A Scalable Color-Based Caching Scheme in Telco-CDNs

Anh-Tu Ngoc Tran*, Thanh-Dang Diep*, Takuma Nakajima†, Masato Yoshimi‡ and Nam Thoai*
*High Performance Computing Laboratory, Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology, VNUHCM, Vietnam
Email: {51304672, dang, namthoai}@hcmut.edu.vn
†Service Strategy Sector, ‡Strategic Technology Center,
TIS Inc., Shinjuku-ku, Tokyo, Japan
Email: {nakajima.takuma, yoshimi.masato}@tis.co.jp

***Abstract*—Internet traffic is growing quickly, and it is majorly contributed by the proliferation of video services. Content Delivery Networks (CDNs) reduce the video traffic by storing replicas of videos in their cache servers. Nonetheless, the cache servers are usually located outside Internet Service Providers (ISPs). This implies that CDNs cannot reduce the video traffic inside ISP networks. To mitigate this issue, many ISPs build their own CDNs called Telco-CDNs. Genetic Algorithm-based caching is deemed the best approach in terms of traffic reduction. However, it is not practical since its computation time to generate content allocations is extremely long even when using a cluster. A color-based approach was devised to help overcome the drawback at the expense of its increase in traffic. Nevertheless, in case the number of content categories or requests proliferates quickly, the approach also has the same limitation like the Genetic Algorithm-based caching. To resolve the limitation, we propose two novel techniques to hamper the increase in the computation time. One is able to cope with the situation when the number of content categories increases while the other can deal with the circumstance when the number of requests rises. The empirical results show that the computation time is reduced 5x for the former and 7x for the latter at the expense of 1% and 12% increase in traffic for a problem of 5,000 contents, respectively.**

***Index Terms*—color-based caching, scalable, telco-CDNs, time complexity**

I. Introduction

A previous study [1] showed Internet traffic will increase 3x in the next 5 years, and, by 2021, 82% of its is caused by Video-on-Demand (VoD) services. Such enormous video traffic will cause congestion, which degrades network performance as well as user experience. To lessen the problem, CDN providers place cache servers in multiple locations [2]. Although CDNs can reduce the video traffic, they are usually located outside ISPs [3], which leads to the fact that CDNs cannot decrease the traffic inside networks of ISPs as well as between a pair of ISPs. Even if few CDN providers sought to build some cache servers at some restricted areas inside ISP networks [4]–[6], the network traffic cannot be reduced considerably [3]. The reason is that CDN providers do not consider the underlying network infrastructure or they have no global knowledge about it which is private information of ISPs. Some ISPs deploy their own CDNs which are called Telco-CDNs. Caching in a Telco-CDN is far different from a CDN because the objective of a conventional CDN is to optimize the performance of an overlay network while the goal of a Telco-CDN is to optimize the performance of both the overlay and the underlying infrastructure [3]. The management of a Telco-CDN is regarded as a scientific problem of growing importance [7]–[10]. Taking full advantage of the network infrastructure factors in optimizing the caching will make its algorithm more complicated, but that is entirely feasible [3] because the number of deployed repositories is less than a hundred for a typical Telco-CDN in a national ISP [11], which cannot be compared to the 100,000 servers of Akamai [12]. Yet, the optimal mapping between contents and cache servers is very problematic because the kind of such problems is deemed NP-complete [3]. Because an inefficient mapping can make the network traffic increase, the creation of an efficient caching scheme is non-trivial.

Li et al. [3] proposed a content allocation algorithm based on Genetic Algorithm (GA) to address this problem. The algorithm always gives a sub-optimal solution in the context of the network traffic. However, the computation time to output the solution is extremely long even when using a cluster. Since 20% to 60% of content popularity dramatically varies every hour due to the insertion of new contents and influential news [13], [14], the long computation time can cause mismatches in the content allocations, which produces additional traffic. Hence, the approach is totally impracticable in reality.

To overcome the limitation of the GA, Nakajima et al. [15], [16] devised a light-weight color-based caching scheme using co-operative caching [3], [17]–[19] and hybrid caching [20]. The scheme can reduce the computation time while preserving an approximate solution compared with the GA in terms of the network traffic. However, it can only keep the computation time acceptable when the number of content categories is 1,000. Its algorithm complexity depends on the number of contents and the number of users' requests which are factors varying arbitrarily. Hence, the issue regarding the computation time still has not been solved completely.

In this paper, we propose two novel techniques to make the computation time of the color-based approach more acceptable at the expense of a reasonable amount of increase in network

978-3-903176-24-9

TABLE I
NOTATIONS

k	The number of steps SRs algorithm runs
C	Cache capacity of a cache server
N	The number of colors
$\#contents$	The number of contents
$\#requests$	The number of requests
T_{est}	Traffic cost
$Time(T_{est})$	Time to estimate traffic

traffic. One is to increase step size when the number of contents increases while the other is to reduce the simulated request set when the number of requests increases.

In the next section, we first give a general description of the color-based caching scheme. Section III presents two novel algorithms to help it scale well when the number of contents or requests dramatically rises. Section IV proves the effectiveness of the proposed algorithms based on empirical findings, and Section V surveys the state of the art and related work. Finally, we conclude in Section VI.

II. BACKGROUND

A. An Overview of a Color-based Caching Scheme

The color-based caching algorithm proposed by Nakajima et al. [15], [16] was thoroughly recapitulated in the work of Tran et al. [21], [22]. The idea of this algorithm is based on colors. Specifically, both cache servers and contents are colorized, and servers only cache contents if they have the same colors. Moreover, the color-based caching scheme uses a hybrid caching scheme to follow changes in content popularity. Besides, the author also proposed a color-based routing strategy to further reduce traffic rather than using the traditional shortest path route (SPR).

B. Normal Separator Ranks Algorithm - NSR

Content colorization happens every interval to adapt to quick changes in content popularity. Based on previously gathered access logs, separator ranks (SRs) algorithm is run to decide how many colors each content should have so that traffic is minimal in a network. In this paper, we refer to the original SRs algorithm in [16] as normal SRs algorithm or NSR. Before further discussing and analyzing NSR, notations are first described in Table I.

For instance, assuming that N = 4 colors are used, and each cache server can store C = 100 contents, to colorize contents, a set of numbers a, b, c, d called SRs are used to categorize contents to corresponding popularity classes (Table II). Each set of SRs has a different traffic cost, and NSR will try to find the optimal set that results in minimal traffic and satisfies two constraints. The first one

$$a \leq b \leq c \leq d \tag{1}$$

indicates that SRs of more popular classes are less than or equal to the ones of less popular classes. The second one

$$a + b + c + d = N * C = 4 * 100 \tag{2}$$

states that the total cached contents in the network does not overflow the total storage capacity of all cache servers.

Next, to further understand NSR, the time complexity of it will be analyzed. Besides, the analysis works under two assumptions: (1) $C = 10\% * \#contents$ and (2) $\#requests$ is linearly proportional to $\#contents$. Generally, the time complexity of NSR is $k *$ the number of sets of SRs evaluated in each step $* Time(T_{est})$, in which

- In each step, approximately $N * C$ sets of SRs will be evaluated for the worst case.
- In each cache server, aggregate time to do Insert, Access, Evict operations for each request is $O(C)$. Therefore, time to estimate traffic $Time(T_{est}) = O(\#requests * C)$.

Hence, the time complexity of normal SRs is

$$\begin{aligned} & k * N * C * O(\#requests * C) \\ = {} & k * N * O(\#requests * C^2). \end{aligned}$$

Moreover, the number of colors (N) remains unchanged once the caching scheme is running. NSR is a heuristic algorithm like GA, and the number of steps (k) it runs is non-deterministic. To simplify the time complexity analysis, the number of steps (k) that NSR runs is considered a constant. Therefore, the worst-case running time of NSR is now

$$\begin{aligned} & O(\#requests * C^2) \\ = {} & O(\#contents * (10\% * \#contents)^2) \\ & (Assumption\ (1)\ and\ (2)) \\ = {} & O(\#^3 contents)). \end{aligned}$$

TABLE II
POPULARITY CLASSES

Content Rank	Popularity Class
From 1 to a	High (4 colors)
From a+1 to b	Mid-high (3 colors)
From b+1 to c	Middle (2 colors)
From c+1 to d	Mid-low (1 color)
From d+1 to $\#contents$	Low (0 color)

TABLE III
AN EXAMPLE OF STEP SIZE = 3

a	0	0	0	0		0	0
b	0	0	0	0		0	0
c	**0**	**3**	**6**	**9**	. . .	**198**	**201**
d	400	397	394	391		202	199
T_{est}	T_1	T_2	T_3	T_4		T_x	*violate condition* (1)

III. ALGORITHMS

To make NSR algorithm scalable when $\#requests$ and $\#contents$ are large, two techniques are proposed. The quality of solutions is reduced to trade off for shorter execution time.

A. A modified separator ranks algorithm - ISR

The idea of the first technique is to try to bring a bigger problem to a base case, which is a problem with 1,000 contents and can be solved in a reasonable amount of time. Specifically, when $\#contents$ increases X times compared to the base case, step size to find separator ranks in NSR algorithm increases X times. For instance, in case of 4 colors, when $\#contents$ increases 3 times, step size increases 3 times (Table III). Hereafter, this algorithm is referred as increase SRs or ISR.

When $\#contents$ increases X times, step size increases X times. As a result, the number of sets of SRs evaluated for traffic reduces X times (1). Besides, since $C = 10\% * \#contents$,

cache size also increases X times (2). From (1) and (2), we can conclude that the number of sets of SRs

$$\frac{k * N * C * X}{X} = k * N * C$$

is constant. Hence, the worst-case running time of ISR is

$$\begin{aligned} & k * N * C * Time(T_{est}) \\ &= k * N * C * O(\#requests * C) \\ &= O(\#requests * C) \\ &= O(\#contents * 10\% * \#contents) \\ &\quad (Assumption\ (1)\ and\ (2)) \\ &= O(\#^2 contents)). \end{aligned}$$

B. Reduce time to estimate traffic using a reservoir algorithm

The second technique aims to reduce execution time in $Time(T_{est})$. Traffic is estimated using a set of requests gathered in the previous interval. However, the set of requests is enormous; therefore, it will take a lot of time to compute T_{est}. Instead of using the original set, traffic can be estimated on a small sample set of requests with the same characteristic as the original by using a reservoir algorithm [23]. Reservoir algorithms are characterized as sampling methods that select without replacement a random sample of size n from a set of size $\mathcal{N}$ in one pass, where $\mathcal{N}$ is unknown beforehand.

As we know, $Time(T_{est})$ is equal to $\#requests$ multiplied by the amount of time to get a content from cache servers. Therefore, when a set of requests are decreased X times in numbers , $Time(T_{est})$ decreases X times. Additionally, the running time of NSR or ISR is $k * N * C * Time(T_{est})$. As a result, the runtime of NSR or ISR decreases X times, when a set of requests is decreased X times.

IV. Evaluation

In this section, we will compare NSR, ISR, and NSR when combined with a reservoir algorithm, namely normal SRs - reservoir (NSR-R), in terms of execution time and traffic.

A. Experimental environment

Experiments are performed on an NTT-like network topology of a Telco-CDN in Japan; as shown in Fig. 1, it consists of 55 cache servers and 1 origin server. Moreover, each cache server has a representative client that generates requests using Gamma distribution, which generates more realistic content access than other distributions (e.g., Zipf and Weibull distribution) [24], and we assume that the number of requests for each server is equal to the number of contents. Therefore, the total number of requests is a product of the number of contents and cache servers. Storage capacity of each cache server is also set to 10% $\#contents$. All the calculations are done using a single host with Intel core i5-3470 CPU 3.20 GHz, 4 physical cores, and 8 GB RAM. Table IV recapitulates experimental parameters used in the simulation.

B. Normal SRs - Increase SRs

In Fig. 2, the line chart demonstrates the runtime of NSR and ISR algorithm along the vertical left axis, and the bar chart

TABLE IV
Experimental parameters

Topology	NTT-like network
C (Cache capacity)	10% $\#contents$
$\#cache\ servers$	55
$\#origin\ server$	1
$\#requests = \#cache\ servers * \#contents$	

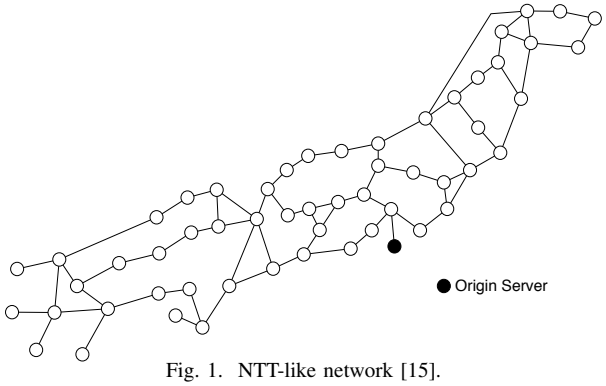

Fig. 1. NTT-like network [15].

represents the speedup ISR over NSR along the vertical right axis as the number of contents varies from 1,000 to 5,000. The experiment confirms the time complexity of those algorithms analyzed in previous sections, which is $O(\#^3 contents)$ for NSR and $O(\#^2 contents)$ for ISR. Besides, a base case of ISR is 1,000 contents, and ISR increases step size X times when $\#contents$ increases X times. Therefore, in case of 1,000 contents, its step size is 1 like NSR, and speedup of ISR over NSR is 1. In addition, when $\#contents$ grows bigger, speedup of ISR is $O(\#contents)$, and the bar chart confirms it.

Next, traffic evaluation of ISR is discussed (Fig. 3). In this experiment, we do not run NSR when $\#contents$ is greater than 5,000 because its runtime is too long. Fig. 3 shows that both NSR and ISR result in better traffic than Perfect LFU [25]. Additionally, solutions found by ISR are as good as ones of NSR as $\#contents$ changes from 1,000 to 5,000, and the quality of solutions of ISR still remains the same when $\#contents$ is greater than 5,000 since the ratio of traffic cost of ISR to Perfect LFU is almost unchanged.

In conclusion, ISR produces solutions equivalently good as NSR, gives at least 26.5% less traffic than Perfect LFU, and runs $O(\#contents)$ times faster than NSR.

C. Normal SRs - Reservoir

In this experiment, NSR algorithm is evaluated when it incorporates a reservoir algorithm to reduce the size of a set

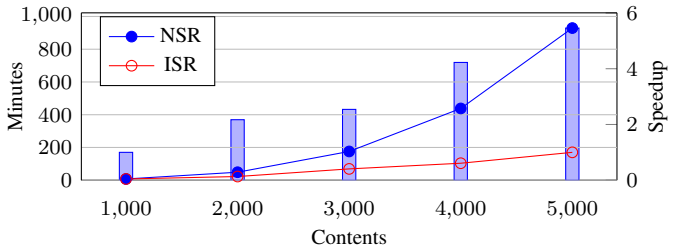

Fig. 2. Execution time of NSR and ISR.

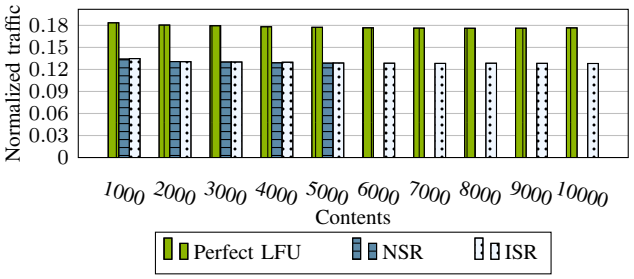

Fig. 3. Traffic evaluation of ISR.

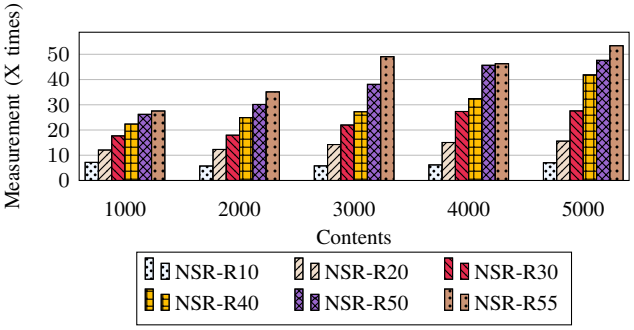

Fig. 4. Relative time of NSR-R when compared to NSR.

of requests used to estimate traffic in a network. Specifically, the set is decreased 10, 20, 30, 40, 50, and 55 times smaller in number. Since there are 55 cache servers, $\#requests = \#cache\ servers * \#contents$, and every content needs at least one request, 55 is the biggest number that a set of requests can be reduced and can still maintain most of the characteristics of the original set of requests.

Fig. 4 presents how many times NSR-R runs faster than NSR. Theoretically, when a set of requests is reduced X times, NSR-R is expected to execute X times more quickly. Experimentally, when $\#contents$ is small, 1,000, NSR-R can only run about X/2 times faster. As $\#contents$ increases, NSR-R can achieve this number. It is because when $\#contents$ increases, which means C (cache capacity) increases ($C = 10\% * \#contents$), reducing the size of a set of requests results in a significant decrease in aggregate time to time to access and evict contents in a cache. In other words, the algorithm will achieve its maximum potential when $\#contents$ is big.

Traffic evaluation of NSR-R is described in Fig. 5. When looking at the picture, for all $\#contents$, NSR and Perfect LFU are considered lower and upper bounds of traffic cost, and NSR using a reservoir algorithm gives solutions that lie between these bounds. In theory, the less a set of requests is reduced in size, the better the traffic cost is produced because it maintains most of the characteristics of the original set of requests. Fig. 5 shows that when a set of requests is reduced from 10 to 55 times, the set reduced 10 times gives the best traffic among those sets reduced.

To sum up, when a set of requests is reduced 10 times, NSR-R can execute 7 times faster NSR in return for approximately 12% increase in traffic.

V. Related Work

There are a few studies related to caching algorithms in the context of Telco-CDNs. Vleeschauwer and Robinson [18]

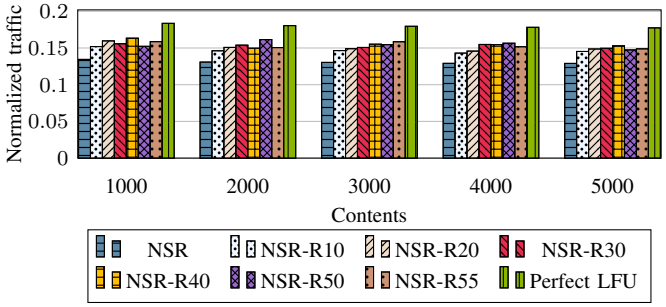

Fig. 5. Traffic evaluation of NSR-R.

proposed an approach encompassing a modified LRU caching algorithm combined one of three cache collaboration strategies. These strategies were developed for various collaborative purposes and based on tree-based networks. However, the approach is difficult to be deployed on most of the ISPs' backbone network [3].

To the best of our knowledge, the GA-based approach [3] is the best effective one in terms of traffic reduction. It will be periodically run to calculate content allocations. However, it takes more than 7 hours when running 1,000 contents on a typical Telco-CDN [15], [16]. Since 20% to 60% of the content popularity fluctuates every hour [13], [14], such long calculation causes mismatches in the content allocations that manufacture additional traffic.

Additionally, there is other work which studied both content distribution and traffic engineering in a joint manner. Sharma et al. [10] empirically showed that unplanned strategies for routing and content replacement such as SPR and LRU outperformed joint optimization strategies which compute the best placement and routing with knowledge from the previous day. Diab et al. [26] preliminarily proposed an algorithm to distribute videos with different bit rates to users in a Telco-CDN after carefully considering traffic paths through the network. In general, our work differs from the aforementioned ones in that we focus solely on content distribution.

VI. Conclusions

In this paper, we present two novel techniques to resolve the limitation of the color-based caching scheme. Both of them aim to make the color-based approach scalable in case the number of content categories or requests rises rapidly. The experimental findings show that the former proposed technique can change the time complexity of the color-based caching scheme from cubic to quadratic when the number of content categories and requests increase while the traffic still remains almost unchanged. Furthermore, the results specify that the latter devised technique can make the execution time of the color-based approach speed up about 7 times faster at the expense of nearly 12% increase in traffic for a problem of 5,000 contents.

Acknowledgment

This research was conducted within the project of Optimizing Color-Based Cooperative Caching Algorithm for Telco-CDNs sponsored by TIS (IT Holding Group).

REFERENCES

[1] V. Cisco, "The zettabyte era: trends and analysis. updated (07/06/2017)," 2017.

[2] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, pp. 52–66, 2015.

[3] Z. Li and G. Simon, "In a telco-cdn, pushing content makes sense," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 300–311, 2013.

[4] N. Inc., "Netflix open connect appliance deployment guide." https://openconnect.netflix.com/deploymentguide.pdf, accessed: May 27, 2019.

[5] A. Technologies, "Faqs: Akamai accelerated network partner (aanp) program." https://www.akamai.com/us/en/multimedia/documents/akamai/akamai-accelerated-network-partner-aanp-faq.pdf, accessed: May 27, 2019.

[6] G. Inc., "Google edge network." https://peering.google.com/\#/infrastructure, accessed: May 27, 2019.

[7] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an isp network," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 239–250.

[8] Z. Li, M. K. Sbai, Y. Hadjadj-Aoul, A. Gravey, D. Alliez, J. Garnier, G. Madec, G. Simon, and K. Singh, "Network friendly video distribution," in *2012 Third International Conference on The Network of the Future (NOF)*. IEEE, 2012, pp. 1–8.

[9] D. Rayburn, "Three new white papers released on the telco cdn space," 2011.

[10] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing content simplifies isp traffic engineering," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 229–242, 2013.

[11] M. Shibuya, Y. Hei, and T. Ogishi, "Evaluation for cost-effective cache deployment in isp networks," in *2012 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. IEEE, 2012, pp. 1–7.

[12] A. Inc., "State of the internet," https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/akamai-q3-2012-state-of-the-internet-connectivity-report.pdf, accessed: May 27, 2019.

[13] H. Yin, X. Liu, F. Qiu, N. Xia, C. Lin, H. Zhang, V. Sekar, and G. Min, "Inside the bird's nest: measurements of large-scale live vod from the 2008 olympics," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. ACM, 2009, pp. 442–455.

[14] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4. ACM, 2006, pp. 333–344.

[15] T. Nakajima, M. Yoshimi, C. Wu, and T. Yoshinaga, "A light-weight content distribution scheme for cooperative caching in telco-cdns," in *2016 Fourth International Symposium on Computing and Networking (CANDAR)*. IEEE, 2016, pp. 126–132.

[16] ——, "Color-based cooperative cache and its routing scheme for telco-cdns," *IEICE TRANSACTIONS on Information and Systems*, vol. 100, no. 12, pp. 2847–2856, 2017.

[17] N. Choi, K. Guan, D. C. Kilper, and G. Atkinson, "In-network caching effect on optimal energy consumption in content-centric networking," in *2012 IEEE international conference on communications (ICC)*. IEEE, 2012, pp. 2889–2894.

[18] D. De Vleeschauwer and D. C. Robinson, "Optimum caching strategies for a telco cdn," *Bell Labs Technical Journal*, vol. 16, no. 2, pp. 115–132, 2011.

[19] Z. Wang, H. Jiang, Y. Sun, J. Li, J. Liu, and E. Dutkiewicz, "A k-coordinated decentralized replica placement algorithm for the ring-based cdn-p2p architecture," in *The IEEE symposium on Computers and Communications*. IEEE, 2010, pp. 811–816.

[20] Y. Zhou, L. Chen, C. Yang, and D. M. Chiu, "Video popularity dynamics and its implication for replication," *IEEE transactions on multimedia*, vol. 17, no. 8, pp. 1273–1285, 2015.

[21] A.-T. N. Tran, M.-T. Nguyen, T.-D. Diep, T. Nakajima, and N. Thoai, "A performance study of color-based cooperative caching in telco-cdns by using real datasets," in *Proceedings of the Ninth International Symposium on Information and Communication Technology*. ACM, 2018, pp. 169–176.

[22] ——, "Optimizing color-based cooperative caching in telco-cdns by using real datasets," in *International Conference on Ubiquitous Information Management and Communication*. Springer, 2019, pp. 288–305.

[23] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[24] X. Cheng, J. Liu, and C. Dale, "Understanding the characteristics of internet short video sharing: A youtube-based measurement study," *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1184–1194, 2013.

[25] G. Einziger, R. Friedman, and B. Manes, "Tinylfu: A highly efficient cache admission policy," *ACM Transactions on Storage (ToS)*, vol. 13, no. 4, p. 35, 2017.

[26] K. Diab and M. Hefeeda, "Cooperative active distribution of videos in telco-cdns," in *Proceedings of the SIGCOMM Posters and Demos*. ACM, 2017, pp. 19–21.