{1}{2}$ solution in content placement subproblem, for request routing subproblem, α is $\frac{1}{2}$. Hence we can obtain an approximation ratio of 1.5 for the joint problem.

The algorithm procedure details can be found in Algorithm 3. Algorithm 3 starts with empty cache. At each iteration, it selects the content placement element which can add the margin value with 0.5-approximate rate to optimal. When the capacity of the cache node is full, the cache placement of this node is excluded from set S. The iteration procedure continues until the storage capacity of all cache node is full.

In Algorithm 3, there are $\sum_{n=0}^{N} C_n$ iterations to fulfill the cache capacity and $\sum_{n=0}^{N} B_n$ iterations to fulfill the bandwidth capacity of all nodes. Hence the total running time of greedy algorithm is $\mathcal{O}(UFN\sum_{n=0}^{N} B_n \sum_{n=0}^{N} C_n)$ which is lower than the total running time of primal-dual algorithm.

Algorithm 3: Pseudocode for Algorithm 3.

```
Input: $\lambda_{u,f}, d_{i,j}, d_0, C_i$,.
Output: $\mathbf{X}$, $\mathbf{q}$.
Initialization: $S = s_{1,1}, s_{2,1}, ..., s_{f,n}, \forall f \in \mathcal{F}$, $X_i = \phi, \forall i \in \mathcal{N}$, $X = \phi$, c=1.
for $c < \sum_{i \in \mathcal{N}} C_i$ do
    $e_{i^*,f^*} \leftarrow argmax_{e_{i,f} \in S \setminus X} R(X + e_{i,f}) - R(X)$;
    $X_{i^*} \leftarrow X_{i^*} \cup e_{i^*,f^*}$
    if $|X_i| = C_i$ then
        $S \leftarrow S \setminus s_{i^*,f}$
    else
        $S \leftarrow S \setminus s_{i^*,f^*}$
    end
end
```

6. Evaluation

6.1. Experiment setup

Configurations. We consider a real network `abilen` [33] and a synthetic network `watts-strogatz (ws)` [36], respectively. The ws graph is generated according to the ws model of a small-world network. We consider the network with users uniformly distributed in a 2-D square, similar to [22] and the size of these networks is set to be 11. To prove the efficiency of our algorithm, we make the synthetic network sparsely. Each node of these networks has a stability parameter, ranging from 0.5 to 1. We assume there are 150 users who access 500 files. The location of users and cache nodes are spatially placed in a random and uniform manner, and users can only get service from cache nodes when their distance less than coverage radius. Since web accesses can be modeled using Zipf-like probability distributions, we assume the user u generates a collection of requests following the Poisson process with the arrival rate λ_u and we assume the file popularity follows a Zipf distribution with skewness parameter 0.8 [22]. The request directed to the remote servers experiences $d_0 = 12$ time units. Assumably, there is a negative correlation between node reliability and the delay. If there are more unreliable nodes in request path, an uncached path would become the user's preference. The remaining configurations in the experiment are depicted in Table 2.

Table 2
Experiment parameter settings.

Parameter	Settings
The area of network	100 × 100
Number of user	150
Number of file	500
Number of SBSs	11
SBS coverage radius	30
Capacity of SBSs	80
Bandwidth capacity of SBSs	300
Shape parameter of Zipf	0.8
Uncache delay	12
Stability score	0.5~1

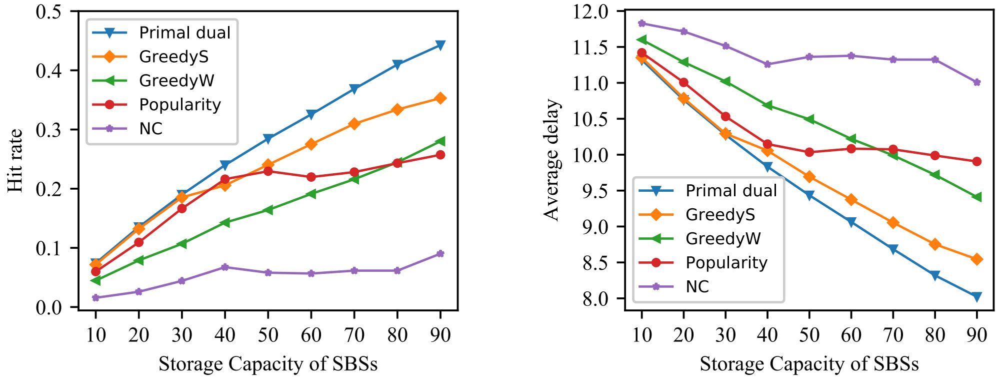
Baselines. We implement the following schemes as our comparative approaches:

1. *Non-Cooperative (NC):* Each cache server independently stores the most popular files based on local popularity at the corresponding cache server. If users fail to find contents in its pertaining SBS, the requests will be routed to remote servers directly. This mechanism has been extensively used as a benchmark by prior work [38,19,49].
2. *Popularity-based Cooperative Approach*: Each cache server stores the most popular contents based on local popularity which is similar to NC. The scheme, however, allows edge servers to fetch contents from other edge nodes collaboratively. The missed requests are randomly routed to any node that caches the required contents.
3. *Primal dual algorithm*: the proposed algorithm in § 4.
4. *Greedy algorithm considering node stability* (*GreedyS*): the proposed algorithm in § 5.
5. *Greedy without considering node stability* (*GreedyW*): the proposed algorithm in § 5 without considering the node stability.

Methodology. We mainly use the average access delay and the cache hit ratio as the main performance indicators. We vary the storage (caching) capacity and the bandwidth capacity of the caching node and observe their impact on the performance indicators, under the synthetic and real network topology, respectively (§6.2 and §6.3). In addition, we evaluate how the content popularity and distribution impact on the caching effectiveness (§6.4).

6.2. Impact of storage capacity

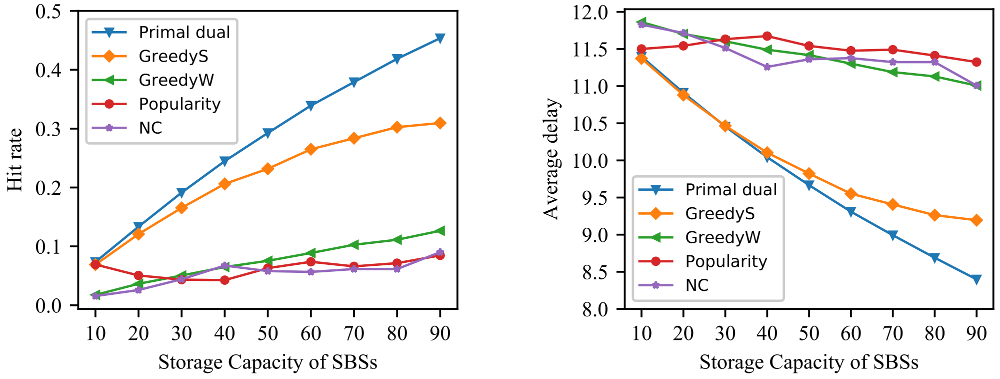
Fig. 2 and Fig. 3 demonstrates the performance comparison between our algorithms and other baselines when we vary the total cache capacity under the `abilene` network and `ws` network, respectively. Observably, the increment of the cache size can significantly improve the performance of all algorithms simply because of the increased probability of retrieving required contents. It is obvious to find out the cache hit ratio is not improved by the increase of the cache size in the NC algorithm, resulting in a negligible delay reduction. This is because caching more local popular contents has little impact on the increase of global caching benefit particularly when the popularity distribution gets deeper. By contrast, the primal dual scheme outperforms all baselines in terms of hit rate and the average access delay. Furthermore, when the cache size is no more than 40, the difference between the primal dual algorithm and GreedyS is marginal, indicating they have similar caching effectiveness, but obviously GreedyS has far lower compute complexity compared against the primal dual algorithm.



(a)

(b)

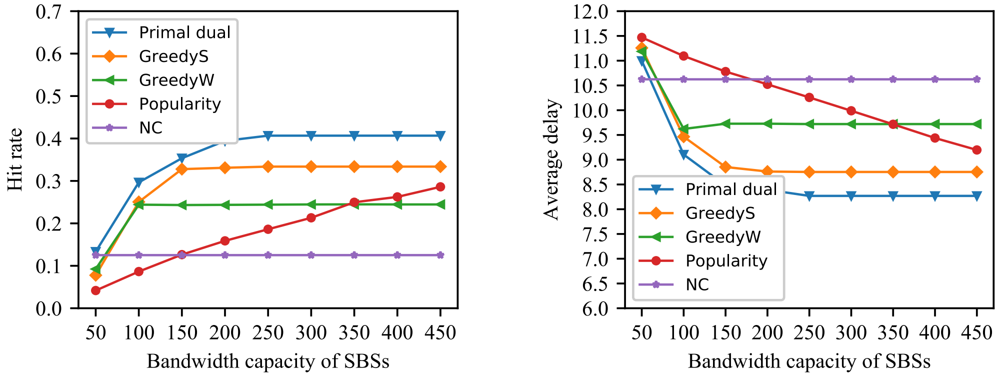
Fig. 2. The impact of storage capacity of SBSs on the cache hit rate and average content delivery delay under `abilen` network.



(a)

(b)

Fig. 3. The impact of storage capacity of SBSs on the cache hit rate and average content delivery delay under `ws` network.



(a)

(b)

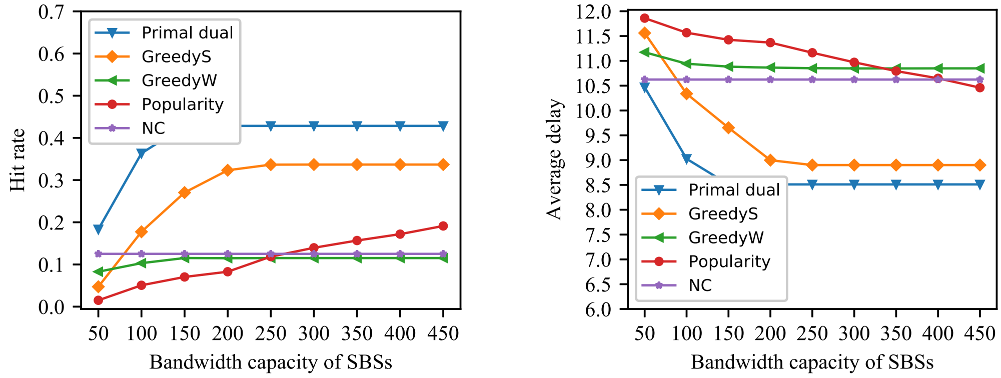
Fig. 4. The impact of bandwidth capacity of SBSs on the cache hit rate and average content delivery delay under `abilen` network.

Hence, the greedy-based algorithm would intrinsically become the preferable option to the content delivery vendors if the network is with limited storage space.

6.3. Impact of bandwidth capacity

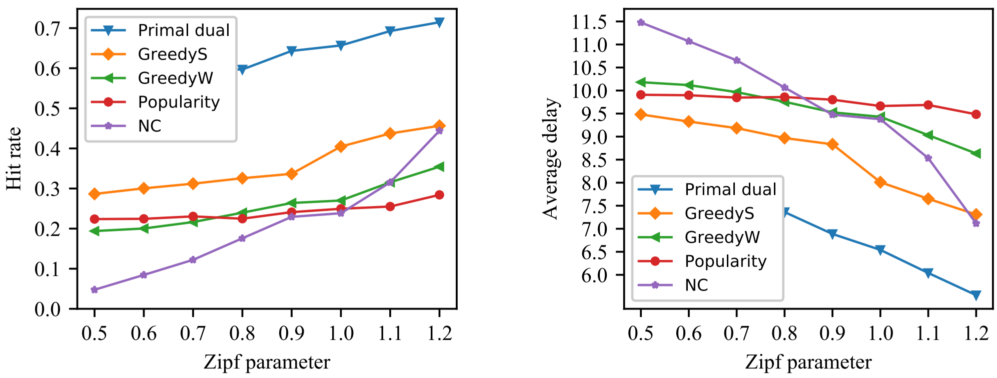
Fig. 4 and 5 compare the delivery delay on average over different bandwidth capacity in different algorithms. Since the cache size is fixed, increasing the bandwidth capacity of cache nodes cannot further improve the efficiency of different cache schemes once the value surpasses a given threshold. This is because all requests for other contents not cached in the node have to request and retrieve the content replica from remote servers even though the bandwidth capacity is large enough.

It is observable that the primal dual algorithm achieves significant reductions in average delay by up to 22% against the NC and popularity based cooperative approaches under both networks. Meanwhile, GreedyS also yields a competitive effectiveness of delay reduction in both networks. In addition, the GreedyS algorithm has a much higher cache hit ratio in both network environments,



(a) (b)

Fig. 5. The impact of bandwidth capacity of SBSs on the cache hit rate and average content delivery delay under `ws` network.



(a) (b)

Fig. 6. The impact of Zip parameter on cache hit rate and average content delivery delay under `abilen` network.

indicating the non-trivial importance of node reliability in the modeling. The cache hit ratio can be linearly increased by the popularity-based cooperative approach. This is because the growth of bandwidth capacity will enable the sufficient data transfer and make a better use of the holistic caching among different cache nodes. As opposed to the cooperative mechanism that can share the global cache capacity, the non-cooperative approach fails to increase the cache hit rate and is less effective in improving the end-to-end performance delivery service. As a result, the bandwidth resources is utilized in an imbalanced manner among different cache nodes.

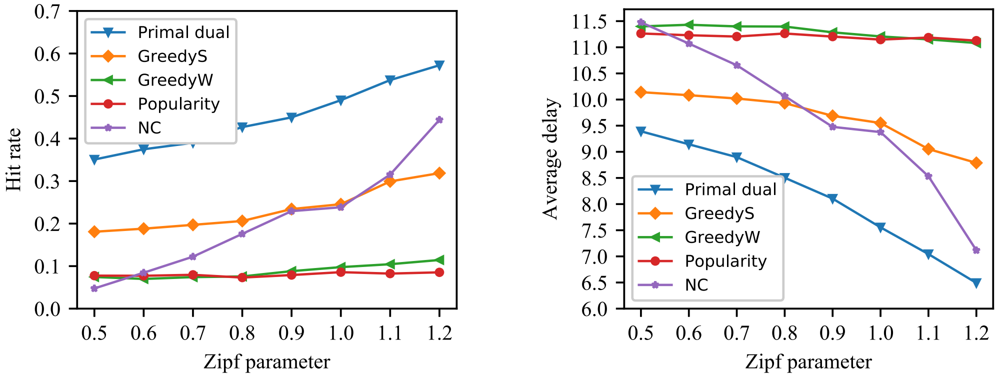
6.4. Impact of Zipf parameter α

We also evaluate the sensitivity of different algorithms to the Zipf parameter (α) implicating the popularity of the cached content. To do so, we fix the cache size and bandwidth capacity. To achieve a fair comparison, files are randomly permuted in each cache node. As shown in Fig. 6 and Fig. 7, the performance of the hit rate and average delay is significantly improved when the popularity of some specific contents take up a large proportion of the total contents. In other words, there is a skewness among different contents and higher α indicates a heavy-tailed distribution of the content popularity. Obviously, higher hit rate and reduced delivery delay manifest when the level of popularity picks up, because duplicated requests target the same piece of content. More specifically, the proposed primal dual algorithm outperforms other competitors with the varied α under both networks. Meanwhile, GreedyS has a less effective performance when compared with primal dual under different network scenarios. However, considering the lower computation complexity, the greedy-based implementation is more suitable for these network scenarios with stringent capacity constraints.

6.5. Discussion

Although we normally assume content requests follows the Poisson process with the arrival rate λ_u and the file popularity follows a Zipf distribution, the surging data flow rate, i.e., the content throughput or request number, indeed has an impact on the users' QoS, particularly when the cache size is in short supply at the edge nodes. The content delivery delay, the critical and explicit performance indicator of a caching network, is susceptible to the cache hit ratio which is largely subject to the amount of content throughput and the number of content requests.

On the one hand, the arrival of surging throughput will undoubtedly challenge the caching capacity and diminish the caching hit ratio, thereby tremendously degrading the content delivery service. On the other hand, the impact of bursting requests can be considered on a case-by-case basis: 1) the surging requests associated with the same content (e.g., emerging popular video published by social influencers or web celebrity) would not necessarily reduce the cache hit ratio despite flooding requests coming through the system. As long as the content replicas have already cached at the edge end, it will have little impact on the end-to-end delivery delay; 2) a large number of disperse requests tend to have direct impact on the cache hit ratio and increase the delivery delay. The cached contents have to be replaced more frequently,



(a) (b)

Fig. 7. The impact of Zip parameter on cache hit rate and average content delivery delay under ws network.

leading to non-negligible costs stemming from request direction to remote servers with substantial bandwidth consumption and potential traffic congestion over the backhaul network.

7. Conclusion and future work

This paper presents a cooperative caching mechanism for content placement and request routing in unreliable CDNs, supported by two new optimization algorithms for minimizing the access delay. Importantly, although the greedy algorithm is inferior to the primal-dual algorithm in terms of hit rate and average delay, it is particularly suitable for networking scenarios where node connectivity is sparse and bandwidth and storage space are scarce. The proposed model can be easily extended to many other areas, including path selections in SDN and constraint-based resource scheduling in cloud data centers. We plan to incorporate user mobility into the solution to tackle the impact of local popularity of requests. The centralized nature of the Caching Management Entity might potentially threaten the system scalability, and managing cache state and network flow is often energy-consuming. Tackling these issues would complicate the current design of the optimization models, but it is an area that deserves further investigation. Adaptive distributed caching schemes based on the measurement of computing energy consumption will be therefore an interesting area to explore in the future work.

CRediT authorship contribution statement

Youmei Song: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft. **Tianyu Wo:** Data curation, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing. **Renyu Yang:** Funding acquisition, Project administration, Supervision, Writing – review & editing. **Qi Shen:** Writing – review & editing. **Jie Xu:** Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Acknowledgments

The work is supported by National Key Research and Development Program (2016YFB1000103) and the UK EPSRC Research Grant (EP/T01461X/1). This work is also partially supported by the White Rose University Consortium.

References

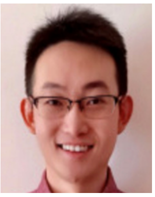
[1] Disk drive prices (1955-2014), http://www.jcmit.com/diskprice.htm.
[2] http://www.networkcomputing.com/networking/high-price-it-downtime/856595126.
[3] Ejder Bastug, Mehdi Bennis, Mérouane Debbah, Living on the edge: the role of proactive caching in 5g wireless networks, IEEE Commun. Mag. 52 (8) (2014) 82–89.
[4] Cache "Less for More" in Information-Centric Networks (Extended Version), Comput. Commun. 36 (7) (2013) 758–770, ISSN 0140-3664.
[5] Alberto Ceselli, Marco Premoli, Stefano Secci, Mobile edge cloud network design optimization, IEEE/ACM Trans. Netw. (2017) 1–14.
[6] Wei Koong Chai, Vasilis Sourlas, George Pavlou, Providing information resilience through modularity-based caching in perturbed information-centric networks, in: 2017 29th International Teletraffic Congress (ITC 29), vol. 1, IEEE, 2017, pp. 214–222.
[7] Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021. White paper, 2017.
[8] Jie Dai, Zhan Hu, Bo Li, Jiangchuan Liu, Baochun Li, Collaborative hierarchical caching with dynamic request routing for massive content distribution, in: 2012 Proceedings IEEE INFOCOM, IEEE, 2012, pp. 2444–2452.
[9] Mostafa Dehghan, Bo Jiang, Anand Seetharam, Ting He, Theodoros Salonidis, Jim Kurose, Don Towsley, Ramesh Sitaraman, On the complexity of optimal request routing and content caching in heterogeneous cache networks, IEEE/ACM Trans. Netw. 25 (3) (2016) 1635–1648.
[10] Peter Garraghan, Renyu Yang, Zhenyu Wen, Alexander Romanovsky, Jie Xu, Rajkumar Buyya, Rajiv Ranjan, Emergent failures: rethinking cloud reliability at scale, IEEE Cloud Comput. 5 (5) (2018) 12–21.
[11] X. Ge, S. Tu, G. Mao, C. Wang, T. Han, 5g ultra-dense cellular networks, IEEE Wirel. Commun. 23 (1) (2016) 72–79.
[12] Ammar Gharaibeh, Abdallah Khreishah, Bo Ji, Moussa Ayyash, A provably efficient online collaborative caching algorithm for multicell-coordinated systems, IEEE Trans. Mob. Comput. 15 (8) (2015) 1863–1876.
[13] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The Google file system, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, 2003, pp. 29–43.
[14] Phillipa Gill, Navendu Jain, Nachiappan Nagappan, Understanding network failures in data centers: measurement, analysis, and implications, in: Proceedings of the ACM SIGCOMM 2011 Conference, 2011, pp. 350–361.
[15] Pranava R. Goundan, Andreas S. Schulz, Revisiting the greedy approach to submodular set function maximization, Optimization (2007) 1–25.
[16] I-Hong Hou, Tao Zhao, Shiqiang Wang, Kevin Chan, Asymptotically optimal algorithm for online reconfiguration of edge-clouds, in: Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2016, pp. 291–300.
[17] C. Hu, J. Xu, M. Zhang, R. Yang, T. Wo, Dˆ2 ps: a dependable data provisioning service in multi-tenant cloud environment, in: 2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE), IEEE, 2016, pp. 252–259.
[18] Stratis Ioannidis, Edmund Yeh, Jointly optimal routing and caching for arbitrary network topologies, IEEE J. Sel. Areas Commun. 36 (6) (2018) 1258–1275.
[19] Wei Jiang, Gang Feng, Shuang Qin, Optimal cooperative content caching and delivery policy for heterogeneous cellular networks, IEEE Trans. Mob. Comput. 16 (5) (2016) 1382–1393.

[20] Abdallah Khreishah, Jacob Chakareski, Collaborative caching for multicell-coordinated systems, in: 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2015, pp. 257–262.
[21] Chunlin Li, Jingpan Bai, JianHang Tang, Joint optimization of data placement and scheduling for improving user experience in edge computing, J. Parallel Distrib. Comput. 125 (2019) 93–105.
[22] Jian Li, Truong Khoa Phan, Wei Koong Chai, Daphne Tuncer, George Pavlou, David Griffin, Miguel Rio, Dr-cache: distributed resilient caching with latency guarantees, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 441–449.
[23] Xiuhua Li, Xiaofei Wang, Peng-Jun Wan, Zhu Han, Victor CM Leung, Hierarchical edge caching in device-to-device aided mobile networks: modeling, optimization, and design, IEEE J. Sel. Areas Commun. 36 (8) (2018) 1768–1785.
[24] Zhen Liu, Philippe Nain, Nicolas Niclausse, Don Towsley, Static Caching of Web Servers, Multimedia Computing and Networking 1998, vol. 3310, International Society for Optics and Photonics, 1997, pp. 179–190.
[25] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, Christophe Diot, Characterization of failures in an operational ip backbone network, IEEE/ACM Trans. Netw. 16 (4) (2008) 749–762, ISSN 1063-6692.
[26] Francesco Pantisano, Mehdi Bennis, Walid Saad, Mérouane Debbah, In-network caching and content placement in cooperative small cell networks, in: 1st International Conference on 5G for Ubiquitous Connectivity, IEEE, 2014, pp. 128–133.
[27] G.S. Park, H. Song, Cooperative base station caching and x2 link traffic offloading system for video streaming over sdn-enabled 5g networks, IEEE Trans. Mob. Comput. 18 (9) (2019) 2005–2019.
[28] Daniel Pérez Palomar, Mung Chiang, A tutorial on decomposition methods for network utility maximization, IEEE J. Sel. Areas Commun. 24 (8) (2006) 1439–1451.
[29] Konstantinos Poularakis, Leandros Tassiulas, On the complexity of optimal content placement in hierarchical caching networks, IEEE Trans. Commun. 64 (5) (2016) 2092–2103.
[30] Konstantinos Poularakis, George Iosifidis, Antonios Argyriou, Iordanis Koutsopoulos, Leandros Tassiulas, Caching and operator cooperation policies for layered video content delivery, in: IEEE INFOCOM 2016-the 35th Annual IEEE International Conference on Computer Communications, IEEE, 2016, pp. 1–9.
[31] Dapeng Qu, Xingwei Wang, Min Huang, Keqin Li, Sajal K. Das, Sijin Wu, A cache-aware social-based qos routing scheme in information centric networks, J. Netw. Comput. Appl. 121 (2018) 20–32, ISSN 1084-8045.
[32] Guocong Quan, Jian Tan, Atilla Eryilmaz, Counterintuitive characteristics of optimal distributed lru caching over unreliable channels, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 694–702.
[33] Dario Rossi, Giuseppe Rossini, Caching performance of content centric networks under multi-path routing (and more), Relatório técnico, Telecom ParisTech, 2011, pp. 1–6.
[34] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G. Dimakis, Andreas F. Molisch, Giuseppe Caire, Femtocaching: wireless content delivery through distributed caching helpers, IEEE Trans. Inf. Theory 59 (12) (2013) 8402–8413.
[35] V. Sourlas, P. Flegkas, L. Tassiulas, Cache-aware routing in Information-Centric Networks, in: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013, pp. 582–588.
[36] R. Steven, H. Strogatz, Duncan J. Watts, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[37] Tuyen X. Tran, Dario Pompili, Octopus: a cooperative hierarchical caching strategy for cloud radio access networks, in: 2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), IEEE, 2016, pp. 154–162.
[38] Tuyen X. Tran, Duc V. Le, Guosen Yue, Dario Pompili, Cooperative hierarchical caching and request scheduling in a cloud radio access network, IEEE Trans. Mob. Comput. 17 (12) (2018) 2729–2743.
[39] Huaimin Wang, Peichang Shi, Yiming Zhang, Jointcloud: a cross-cloud cooperation architecture for integrated Internet service customization, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 1846–1855.
[40] T. Wang, C.Q. Wu, Y. Wang, A. Hou, H. Cao, Multi-path routing for maximum bandwidth with k edge-disjoint paths, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 1178–1183.
[41] Xiaofei Wang, Min Chen, Tarik Taleb, Adlen Ksentini, Victor CM Leung, Cache in the air: exploiting content caching and delivery techniques for 5g systems, IEEE Commun. Mag. 52 (2) (2014) 131–139.
[42] Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, Michael Rovatsos, Fog orchestration for Internet of things services, IEEE Internet Comput. 21 (2) (2017) 16–24.
[43] Zhenyu Wen, Tao Lin, Renyu Yang, Shouling Ji, Rajiv Ranjan, Alexander Romanovsky, Changting Lin, Jie Xu, Ga-par: dependable microservice orchestration framework for geo-distributed clouds, IEEE Trans. Parallel Distrib. Syst. (2019).
[44] Zichuan Xu, Weifa Liang, Wenzheng Xu, Mike Jia, Guo Song, Efficient algorithms for capacitated cloudlet placements, IEEE Trans. Parallel Distrib. Syst. 27 (10) (2015) 1.
[45] Renyu Yang, Jie Xu, Computing at massive scale: scalability and dependability challenges, in: 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, 2016, pp. 386–397.
[46] Renyu Yang, Yang Zhang, Peter Garraghan, Yihui Feng, Jin Ouyang, Jie Xu, Zhuo Zhang, Chao Li, Reliable computing service in massive-scale systems through rapid low-cost failover, IEEE Trans. Serv. Comput. (2017).
[47] Bo Yi, Xingwei Wang, Min Huang, Content delivery enhancement in vehicular social network with better routing and caching mechanism, J. Netw. Comput. Appl. 177 (2021) 102952, ISSN 1084-8045.
[48] F. Zhang, Y. Zhang, D. Raychaudhuri, Edge caching and nearest replica routing in information-centric networking, in: 2016 IEEE 37th Sarnoff Symposium, 2016, pp. 181–186.
[49] Shan Zhang, Peter He, Katsuya Suto, Peng Yang, Lian Zhao, Xuemin Shen, Cooperative edge caching in user-centric clustered mobile networks, IEEE Trans. Mob. Comput. 17 (8) (2017) 1791–1805.

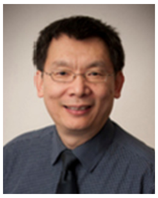
Youmei Song received her MS degree in the School of Computer Science and Engineering from Qingdao University in 2017. She is currently working toward the PhD degree in the National Key Laboratory of Software Development Environment, Beihang University, Beijing, China. Her research interests include edge computing, distributed system, resource allocation, optimization.

Tianyu Wo (Member, IEEE) received the BEng and PhD degrees both in computer science from Beihang University, Beijing, China, in 2001 and 2008, respectively. He is currently an associate professor with the National Key Laboratory of Software Development Environment, Beihang University, Beijing, China. His current research interests include distributed systems, network operation systems, and IoV systems.

Renyu Yang received the PhD degree from Beihang University. He is a research fellow with the University of Leeds, United Kingdom. He is also an adjunct research scientist with BDBC, Beihang University, China. He was previously with Alibaba Group and had industrial experience in building large-scale systems. His research interests include reliable resource management, distributed systems, data processing, etc. He is a member of the IEEE.



Qi Shen is a postdoctoral researcher in Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, where he is investigating data processing systems for geographically distributed clusters. He received his PhD degree in 2017 from The University of Newcastle Upon Tyne for work on semantically aware transactional concurrency control for GPGPU computing. His research interests include parallel programming, software transactional memory, and distributed architectures.



Jie Xu (Member, IEEE) is currently a chair professor of computing with the University of Leeds, director of UK EPSRC WRG e-Science Centre, and chief scientist of BDBC, Beihang University, China. He has worked in the field of dependable distributed computing for more than 30 years. He is a Steering/Executive Committee member for numerous IEEE conferences including SRDS, ISORC, HASE, SOSE, and is a co-founder for IEEE IC2E. He has led or co-led many research projects to the value of more than $30M, and published more than 300 academic papers, book chapters, and edited books.