Baiji: Domain Planning for CDNs under the 95th Percentile Billing Model

Juan Vanerio[*][†], Huiran Liu[‡], Qi Zhang[‡] and Stefan Schmid[‡][§][*]
[*]Faculty of Computer Science, University of Vienna, Austria
[†]AIT Austrian Institute of Technology, Austria
[‡]INET, Faculty IV (EECS), Technical University Berlin, Germany
[§]Fraunhofer SIT, Germany

***Abstract*—Content Distribution Networks (CDNs) play a crucial role in efficiently delivering online content to end-users. In this paper, we initiate the study of CDN domain planning with flexible assignments of domains to Points of Presence (PoPs) within a CDN, with the objective of minimizing the cost of transmissions while providing sufficient resources to serve the communication demands. The problem is subject to practical constraints of network deployment such as a percentile-based billing model, PoP's bandwidth and committed rate limits, geographic locality and quota constraints and minimum per domain cache-hit-ratios.**

We formulate the problem as an offline optimization task with a nonlinear objective function and linear constraints, which becomes computationally intensive for medium-sized instances. The 95th percentile billing model, commonly used by service providers, contributes significantly to this non-linearity. To address this, we propose *Baiji*, a multi-algorithm approach leveraging insights from our formulation.

Our empirical evaluation of *Baiji* on two synthetic and one real-world workloads demonstrates its effectiveness in approaching the upper bound on system performance. *Baiji* provides high-quality solutions for CDN monthly planning, with tunable execution times (from tens of seconds up to four hours), making *Baiji* suitable for practical deployment in CDNs.

I. Introduction

In today's Internet architecture, Content Delivery Networks (CDNs) play a crucial role in facilitating effective content dissemination. CDNs are widely used for media streaming, e-commerce, social media, and cloud computing. As a reference, in 2022, CDNs served most of the video-based traffic, which constitutes 65% of global Internet traffic [1].

Operating as large distributed systems, CDNs are designed to expedite content delivery and ensure reliability through the dissemination of content across diverse locations. By caching content closer to users and distributing it, CDNs effectively mitigate latency, alleviate network congestion, and substantially enhance overall performance [2], [3].

A key challenge for CDNs is to plan and optimize the placement of domains across multiple Points of Presence (PoPs). Each PoP is composed of caching devices dedicated to disseminating content spanning from web pages to images and videos. These interconnected PoPs form an intricate network architecture facilitated by high-speed networks, thereby establishing a robust content delivery infrastructure. The placement of domains to PoPs affects the quality of the service to end users. It also affects the operational costs that CDNs have to pay to Internet service providers (ISPs), such as bandwidth fees, peering charges, and infrastructure expenses.

Research partially funded by the German BMBF (Federal Ministry of Education and Research) project, 6G-RIC: 6G Research and Innovation Cluster, grant 16KISK020K, 2021-2025. We thank our anonymous reviewers for their insightful comments. ISBN 978-3-903176-63-8 © 2024 IFIP

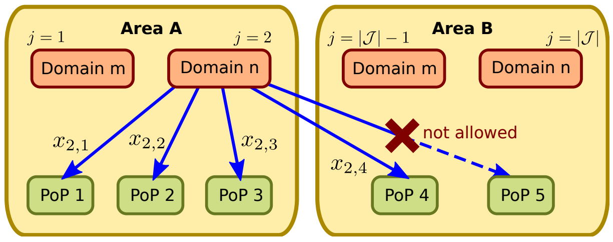


Fig. 1. **An instance of the CDN domain planning problem.** Domain names n and m must be served to users on areas A and B, containing resp. three and two PoPs. The areas and domains form domain-area pairs (indexed by j). PoP 5 is forbidden to serve domain n in area A. Attributes x_{ji} indicate the demand fraction for pair j served from PoP i.

The 95th percentile billing model is widely used for paying for traffic transmission. It calculates charges based on the 95th percentile of monthly traffic, exempting the top 5% from fees [4]. This allows CDNs to exceed their nominal rate without extra costs, as long as their 95th percentile stays below it.

This paper initiates the study of the flexible domain to PoP assignment in CDNs (henceforth referred to as *CDN domain planning*), based on the prediction of monthly traffic demand, the 95th percentile billing model, and realistic constraints involving available bandwidth capacities, committed rates, cache-hit-ratios and geographic distances. The main challenge in addressing the problem stems from the difficult objective function and the extensive number of variables and constraints. Typical instances involve hundreds of PoPs and tens of thousands of domains.

Regarding the constraints, the *bandwidth* constraint requires that a predefined limit not be exceeded at any time when allocating traffic to a PoP. To keep latency low, users should be served from nearby PoPs. In some cases, a significant fraction of the domain traffic must be served from PoPs in the same geographical region as the users to ensure sufficient service quality. The locality-related constraints must not be confused with the *allocation granularity* constraints [5]. The latter require a minimum allocation granularity per PoP, while the former restrict the distances between PoPs and their users.

First, we present a mathematical formalization for the domain planning problem, formulating it as an offline optimization task with a nonlinear objective based on the 95th percentile billing model and linear constraints. We then propose *Baiji*, a multi-algorithm system that combines heuristic methods to generate good solutions.

Baiji executes each algorithm and gathers the best solution from each. Then, a custom genetic algorithm utilizes these solutions as members of its initial population, yielding the best solution for domain placement as the output. The solution produced by the genetic algorithm is guaranteed to be no worse than its initial candidates.

Baiji provides high-quality solutions while balancing computational efficiency. Empirical performance evaluation of *Baiji* on two synthetic and one real-world datasets shows the effectiveness of the approach. Our key contributions can be summarized as follows:

- We ***initiate the study*** of CDN domain planning with flexible assignments of domains to PoPs, accounting for practical network deployment constraints.
- We present a ***mathematical formalization*** of the problem, enabling the derivation of exact solutions.
- We introduce the concept of ***affinity*** between a fractional demand and a PoP's current traffic time series as an optimization subproblem derived from the 95th percentile billing model and use it to develop heuristic solutions.
- We propose *Baiji*, a ***system composed of multiple algorithms*** of different natures utilizing insights derived from the model, integrated through a customized genetic algorithm, to solve the CDN planning problem.
- We report an ***empirical evaluation on a real dataset*** and two synthetic datasets, finding that *Baiji* can solve realistic monthly planning problems in controllable time. The solutions are close to the system's upper bound and can outperform a single-PoP solution.

The rest of the paper is organized as follows. Section II describes the problem as an optimization model and analyzes it. Section III presents *Baiji* system's design and components, and Section IV provides the results of the performance evaluation. Finally, we review the related work in Section V, conclude our work and discuss future research directions in Section VI.

II. Mathematical Model

Each PoP serves specific domain names to users, and each domain name needs to be responsive in multiple regions (*areas*). This combination of domain names and areas is referred to as domain-area pairs (or simply *pairs*), and we assume knowledge of their traffic patterns or time series. In a nutshell, the planning problem consists of deciding the fraction of the demand of each domain-area pair to be served from which PoPs so that communication costs are minimized while physical and operating constraints are satisfied.

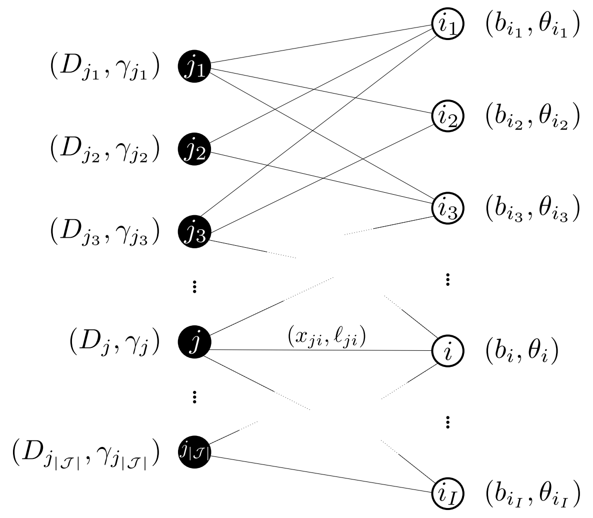

(D_{j_1}, γ_{j_1})
j_1
(D_{j_2}, γ_{j_2})
j_2
(D_{j_3}, γ_{j_3})
j_3
(D_j, γ_j)
j
$(D_{j_{|\mathcal{J}|}}, \gamma_{j_{|\mathcal{J}|}})$
$j_{|\mathcal{J}|}$
(x_{ji}, ℓ_{ji})
i_1
(b_{i_1}, θ_{i_1})
i_2
(b_{i_2}, θ_{i_2})
i_3
(b_{i_3}, θ_{i_3})
i
(b_i, θ_i)
i_I
(b_{i_I}, θ_{i_I})

Fig. 2. **Bipartite graph for the domain allocation problem.** Black nodes represent domain-area pairs, with demands and local ratio quotas as attributes. White nodes stand for PoPs, characterized by their bandwidth and committed rates. Edges indicate that a PoP is allowed to serve a pair, with ℓ_{ji} indicating whether the PoP i in the area of pair j.

An example setup is shown in Fig. 1. Areas A and B and domain names n and m constitute four domain-area pairs (indexed by $j \in [1, |\mathcal{J}|]$). Notice that not every domain is required to be served on every area, therefore, $\mathcal{J}$ is a subset of the set product of domains and areas. Values x_{ji} indicate the fraction of demand for pair j served from PoP i towards the users. Simultaneously, PoP 5 is not allowed to serve traffic for domain n in area A ($x_{2,5}$ = 0) due to latency reasons.

Various additional constraints must be accounted for. The throughput of each PoP must not exceed its nominal bandwidth and only serve traffic for allowed pairs. Also, some pairs require a minimum fraction of their traffic to be served from *local* PoPs to ensure low service times. In Fig. 1, PoPs 1 to 3 are local to area A, while 4 and 5 are local to area B. Additionally, ISP connectivity contracts for the PoPs may include minimum committed access rates that must be paid even if the traffic is below the reference value.

The problem can be modeled as an optimization problem for the aggregated traffic transmission costs subject to the discussed constraints. The main task is, therefore, to find a suitable distribution of pairs' demand fractions to PoPs.

A. Notation

We use the following notation in this paper.

- $i \in [1, I]$: Point of Presence (PoP).
- b_i: maximum bandwidth of PoP i.
- $a \in [1, A]$: geographical *area*. Each PoP is located in a single area.
- $n \in [1, N]$: domain name (index).
- $j = (n, a) \in \mathcal{J} \subseteq [1, N] \times [1, A]$: domain-area pair index. Indicates a domain name n being served in a specific area a, and $j = (n, a)$. As some domains may not be served in some areas, $\mathcal{J}$ is a subset of the product of domain names and PoPs.

- $x_{ji} \in [0,1]$, $\forall j \in \mathcal{J}, \forall i \in [1,I]$: control variables indicating the demand traffic fraction from domain-area pair j to be served from PoP i.
- $X = \{x_{ji}\}$: $|\mathcal{J}| \times [1,I]$ allocation matrix.
- T: number of 5-minute time slots on the time series.
- $\mathbf{D}_j = (d_{j1}, \ldots, d_{jt}, \ldots, d_{jT})$: row vector of size T of traffic demand time series for domain-area pair j.
- $D = \{d_{jt}\}$: demand matrix with dimensions $|\mathcal{J}| \times [1,T]$.
- $Q_{95}(\cdot)$ returns the 95th percentile of a time series (along the time dimension). If the time series is a vector, the result is a scalar. If the input is a matrix, the result is a vector with the same size as rows have the matrix.
- $\ell_{ji} \in \{0,1\}$, $\forall j \in \mathcal{J}, \forall i \in [1,I]$: locality indicator:

$$\ell_{ji} = \begin{cases} 1 & \text{if } i \in a, \text{ for } j = (n,a) \\ 0 & \text{otherwise.} \end{cases}$$

- $c_{ji} \in [0,1]$, $\forall j \in \mathcal{J}, \forall i \in [1,I]$ indicate whether traffic for pair j is allowed to be delivered from PoP i ($c_{ji} = 1$), or not ($c_{ji} = 0$). Local traffic ($\ell_{ji} = 1$) is always allowed.
- $\gamma_j \in [0,1]$: minimum required local traffic ratio for the pair j; at least γ_j of the pair's demand must be delivered from PoPs in the same area as j.
- $\theta_i \in [0,1]$: commited rate or minimum pre-agreed bandwidth usage for PoP i.
- $\mathbf{P}_i = \sum_{j \in \mathcal{N}_G(i)} \mathbf{D}_j x_{ji}$: vector of daily throughput pattern served from pop i, with elements p_{it}.
- $\mathbf{P}$: PoPs' throughput matrix, whose i-th column is $\mathbf{P}_i$.
- $(M)_i$, for any matrix M returns its i-th row.

B. Graph Abstraction

We model the problem with a graph approach illustrating the relationship among concepts. In a nutshell, the set of domain-area pairs $\mathcal{J}$ and the set of PoPs $[1,I]$ constitute a bipartite graph $G = (\mathcal{J}, [1,I], E)$, as shown in Fig. 2. Each tuple $j = (n,a)$ of domain-name n and area a is represented by a node in the first set, and each PoP i by a node in the second set. Let the forbidden allocations be a set of pairs $C \subseteq \mathcal{J} \times [1,I]$, indicating which PoPs cannot be used to serve traffic for j. An edge is drawn between every domain-area pair node and PoP node allowed to provide traffic. Therefore, the set $E := (\mathcal{J} \times [1,I]) \setminus C$ captures all allowed domain to PoP allocations. Each edge $(j,i) \in E$ has attributes x_{ji} and ℓ_{ji}: the control variable and the locality indicator, respectively.

The domain-area pair nodes j have attributes D_j and γ_j, respectively, the demand pattern and the minimum required local traffic ratio. The nodes i in the second set have attributes b_i and θ_i, respectively, bandwidth and committed rate. The set of neighbors of each domain-area pair node j in graph G, $\mathcal{N}_G(j)$, is the set of PoPs allowed to serve traffic for the pair. On the other hand, the set of neighbors of PoP node i, $\mathcal{N}_G(i)$, is the set of pairs allowed to be served from the PoP.

With this conception, matrix X becomes the weighted bi-adjacency matrix for graph G. Furthermore, the total traffic matrix can be computed as $P = X^T D$.

C. Optimization Framework

With the notation and graph models presented in previous paragraphs, the CDN domain planning project can be stated as the following optimization problem:

$$\max_X r = \frac{Q_{95}(\sum_i (X^T D)_i)}{\sum_i \max(Q_{95}(X^T D)_i, \theta_i)} - f(X) \tag{1}$$

subject to:

- **Bandwidth constraint**:

$$\sum_{j \in \mathcal{N}_G(i)} \mathbf{D}_j x_{ji} \leq b_i \quad \forall i \in [1,I], \quad \forall t \tag{2}$$

- **Demand conservation**:

$$\sum_{i \in \mathcal{N}_G(j)} x_{ji} = 1 \quad \forall j \in \mathcal{J} \tag{3}$$

- **Forbidden allocations** Matrix $C = \{c_{ji}\}$ with $c_{ji} \in \{0,1\}$, such that:

$$c_{ji} = 0 \Rightarrow x_{ji} = 0 \tag{4}$$

- **Local ratio**: Guarantee a given minimum per-domain traffic ratio quota in each area.

$$\sum_{i \in \mathcal{N}_G(j)} x_{ji} \cdot \ell_{ji} \geq \gamma_j \quad \forall j \in \mathcal{J} \tag{5}$$

Eq. 1 is the non-linear and non-differentiable objective function with penalization term $f(X)$. The constraint in Eq. 2 represents the limit imposed by the finite capacity of the PoPs to transmit traffic, Eq. 3 captures the conservation of the domain's demand and Eq. 4 represents an allocation filter implicit in graph G, as forbidden associations between domain name-area tuples and PoPs do not have an edge in G. Finally, Eq. 5 defines the minimum local PoPs traffic fraction requirements for each pair to ensure low latency and the required service quality.

a) Objective Function: $r(\cdot)$ measures the performance ratio of the distributed solution. Its first component's numerator represents the 95th percentile of the sum of traffic from all PoPs, and the denominator is (roughly) the sum of the 95th percentiles of all pops.

The objective function accommodates two 'soft' constraints. First, to provide a minimum Cache Hit Rate (CHR) per PoP, each domain name traffic cannot be spread too thin between too many PoPs. CDNs are interested in high CHRs to ensure sufficient traffic flow for each PoP and a low amount of content replication: too much replication implies a waste of PoP's storage capacity. As this requirement can be addressed by favoring sparse allocation matrices, a direct penalty $f(X) \geq 0$ on the cardinality of the superset of domain placements induced by X is introduced:

$$f(X) = \lambda \sum_{i,j} \mathbb{1}_{x_i, j > 0} \tag{6}$$

with penalization factor λ being an hyper-parameter. Larger penalization factors favor sparser solutions.

Second, the objective function integrates the committed rates or minimum prepaid traffic (θ_i) and actual traffic $Q_{95}(X^T D)_i$ for each PoP in its denominator. Allocating at least θ_i of traffic to each PoP i aligns with cost minimization by utilizing prepaid resources effectively.

b) A single-PoP Baseline: In the objective function, the denominator relates to total traffic transmission costs while the numerator represents the cost if all traffic were served from a *single-PoP* with sufficient resources. Interpreted this way, the objective function compares the transmission cost of having a distributed versus a large single-PoP. A value of 1 signifies equal outcomes (ignoring penalties), while a value larger than 1 indicates that the distributed solution is more cost-efficient.

The single-PoP setup (e.g., a large data center) provides a baseline to compare against a distributed deployment. It simplifies content planning as there is just one hosting option for domains and no geographical constraints, meaning its costs are determined by the demand patterns alone. Nevertheless, the single-PoP solution may incur an opportunity cost as under the 95th percentile billing model costs might be reduced further by strategically placing together traffic patterns that make use of the 5% "free" time slots simultaneously.

c) Upper Bound on Performance: From the problem formulation, it is possible to derive a theoretical upper bound to compare potential solutions.

Using the demand conservation constraint (Eq. 3), it can be proven that the numerator expression in the first component of r is independent of the control variable X after the sum:

$$\begin{aligned} Q_{95}\left(\sum_i X^T D\right) &= Q_{95}\left(\sum_i \sum_j d_{jt} x_{ji}\right) \\ &= Q_{95}\left(\sum_j d_{jt} \sum_i x_{ji}\right) \\ &= Q_{95}\left(\sum_j d_{jt}\right) \end{aligned} \tag{7}$$

Second, a natural lower bound on the denominator can be obtained by adding up all the committed traffic (θ_i):

$$\sum_i \max\left(Q_{95}(X^T D)_i, \theta_i\right) \geq \sum_i \theta_i \tag{8}$$

Putting together Eq. 7, Eq. 8 and the fact that $f(X) \geq 0$,

$$\begin{aligned} r &= \frac{Q_{95}(\sum_i (X^T D)_i)}{\sum_i \max\left(Q_{95}(X^T D)_i, \theta_i\right)} - f(X) \\ &\leq \frac{Q_{95}\left(\sum_j d_{jt}\right)}{\sum_i \theta_i} \end{aligned} \tag{9}$$

As the elements on the right-hand side of Eq. 9 are given by the dataset, the upper bound can be computed directly. Notice that this upper bound is idealized and not always reachable in practice (and in our model), as effects such as the forbidden constraints and the interaction between multiple demand patterns may also prevent achieving the bound.

d) Direct Solution: The domain planning problem is expressed as an optimization problem with a non-continuous, non-convex objective function with different linear constraints. Assuming that at least one feasible solution exists, it can be attempted to be solved directly. Most commercial solvers are designed for Mixed Integer-Linear Problems, which do not apply to the problem in Section II-C. Attempts at solving the problem directly using general solvers (e.g., Scipy [6]) run for days for most instance sizes of interest, which is unpractical.

III. The Baiji System

In this section, we present the design of *Baiji*. As attempting to solve the problem directly is not practical (see Section II), we develop a set of fast heuristic algorithms that compute solutions and aggregate them with a genetic algorithm capable of improving further.

A. Algorithms.

The following algorithms were designed and implemented as part of *Baiji* for this purpose:

a) ***Monte Carlo approach (MC):*** Generates many candidate matrices X following a $[0, 1]$ uniform distribution on each cell, and returns the candidate with the highest objective function value among those that satisfy the constraints.

b) ***Greedy Fast with Post Rebalance (GFPR):*** Adopts a two-stage strategy. The first stage implements a greedy approach that iteratively updates matrix X by selecting the domain-area pair with the highest *peak* unallocated traffic demand and assigning it to the PoP with the largest remaining bandwidth (represented as a scalar). The second stage analyzes the candidate solution in search of domain-area pairs with local ratio constraint violations and attempts at relocating traffic among allowed PoPs until satisfaction. This method is usually quick, although not guaranteed to terminate, so a maximum number of fixing attempts is implemented.

c) ***Greedy Deficit - Affinity-Based Algorithm (GDAA):*** Implements a greedy allocation method that accounts directly for traffic percentiles and introduces the concept of *deficits* to track unsatisfied constraints. On each iteration, GDAA allocates the domain-area pair with the largest positive $\gamma-$deficit $= \max\left(0, \gamma_j - \sum_{i \in \mathcal{N}_G(j)} x_{ji} \cdot \ell_{ji}\right)$, i.e., among those with unsatisfied local ratio constraint. If there are no pairs with positive γ-deficit left, it selects the pair with the highest 95th percentile remaining unallocated demand.

Next, the algorithm chooses a PoP among local PoPs first. Since θ_i is the pre-paid minimum bandwidth of PoP i, the algorithm prioritizes those PoPs observing positive $\theta-$deficit $= \max(0, \theta_i - Q_{95}(\mathbf{P}_i))$. If there is no local PoP with a positive θ-deficit, then the selection is made among all allowed PoPs with θ-deficit. Otherwise, GDAA chooses the PoP with the highest *affinity score*. For two same-length time series y and z, the affinity score is defined as:

$$\text{affinity}(y, z) = Q_{95}(y) + Q_{95}(z) - Q_{95}(y + z) \tag{10}$$

Typically, y stands for the current PoP traffic p_{it} and z for a pair's unallocated demand pattern d_{jt} multiplied by a scalar by $\alpha_{ji} \in [0,1]$ used to maximize the affinity. Therefore, the PoP chosen to allocate pair j results from:

$$\begin{aligned} \arg\max_{i} \quad & \text{affinity}(\mathbf{P}_i, \alpha_{ji}\mathbf{D}_j) \\ \text{s.t.:} \quad & p_{it} + \alpha_{ji} d_{jt} \leq b_i \quad \forall t \in [1,T] \end{aligned} \tag{11}$$

Finally, the algorithm updates all related variables: fractional allocation x_{ji} of pair j into PoP i matrix, remaining demand $\mathbf{D}_j \leftarrow (1-\alpha_{ji})\mathbf{D}_j$ of pair j, current throughput $\mathbf{P}_i$ of PoP i and the deficits. The loop terminates when all pairs have been served or no more demand can be allocated.

*d) **Randomized GDAA (RGDAA):*** Similar to GDAA, RGDAA differs in that it selects pairs to allocate in a randomized order rather than in descending order of the 95th percentile of the pair demand traffic. In this fashion, a certain amount of solution space exploration is introduced.

*e) **Max-Based GDAA (GDMAA):*** GDMAA differs from GDAA in that instead of the affinity criterion, the PoP with the largest remaining bandwidth is chosen for allocation. The fraction of traffic is computed to be the largest allowable.

*f) **Uniform Balancing (UB):*** Begins by evenly spreading traffic from all pairs to allowed PoPs. Next, bandwidth-exceeding traffic is relocated using a different adjustment method on each iteration: scaling down by a factor (straightforward traffic rebalance), subtracting a small value (to induce sparser matrices), or adding random perturbations (for further exploration). Pairs with positive γ-deficit are further adjusted by concentrating traffic into local PoPs. The algorithm ends upon finding a feasible solution or reaching the iteration limit.

The rationale is that with sufficient bandwidth and no forbidden allocations, UB converges to a solution in which each PoP gets the same domain distribution. Such a solution is cost-wise indistinguishable from the single-PoP solution, and serves as a performance anchor. Nevertheless, UB's solution is usually too dense to be usable.

*g) **Genetic Algorithm (GA):*** Implements a tailored genetic algorithm to enhance the allocation matrix through iterative refinement. Inspired by the mechanism of natural selection, genetic algorithms are powerful randomized methods for solving search and optimization problems [7], where the best individuals are chosen to reproduce. Each individual in the population corresponds to a candidate allocation matrix $\mathbf{X}$, and a gene is a cell x_{ji} in $\mathbf{X}$. The population of potential solutions evolves through multiple iterations towards improved solutions to the optimization problem.

Baiji randomly initializes the population including both feasible and unfeasible solutions. For diversity and sparsity purposes, roughly half the individuals contain genes in the range $[0,1]$, and the rest take binary genes in $\{0,1\}$ (whole allocation solutions). The initial population includes the solutions obtained by the other algorithms.

The GA terminates after a given number of iterations (*generations*) or if the solution does not improve for 200 generations. On each iteration, the GA's process can be roughly divided into four main functionalities:

Fitness computation: The fitness function $\phi(X)$ is used to evaluate the quality of an individual solution X. If a candidate solution is feasible, its fitness matches its objective function value in Eq. 1. Otherwise, its fitness value relates to its violations' margin. So, if all constraints in the model are expressed as $g_k(X) \geq 0 \, \forall k \in [1,K]$, the fitness can be computed as:

$$\phi(X) = \begin{cases} r(X), & \text{if X is feasible,} \\ \sum_{k=1}^{K} \min(0, g_k(X)) & \text{otherwise} \end{cases} \tag{12}$$

Therefore, unfeasible candidates get negative fitness values, while feasible ones tend to get positive values if they are sparse enough. The fitness of an unfeasible candidate indicates the extent to which it failed to meet the constraints, which helps the GA learn as it explores the solution space.

Selection operation: At the beginning of each iteration, *Baiji* selects the individuals within the top 20% fitness value and those within the top 5% among all unfeasible candidates for reproduction. These individuals also survive into the next generation (*elitist strategy*), guaranteeing that the final solution from the algorithm is at least as good as the initial population's best one. Keeping unfeasible candidates in the reproductive pool preserves population diversity and provides a reference point for the GA to probe constraint boundaries, where solutions to optimization problems often lie.

Crossover operation: On each generation, new candidates (*offspring*) are created by combining genes from two parents into a new individual. The population size is kept constant. *Baiji* implements four crossover mechanisms in a round-robin fashion across generations. These mechanisms stem from two parent-selection methods and two gene-mixing processes. Parents are selected randomly, either with uniform probability or probability proportional to their fitness value. The former method benefits exploration, and the latter exploitation. The gene mixing process interleaves a random linear interpolation and binary gene swapping. The former benefits exploitation on the premise of a locally convex function but hinders solutions sparsity. The latter explores a wider range of possibilities.

Mutation operation: Random Gaussian noise is added to each new individual's genes to maintain exploration capabilities. The noise intensity decreases exponentially while the highest fitness value keeps increasing and stays fixed otherwise. If the population's average gene-wide variance becomes smaller than a predefined fraction of its original value, the noise intensity is reset to foster further exploration. Sparsity is encouraged by zeroing all genes lower than 0.01.

B. System details.

a) Trade-offs: The algorithms in *Baiji* provide different trade-offs between computational complexity (speed) and accuracy. Greedy algorithms, particularly those working with scalar values (e.g., GFPR) are notably fast but may lack precision in some scenarios due to their simplistic assumptions. Conversely, more complex and powerful algorithms (e.g., GA) require greater computation time for more refined solutions.

	Real-world	***SYN1***	***SYN2***
Domains	267	450	240
Areas	30	30	30
PoPs	140	120	100
Pairs	6,140	11,475	6,120
Timeslots	8,879	8,000	8,000
Allowed options	145,457	228,761	101,843

TABLE I
CHARACTERISTICS OF THE DATASETS USED FOR EVALUATION.

b) Robustness: Baiji consider the traffic patterns to be exact. In practice, they usually are obtained as predictions subject to errors with respect to actual traffic demand patterns, which may affect the performance of *Baiji*. In principle, drifts in traffic values should be smoothly handled by *Baiji* (with a slight performance drop) as long as the temporal location of the top 5% traffic load time slots does not change. On the other hand, shifting time slot positions could result in affinity mismatches. Nevertheless, if a large deviation is detected, *Baiji* can be re-executed multiple times during the month to adjust to changes. Robustness is an important concern that is planned to be carefully addressed in future work.

IV. Performance Evaluation

In this section, we conduct experiments to evaluate Baiji's performance in three different scenarios. The first is a real-world scenario provided by a large CDN collaborator. The others are synthetic scenarios with similar characteristics to the real-world one, although with different variations. Each dataset provides per domain time series for a whole month. Table I shows the main characteristics of each dataset.

We implement all the related code in Python 3.10. All experiments were executed using a single instance of a 64 GB RAM virtual server running Ubuntu 22.04.2 LTS.

A. Real-world dataset description

The real-world dataset includes 140 PoPs with bandwidths ranging between 530 Gbps and 1 Gbps (2 orders of magnitude in difference). Minimum bandwidth agreements (θ parameters) range from zero to the PoP's full bandwidth. The aggregated system bandwidth and pre-agreed paid traffic are 26,488 Gbps and 9,192 Gbps respectively. Non-zero γ parameters for locality constraints (Eq.5) for some pairs are also provided, as well as forbidden pair-to-PoP allocations.

The dataset details a full month's demand time series patterns (of 5-minute time slots) for each of the 6,140 domain-area pairs. The aggregated demand time series statistics include: a maximum of 13,281 Gbps, a 95th percentile of 6,507 Gbps, a 75th percentile of 5,789 Gbps, a median of 4,880 Gbps, and a 25th percentile of 3,759 Gbps.

There are large differences concerning the traffic for each domain-area pair. Fig. 3 shows a cactus-plot with the mean and 95th percentile traffic for all pairs in ascending order. On average, there are more than 8 orders of magnitude of difference between the most and least demanded pairs. The curve is smooth and concave for the initial pairs and hits an

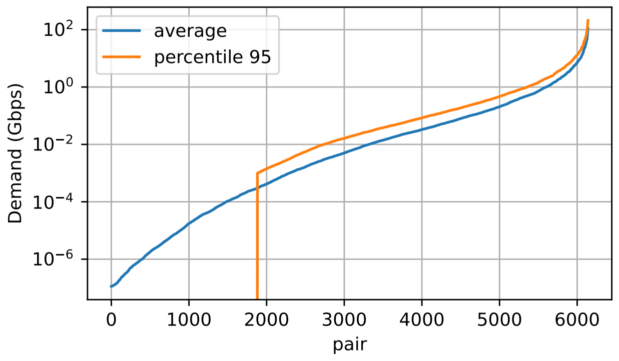


Fig. 3. Traffic demand of time's average and 95th percentile values for each domain-area pair on the real-world dataset, sorted ascendingly. The plot uses a semi-logarithmic y-axis.

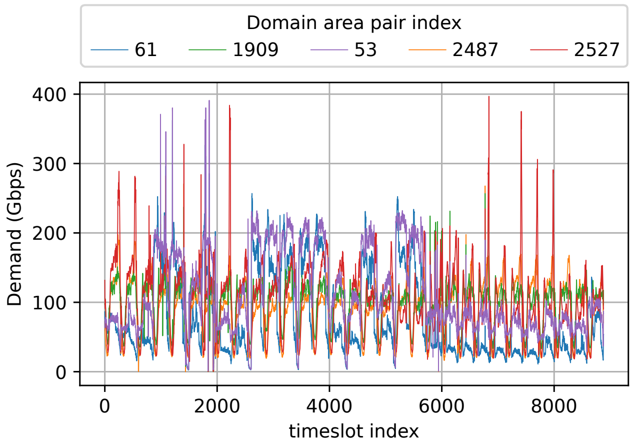


Fig. 4. **Monthly traffic patterns of the top five time-averaged demand domain-area pairs on the real-world dataset.** The legend indicates each pair's index number on the dataset. they account for more than 10% of the total average traffic demand.

inflection point around two-thirds of the pairs. The top 1% of pairs account for half of the total (average) traffic.

Similarly, the 95th traffic percentile also spans multiple orders of magnitude. Noticeably, nearly 30% of the pairs are so rarely requested that their 95th traffic percentile is 0 Gbps. While these pairs could potentially facilitate cost-effective distributed allocations, they collectively represent only 0.014% of the total traffic. This limited volume leaves little room for algorithms to exploit their presence.

To build intuition about the temporal behavior of the most dominant pairs, Fig. 4 shows the monthly traffic pattern of the top five most demanded pairs. Aggregated, they account for more than 10% of the total average traffic.

Daily and weekly patterns can be seen for all pairs, although the demand is uneven from one week to the next. Remarkably, there is a ***high correlation*** between any two traffic traces. Intuitively, this fact makes it ***harder to allocate*** pairs in a way that can reduce the total system cost.

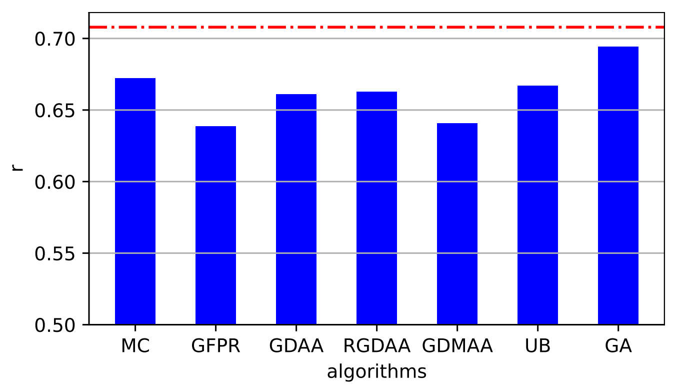


Fig. 5. **Without penalization, GA performs better than other algorithms** The figure shows non-penalized performance results for algorithms on the real-world dataset. The red dashed line indicates the theoretical upper bound. GA performs closest to this bound, while GFPR and GDMAA perform the worst. Other algorithms fall in between.

Algorithm	Elapsed Time	Comments
MC	5,467 s	10,000 candidates
GFPR	0.71 s	
GDAA	23 s	
RGDAA	2,315 s	200 sortings
GDMAA	17 s	
UB	0.56 s	
GA	14,753 s	2,000 generations, 500 candidates

TABLE II
ELAPSED EXECUTION TIMES FOR *Baiji*'S ALGORITHMS.

B. Results on real-world dataset

Performance. Fig. 5 displays the non-penalized performances of the solutions found by the algorithms on the real-world dataset, which hover around 0.65 while the upper bound is 0.71. Algorithms GFPR and GDMAA, the simplest algorithms, show the lowest performances. MC, GDAA, RGDAA and UB yield different solutions of similar performance, proving the viability of achieving ***multiple solutions*** for the same dataset. ***GA surpasses the other solutions***, achieving a performance that is less than half as close to the upper bound compared to the solution produced by the next best algorithm.

Density and sparsity. Fig. 6 shows the density of solutions, quantified by the percentage of non-zero traffic fraction entries among all allowed allocations. Solutions with densities near 100%, such as those from MC and UB, violate the cache-hit-ratio requirement and are deemed unusable. Conversely, GFPR, GDAA, RGDAA, and GDMAA achieve very sparse solutions, close to the theoretical minimum. Finally, GA achieves a density below 10%, roughly double the previous set of algorithms. This density value is deemed as a reasonable ***compromise value between sparsity and performance***.

Execution times. The execution times for the algorithms are shown in Table II. GFPR and UB were the fastest by two orders of magnitude (under one second), although the former tends to find low-performance and the latter too dense solutions. GDAA and GDMAA have running times in the tens of seconds while providing better solution candidates. Randomized solutions MC and RGDAA are slower (under 90 minutes) as they require many attempts to find solutions. Their execution times can be controlled by adjusting the number of random instances. Finally, GA exhibits the slowest execution time (around four hours) yet it ensures a solution no worse than the others. Its runtime scales linearly with the number of generations and super-linearly with population size. ***All the execution times are acceptable for monthly planning*** (between seconds and a few hours).

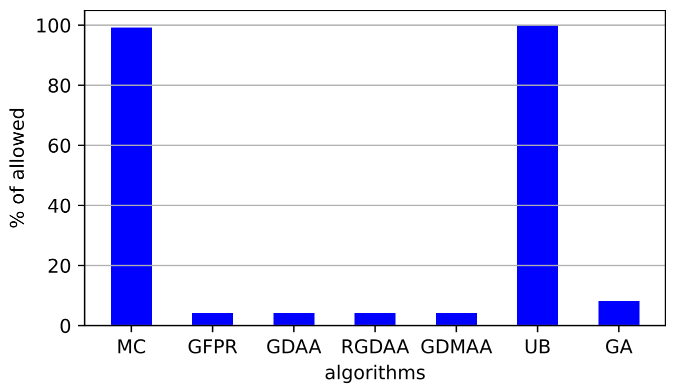


Fig. 6. **MC and UB algorithms are less sparse than others.** This figure shows the solution densities of each algorithm on the real-world dataset. The y-axis indicates the percentage of utilized domain-area pair to PoP allocations among all allowed options. MC and UB have insufficiently low cache-hit ratios. GA strikes a reasonable balance between sparsity and performance.

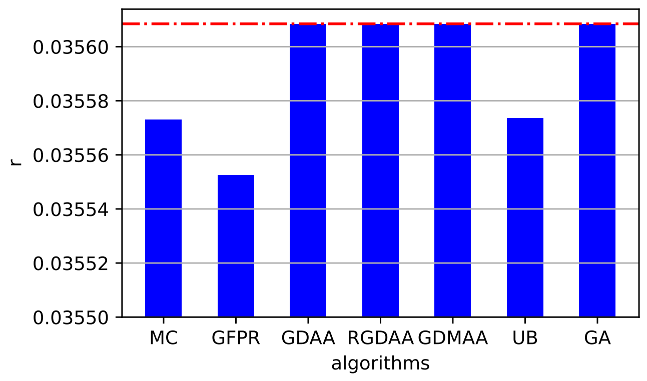
C. Results with synthetic workloads

We create two synthetic datasets (*SYN1* and *SYN2*) with similar characteristics to the real-world dataset (see Table I) but with specific differences to evaluate the performance and robustness of algorithms more comprehensively.

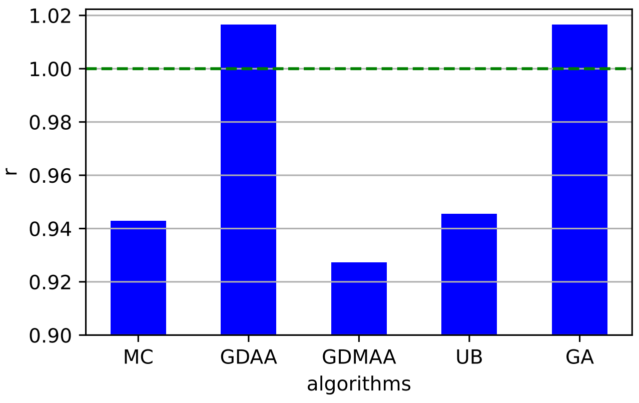
Both synthetic datasets have committed-rate distributions similar to the real-world dataset and a flatter PoP bandwidth distribution (although with same median). The total bandwidth across all datasets remains the same. *SYN1* contains approximately 68% more domains, 87% more domain area pairs, and 57% more allowed allocation options than the real-world dataset. The demands' time series are generated as sine curves with amplitudes one-tenth the size of the real-world dataset's traffic patterns, daily periods, and uniformly distributed phases. In such a scenario, the committed rates are dominant (the upper bound becomes 0.03561) and are expected to dominate the algorithms' performances. Moreover, the random phases make it challenging for the algorithms to identify usable affinities among the patterns.

Fig. IV-B depicts the performance of all algorithms on *SYN1*. GDAA, RGDAA, GDMAA, and GA are less than 0.0002 % away from the upper bound, virtually matching it. The worst performance is achieved by GFPR, just 0.15 % less than the bound. This result shows that ***all algorithms have the potential to solve the problem***, at least in some instances.

The greatest difference between *SYN2* and the real-world dataset is that *SYN2*'s demand patterns are pulses with *on-off* shapes, with exactly zero demand for 96% of the time, with contiguous and constant *ON* mode traffic. The pulse



(a) ***SYN1*: All algorithms approach the theoretical upper bound**. Dataset *SYN1* is an instance with low demands with respect to bandwidths.



(b) ***SYN2*: GDAA and GA demonstrate better performance than the single-PoP solution** ($r > 1.0$**, dashed green line**). Dataset *SYN2* has pulse-shaped demand patterns.

Fig. 7. Non-penalized performance results for finished algorithms on synthetic datasets. The red dashed line indicates the theoretical upper bound.

location on each pattern is generated at random. Placing all pairs together in the single-PoP solution yields a cost twice as large as the original dataset's. The upper performance bound becomes larger than 1, meaning that outperforming the single-PoP solution ($r = 1$) is theoretically possible.

On *SYN2*, as depicted in Fig. IV-B, algorithms GDAA and GA achieve solutions ***outperforming a single-PoP solution*** approach. This fact indicates cost-saving potential for the CDN by carefully planning a geographically distributed deployment.

Across both real-world and synthetic datasets, ***algorithmic densities and execution times remain consistent***, reaffirming conclusions drawn from the real-world dataset analysis.

The results show that ***Baiji can provide high-quality solutions*** to the CDN planning problem by approaching the upper bound. Furthermore, all algorithms are useful beyond their merits, as they contribute alternative solutions that later ***nourish the GA with different insights***. In *Baiji*, the whole is greater than the sum of its parts.

V. Related Work

Resource allocation problems in CDNs have received much attention already, covering several important aspects such as bandwidth limits, geographic distribution, and other constraints [10], [11]. Chen et al. [5] study online traffic allocation in CDNs to minimize the bandwidth cost. They propose OnTPC, a prediction-based algorithm addressing constraints such as allocation granularity and deviations raised in practical deployment. Following this, Zhan et al. [9] propose Iris for online midgress-sensitive traffic allocation in CDNs. It optimizes edge and midgress costs while accounting for traffic dynamics and allocation granularity. The edge cost is similar to the cost in our work, but they do not consider per-domain locality-based and cache-hit-ratio requirements.

There are some similar cases of resource optimization in the cloud and virtual networks. Hu et al. [13] address the joint problem of resource provisioning and content caching in cloud-based CDNs and design DPC, a two-step algorithm framework to minimize the total rental cost from cloud vendors while satisfying all demands. Rankothge et al. [14] consider the initial provisioning and the dynamic scaling of virtualized networks and propose a genetic algorithm for cloud resource allocation, achieving good scalability and efficiency. However, most of the algorithms mentioned above do not take into account real constraints such as the distance between different PoPs and domain-area pairs, PoP's committed rates, and local traffic quotas, as *Baiji* does.

The traffic rates and charging plans agreed upon between ISPs and their users determine the traffic transmission costs. Most ISPs charge for traffic to all destinations at a blended rate [12]. Notably, transmission costs for billing plans like the q-th percentile billing model do not affect the $(100 - q)$ time slots with the highest data transfers. TrafficShaper [4] leverages this fact, scheduling inter-data center traffic peaks during "cost-free" time slots, and balancing the rest of the time. It assumes the data sizes and the flows' deadlines are independent and identically distributed (i.i.d). This differs from our work. In addition, geographic distribution is an important factor we consider in *Baiji*, but TrafficShaper ignores it.

Besides the 95th percentile billing model, an ISP can use other charging schemes for bandwidth usage, such as maximum (MAX) or average (AVG). Adler et al. [8] study offline and online algorithms to minimize bandwidth costs under MAX, AVG, and 95th percentile billing contracts. While their method is theoretically sound, it lacks empirical validation and lacks consideration of practical constraints as discussed in Section II. Notably, the study proves the NP-hard nature of optimizing costs under the 95th percentile billing model, which is not the case for MAX and AVG schemes.

The application of machine learning in network resource allocation problems is expanding. Aibin [15] proposes a Long-Short Term Memory method to predict network traffic patterns and improve the efficiency of resource allocation in an optical network. Instead, *Baiji* takes traffic patterns as inputs, although

Work	Offline CDN traffic cost optimization	Assignments of domains to PoPs	95th percentile billing	PoP's bandwidth constraints	PoP's committed rates	Geographical / Locality-based constraints	Cache-hit-ratio considered	Balanced system complexity and scale
[4]			✓					✓
[8]	✓		✓		✓			
[5], [9]			✓	✓	✓			
[10], [11]	✓			✓	✓	✓		
[12]			✓					
[13]				✓	✓	✓		
[14], [15]								✓
Baiji (ours)	✓	✓	✓	✓	✓	✓	✓	✓

TABLE III
CHARACTERISTICS OF RELATED WORKS.

Aibin's technique could be used to forecast them. Moreover, *Baiji*'s allocation follows more complex allocation criteria than Aibin's. More recently, Fang et al [16] propose a deep reinforcement learning-based content distribution and caching scheme to minimize latency in next-generation wireless networks. Unlike *Baiji*, their problem is online, and their linear cost function is simpler.

While studies exist on CDN optimization and domain planning within the 95th percentile billing model, to the best of our knowledge, there is no prior research addressing domain to multi-PoP planning considering 95th percentile billing, distance, committed rates, and cache-hit-ratio constraints.

Table III summarizes various approaches in the literature. [14]–[16] apply machine learning techniques or evolutionary algorithms to address resource allocation and prediction, with [14], [15] acknowledging implementation complexity. Similarly, [4] addresses the complexity and the 95th percentile billing model. Moreover, [5], [9] consider PoPs' bandwidths and committed rates alongside the same billing model. Constraints in [10], [11], [13] include PoP's bandwidth and geographical factors. In contrast, *Baiji* integrates these constraints while also considering minimum per-domain cache-hit-ratios, enabling flexible assignments of domains to PoPs.

VI. CONCLUSIONS AND FUTURE WORK

We initiated the study of a novel domain to multi-PoP flexible assignment problem in CDNs under the 95th percentile billing model and realistic capacity, locality and hit-ratio constraints. We formulated a mathematical model for the problem and derived a natural performance upper bound.

We presented several heuristic algorithms for finding candidate solutions and integrated them into our multi-algorithm approach *Baiji*. In particular, *Baiji* uses a specially tailored genetic algorithm to improve the initial solutions further.

The evaluation on a real-world dataset from a large CDN provider shows that *Baiji* can effectively reduce transmission costs paid and approach the upper bound. On a synthetic dataset *Baiji* matched the upper bound, and on another, it managed to outperform a challenging single-PoP abstraction.

In future work, we plan to explore the robustness of the prediction against errors on the demand prediction, evaluate more datasets and consider extending the multi-algorithm approach to other aspects of the CDN network, such as routing, caching, and load-balancing. Another interesting research direction concerns the security aspects of such architectures.

REFERENCES

[1] Sandvine, "2023 global internet phenomena report," online, Jan 2023, accesed on 18 Jun 2023.

[2] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The Growing Complexity of Content Delivery Networks: Challenges and Implications for the Internet Ecosystem," *Telecommunications Policy Journal*, vol. 41, no. 10, pp. 1003–1016, November 2017.

[3] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, p. 52–66, jul 2015.

[4] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "Trafficshaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1193–1206, 2018.

[5] H. Chen *et al.*, "Online Traffic Allocation Based on Percentile Charging for Practical CDNs," in *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, Oslo, Norway, 2022, pp. 1–10.

[6] P. e. a. Virtanen, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[7] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[8] M. Adler, R. K. Sitaraman, and H. Venkataramani, "Algorithms for optimizing the bandwidth cost of content delivery," *Computer Networks*, vol. 55, no. 18, pp. 4007–4020, 2011, internet-based Content Delivery.

[9] H. Zhan *et al.*, "Online Midgress-Sensitive Traffic Allocation for Percentile Charging in Practical CDNs," in *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, Orlando, FL, USA, 2023, pp. 1–10.

[10] G. Tang, H. Wang, K. Wu, and D. Guo, "Tapping the knowledge of dynamic traffic demands for optimal cdn design," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 98–111, 2019.

[11] G. Tang, K. Wu, and R. Brunner, "Rethinking cdn design with distributee time-varying traffic demands," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[12] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, and V. V. Vazirani, "How many tiers? pricing in the internet transit market," in *Proceedings of the ACM SIGCOMM 2011 Conference*. New York, NY, USA: Association for Computing Machinery, 2011, p. 194–205.

[13] M. Hu, J. Luo, Y. Wang, and B. Veeravalli, "Practical Resource Provisioning and Caching with Dynamic Resilience for Cloud-Based Content Distribution Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2169–2179, Aug 2014.

[14] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 343–356, June 2017.

[15] M. Aibin, "LSTM for cloud data centers resource allocation in software-defined optical networks," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2020, pp. 0162–0167.

[16] C. Fang, H. Xu, Y. Yang, Z. Hu, S. Tu, K. Ota, Z. Yang, M. Dong, Z. Han, F. R. Yu, and Y. Liu, "Deep-reinforcement-learning-based resource allocation for content distribution in fog radio access networks," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 16 874–16 883, 2022.