A GRASP-based Approach for Dynamic Cache Resources Placement in Future Networks

Hamza Ben-Ammar, Yassine Hadjadj-Aoul

Noname manuscript No.
(will be inserted by the editor)

A GRASP-based Approach for Dynamic Cache Resources Placement in Future Networks

H. Ben-Ammar · Y. Hadjadj-Aoul

Received: date / Accepted: date

Abstract Dealing with the ever-increasing video traffic is certainly one of the major challenges facing Internet Service Providers (ISPs). In this context, the strategic placement of caches is seen as one of the most important remedies, especially with recent advances in the field of virtualization. Unlike the existing works, which only focus on the placement issue, we also consider the problem of determining the optimal amount of cache to place at each possible location. We formalize, in this paper, the problem of caches placement as a multi-objective optimization problem, in which we minimize both the average distance from which contents are retrieved and the peering links utilization. As the proposed problem is NP-hard, we propose to solve it using the Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic. Simulations results reveal the quality of the obtained solutions compared to an exhaustive search method. At the same time, they reveal that the solution is not to put all resources at the edge or at the core, as some studies claim, but to partition them judiciously, which mainly depends on the objectives of the ISPs.

Keywords Multi-cache systems · Cache allocation · Multi-objective optimization · GRASP · VNF placement.

1 Introduction

The recent rise of Network Function Virtualization (NFV) enables Internet Service Providers (ISPs) to better leverage their infrastructures by enabling

Hamza Ben-Ammar
Univ Rennes, Inria, CNRS, IRISA
E-mail: hamza.ben-ammar@irisa.fr

Yassine Hadjadj-Aoul
Univ Rennes, Inria, CNRS, IRISA
E-mail: hamza.ben-ammar@irisa.fr

more dynamic/agile services' placement, with the ability of on-demand deployment [1].

The expansion of virtualization to the field of content distribution offers new opportunities for ISPs to dynamically manage their shared storage assets [2]. . The question of caches' placement becomes, therefore, central given the limited resources and the continuous increase in data consumption [3].

Several studies in the literature have dealt with the problem of caches' placement, but the results are, somehow, equivocal. In fact, some studies point out the effectiveness of caching at the edge of the network to bring content closer to end-users [4]. This allows popular contents to be retrieved much faster while significantly reducing the intra and inter ISPs amount of exchanged traffic. Some other studies highlighted the benefit of caching at the core of the network (i.e. at the Point of Presence (POP) level) [5][6], this minimizes the use of peering links given the complete absence of redundancy between caches, which guarantee reduced costs for ISPs and good quality for contents providers. Consequently, the real problem is not to place all the content at the edge or at the core of the network, but rather to manage it efficiently, in such a way that it sometimes guarantees the contradictory objectives of the various stakeholders.

The deployment of an optimal caching system within a network infrastructure, up to the edge, is a very complex problem and remains an open issue [7]. Unlike the vast majority of existing work, we are not only interested in the spatial placement of caches in an infrastructure, but also in determining the optimal amount of cache to be placed at the different locations of the network, which makes the placement problem even harder.

We formalize the problem of caches placement as a multi-objective optimization problem, in which we aim to satisfy two conflicting objectives. The first objective is to minimize the average distance from which contents are retrieved, which tends to put the contents at the edge. The second objective is to minimize the content provider's load (i.e. peering links' utilization), which tends to put the contents at the core of the network. Since measuring these objectives is not easy in a practical use case, we consider a mathematical model that we have developed [8], and that can easily be adapted to different caching strategies.

The problem being NP-hard, as explained in this paper, the resolution of the optimization problem can only be done for small and unrealistic network configurations. In this respect, we propose in the following to solve the problem using the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic [9], which has demonstrated its effectiveness in resolving different types of combinatorial optimization problems.

In this paper, we propose new extensions and substantial improvements of our previous work [10], which are summarized in the following points:

- We propose a detailed and up to date state of the art.
- We describe the solution in more detail by introducing three new algorithms, one explaining the adaptation of GRASP to the problem of cache

placement, a second algorithm describing the construction phase and a third one describing the local search strategy.
- We analyze the complexity of the considered cache placement problem.
- We provide more results and an in-depth analysis of the problem.

The reminder of this paper is organized as follows. In section 2, we present the related work. Section 3 provides the detailed description of the cache placement problem and the proposed solutions. Then, the evaluation of our proposal is expressed in section 4. Finally, section 5 summarizes the achievements in this paper and introduces our future work.

2 Related Work

Many studies have investigated the problem of cache resources allocation and placement in the context of multi-cache networks [11] (e.g., Content Delivery Networks "CDN", Information Centric Networking "ICN", etc.).

The work presented in [12] by Krishnan et al. was one of the earliest studies that tackled the problem of the optimal cache resources placement. They examined the cache location problem in the case of transparent in-route caches in the context of web caching. The problem was modeled as a k-median problem, where the objective is to minimize the network traffic flow, and solutions were proposed based on dynamic programming and greedy heuristics. The classic k-median problem consists on finding k centers (i.e. cache resources) such that the clusters they form are the most compact (i.e. lower cost). The authors in [13] and [14] studied the storage capacity allocation in hierarchical content distribution systems through a multi-commodity problem, which generalizes the single commodity k-median problem. They propose a two-step algorithm capable of solving such problems when tree graphs are considered, which has been extended through approximations to cover the case of general graphs. They also provided a greedy algorithm due to the high complexity of the exact solution. The authors in [15] addressed the problem of placing mirrors of Internet content on a restricted set of hosts through modeling it as a slightly different version of the minimum k-center problem, considering the latency as the cost function to be optimized. In the minimum k-center problem, which is an NP-complete one, the objective is to find a placement of a given number of centers (i.e. content servers) such that the maximum distance from a node (i.e. end-user) to the nearest center is minimized. They proposed a greedy algorithm and a heuristic based on nodes' degree to solve the problem.

Later on, and with the proliferation of Content Delivery Networks, many works have focused on replica server (or cache resources) placement solutions in traditional and emerging CDN-based paradigms (cloud-based CDN, NFV-based CDN, etc.). In [16], the authors proposed a solution to the media server placement problem by modeling it using the uncapacitated facility location problem. To ensure the scalability of their algorithm, they have considered the case where all the end-users locations can be potential placement of replica servers. In [17], Rodolakis et al. introduced polynomial and pseudo-polynomial

algorithms to solve the replica server placement problem using the splittable soft capacitated facility location model. Their aim is to find the best location of replica servers and the number of servers that should be used and assigned to end-users groups in a way that minimizes the cost and satisfies the Quality of Service (QoS). In [18], Chen et al. studied the problem of building distribution paths and placing Web server replicas in cloud CDNs to minimize the cost incurred on the CDN providers while satisfying QoS requirements for user requests. They provide an ILP formulation of the problem that happened to be NP-hard, and propose offline and online heuristics to solve it. In [19], the authors presented a CDNaaS platform (CDN as a Service) that enables the creation of CDN slices across multiple cloud domains and the deployment of virtual resources (including virtual caches) from multiple IaaS providers (Infrastructure as a Service), where different VNFs (Virtual Network Functions) are running. They studied in their work the optimal placement of these VNFs by modeling this problem as two distinct Linear Integer problems, where the aim is to minimize the incurred cost of resource placement and maximize the Quality of Experience (QoE) of the virtual streaming service. By means of the bargaining game theory, they proposed a solution that ensures a fair trade-off between the cost and the QoE.

The study in [6] was probably the first attempt to investigate the cache allocation problem in Content Centric Networking (CCN). They used in their study different metrics to measure the centrality of routers like degree, closeness and betweenness in order to decide where the cache should be distributed along the network's nodes. They suggest that deploying more cache resources at the core nodes of the network is better than a placement at the edge. In later works [20] [21], the authors have concluded the opposite, suggesting that placing larger caches at the edge is more effective. Wang et al., in their work [22], have studied the impact of content popularity distribution on caching performance in CCN. They show that placing caches into the network core is better suited for content requests with uniform distribution and that in case of highly skewed popularity demands patterns, pushing cache resources to the edge yields better performance.

Considering a single metric or objective when dealing with the cache allocation problem clearly reduces the complexity of the problem, but has led many studies to find results that appear contradictory. There are many aspects in our proposal in this matter that makes it different from the existing works. We propose a versatile solution that takes into account at the same time more than one performance metric to solve the cache allocation problem and it can be tuned in order to seek some specific results. Moreover, the proposed tool builds a solution by measuring the contribution of all the nodes by the means of an analytic model capable of estimating the network performance, which will allow to take into account the impact of a node's performance on the others. In addition, more than one solution can be generated for the same use case, which gives more flexibility and enables adapting to additional constraints. We will see later that placing most (or all) of the cache resources at the edge or in the network core cannot be an absolute solution since it will

depend on what performance metrics we try to optimize and what outcomes we aim to obtain.

3 Multi-objective cache placement strategy

3.1 Motivations

During the last decade, ISPs and CPs infrastructures underwent a major metamorphosis driven by new networking paradigms, namely: Software Defined Networks (SDN) [23] and Network Function Virtualization (NFV) [24]. The upcoming advent of 5G will certainly represent the most important achievement of this evolution [25]. In this context, static (planning) or dynamic (on-demand) network resources placement, especially caching, remains an open issue. Indeed, co-locating caching resources at the core of the network optimizes caches, but not the network. Distributing caches optimizes network resources, but reduces the efficiency of caches, due to the existing redundancy at the edge. The optimal placement of caching resources is one of the most important issues to address in multi-cache networks in general, especially due to the very expensive cost of deploying distributed storage capacities along the network.

In previous works that addressed the cache allocation problem, only one performance gain metric is generally considered (e.g. access latency, cache hit, etc.) to find a storage distribution solution or to compare between different possibilities. In this work, which was introduced in [10], we use jointly the following two performance metrics to evaluate the cache gain: content provider load and average distance ratio. The first metric represents the amount of contents that were served by the origin server over all the requests sent in the network. The second one depicts the average distance travelled to get contents in the network over the obtained distance without caching. We chose these two metrics for their importance in representing the cost and gain obtained from the use of the in-network caching. A high content provider load means most of the requested contents are not served by the intermediate caches, and thus retrieved from their original location. Accessing the main source of contents is very expensive for network operators, and this is why it is important for them to keep the content provider load as low as possible. On the other hand, a low average distance to get contents means a better Quality of Experience for users (QoE). Hence, a good cache allocation strategy should find the best trade-off between these two metrics.

3.2 System assumptions

Let $G = (V, E)$ be the graph representing a general network of caches, where $V = \{v_1, ..., v_M\}$ depicts the nodes of the network and $E \subset V \times V$ is the set of links connecting the nodes. Each node in the network is equipped with

Table 1: Summary of the notations.

Term	**Desciption**
M	Number of nodes in the network
R	Number of items in the catalog
N	Cache size
c_r	Content with a popularity/rank r
p_r	Probability to request an item c_r
α	Zipf law distribution parameter
X	Cache placement configuration
$f_1(X)$	Content provider load of a cache configuration X
$f_2(X)$	Average distance ratio of a cache configuration X
T_c	Represents the total cache resources to be distributed in the network
$P_{hit}(r)$	Hit probability of content c_r
$P_{miss}(r)$	Miss probability of content c_r
$Dist(i)$	Distance from where the clients requests were generated to the node v_i
λ	GRASP parameter

a caching module used to store contents locally. Let $C = \{c_1, ..., c_R\}$ be the set of the catalog's contents available for the users. We assume that all the accessible contents in the system have an identical size and are divided into small packets or chunks, which are in turn of the same size. The cache capacity is then expressed in terms of the number of contents or chunks that can be stored. All the available contents are stored permanently at one or more servers attached to some nodes within the network (i.e. origin server(s)). In the rest of the paper and for the sake of readability, we will use the term node/cache interchangeably as well as the terms rank/popularity and content/item/object.

Clients are attached to the network nodes, sending requests into the network looking for contents. The pattern of these requests is characterized by the Independent Reference Model (IRM) [26]. By considering the IRM model, users generate an independent and identically distributed sequence of requests from the catalog C of R objects. Specifically, the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law, where the contents are ranked decreasingly according to their popularity from 1 to R. Since about 80% of Internet traffic is generated by video related applications [3], we address in our work videos services, where the contents feature a skewed popularity distribution. As already argued in many previous studies [27], the latter fits the Zipf law: the probability to request the content of rank r is: $p_r = r^{-\alpha} / \sum_{i=1}^{R} i^{-\alpha}$, where α, the skew of the distribution, depends on the type of the accessible objects [28]. In the present work, the LRU algorithm is used to manage the node's content store and two caching schemes will be considered: Leave Copy Everywhere (LCE) [29] [30] and Two Queue (2Q) [31]. Under the LCE scheme, every data packet is always stored once received by a caching node. When the 2Q algorithm is used, the incoming contents are filtered to admit in the cache only the most popular ones.

3.3 Problem formulation

A cache allocation solution, which could be handled by an NFV orchestrator, can be defined by a vector $X = (x_1, \ldots, x_M)$, where x_i represents the amount of storage capacity placed in node v_i. To compute the content provider load or the average distance ratio of a configuration of caches X, we use our model MACS (Markov chain-based Approximation of Caching Systems) [8][32]. MACS is an analytic tool that allows us to estimate the cache hit ratio of an interconnection of caches, which can be used to compute other performance metrics like the content provider load and the average distance ratio. By using MACS and for each cache allocation configuration X, we can evaluate the performance of the whole system in its steady-state and not just during a transient phase. This is opposed to the simplified and use case-specific models used to deal with the deployment of caches in previous works dealing with the cache placement problem.

Since in this work the caching capacity is expressed in terms of the number of contents that can be stored, then we have $x_i \in \mathbb{N}$. As we measure the caching gain through evaluating the content provider load and the average distance ratio in the network, our objective is to find a cache distribution solution such that the evaluation metrics that we have chosen are optimized (i.e. minimized). The cache placement is then formulated as a multi-objective optimization problem as follows:

$$\begin{aligned} \min \quad & f_1(X), f_2(X) \\ \text{s.t.} \quad & \sum_{i=1}^{M} x_i \leq T_c, \ x_i \in \mathbb{N} \text{ for all } i. \end{aligned} \tag{1}$$

The value of T_c represents the total cache resources to be distributed in the network. The expressions $f_1(X)$ and $f_2(X)$ are both expressed as a percentage and represent, respectively, the content provider load and the average distance ratio of a cache placement configuration X using MACS. The primary function of MACS is to calculate the cache hit ratio of the network's nodes. Then, we can compute $f_1(X)$ and $f_2(X)$ as follows:

$$\begin{aligned} f_1(X) &= 100 \times \prod_{i=1}^{s} P_{miss}(i), \\ f_2(X) &= 100 \times \frac{\sum_{i=1}^{s} \left(P_{hit}(i) \times \mathit{Dist}(i) \times \prod_{j=1}^{i-1} P_{miss}(j) \right)}{\mathit{Dist}(s)}. \end{aligned} \tag{2}$$

The values $P_{miss}(i)$ and $P_{hit}(i)$ represent, respectively, the cache hit and cache miss of a network's node v_i. The expression $\mathit{Dist}(i)$ is the distance from where the clients requests were generated to the node v_i. The index s represents the nearest node v_s to the clients to which a content provider is attached (i.e. where a permanent copy of the contents is available). In the definition of $f_1(X)$ and $f_2(X)$ and for sake of clarity, we supposed that the network is formed by a line of s nodes numbered from 1 to s where the clients are attached to node

v_1 and the content repository is located just after node v_s. It has to be noted that of course, we can compute these metrics in any type of network topology and that we could consider the delay between nodes instead of distance and thus calculate the average network delay.

Since here we are dealing with a multi-objective optimization problem, in which we want to minimize $f_1(X)$ and $f_2(X)$, the solutions will be a set of *efficient* points usually called the Pareto frontier. Pareto efficiency or optimality implies that a solution to a multi-objective problem is such that no single objective can be improved without deteriorating another one. In our case, a solution X^* is said to be efficient if there is no other solution X such that $f_1(X) < f_1(X^*)$ and $f_2(X) \leq f_2(X^*)$ at the same time, or $f_2(X) < f_2(X^*)$ and $f_1(X) \leq f_1(X^*)$. Given that integer nonlinear programming is an NP-hard problem, solving the cache allocation problem as we modeled below will come at a very high computational cost. More specifically and due to the non-linearity of the objective functions, we need to perform an exhaustive search method in order to enumerate all possible candidates that respect the problem constraints and find the set of optimal cache distributions. If we look closely to our formulation of the problem, the task of enumerating all possible candidates comes down to computing the *weak composition* of an integer n into k parts, i.e., writing n as the sum of a sequence of non-negative integers. A weak composition $C_{n,k}$ [33] has a cardinality of

$$|C_{n,k}| = \binom{n+k-1}{k-1} = \frac{(n+k-1)!}{n!(k-1)!}. \tag{3}$$

In our case, $n = T_c$ and $k = M$, where the set of M non-negative integers has a sum equal to T_c and represents the cache resources allocated to each node of the network. It is clear that $|C_{n,k}|$ is huge for high values of n and k. Therefore, we propose the use of the meta-heuristic GRASP to solve the cache placement problem (see the model (1)).

3.4 Solving cache allocation problem using GRASP

3.4.1 Mono-objective GRASP

The Greedy Randomized Adaptive Search Procedure or GRASP [9] is an iterative process, where each iteration consists basically of two steps: construction and local search. The construction phase seeks to build a feasible solution using a greedy randomized approach, whose neighborhood will be investigated during the local search in order to find a local optimum. The pseudo-code of Algorithm 1 depicts the main blocks of a mono-objective GRASP procedure, where *Max_Iterations* is the number of iterations that are performed (later, we will see the role of the parameter λ and the case of multi-objective GRASP). The best overall solution is, then, kept as the final result. The construction phase operations are shown in Algorithm 2. Let's recall first that

Algorithm 1 GRASP-based cache allocation

```
1: function GRASP(Max_Iterations, λ)
2:     Solution ← Greedy_Randomized_Construction(λ);
3:     Solution ← Local_Search(Solution);
4:     Best_Solution ← Solution;
5:     for k = 1, ..., Max_Iterations − 1 do
6:         Solution ← Greedy_Randomized_Construction(λ);
7:         Solution ← Local_Search(Solution);
8:         if f(Solution) < f(Best_Solution) then
9:             Best_Solution ← Solution;
10:        end if
11:    end for
12:    return Best_Solution;
13: end function
```

the solution S to be built during the construction phase is defined by a vector $X = (x_1, \ldots, x_M)$, where x_i represents the amount of storage capacity placed in node v_i. Initially, no cache resources are allocated to the nodes, so at the beginning, $S = (0, \ldots, 0)$. The set CS will contain the candidate elements, which will be used for the solution S. In our case and at each step, CS will contain a set of cache placement configurations X_i where a partial resource that we denote P_c, is taken from the total available cache T_c and allocated to one of the network's nodes. If we have, for example, a network with three nodes, $T_c = 100$ and $P_c = 10$, the initial candidate set will then be: $CS = \{(10, 0, 0), (0, 10, 0), (0, 0, 10)\}$.

Each candidate is then evaluated with a greedy function in order to build a restricted candidate list RCL, which will contain some of the candidate set who have the best evaluation values (e.g., $RCL = \{(10, 0, 0), (0, 10, 0)\}$). The limitation criteria of the list cardinality can be either based on the number of elements or based on their quality, as we did in Algorithm 2 (line 18). The elements added to the RCL list will then be those having an evaluation value inferior to the threshold (i.e., $f(X) \in [f^{\min}, f^{\min} + \lambda(f^{\max} - f^{\min})]$). The value of λ will control the insertion condition of candidate elements to RCL ($\lambda \in [0, 1]$). The case $\lambda = 1$ is equivalent to a random construction, while $\lambda = 0$ corresponds to a pure greedy algorithm. Once RCL is built, one element is randomly selected and added to the solution S being built. The candidate list CS and the evaluation function $f(CS)$ are then updated and the construction is repeated (line 7 to 24) until the total use of the cache budget T_c. If we consider for example that the second element from RCL has been chosen, the current partial solution will then be $S = (0, 10, 0)$ and the new candidate list will contain: $CS = \{(10, 10, 0), (0, 20, 0), (0, 10, 10)\}$.

Once the cache budget T_c is distributed, the local search will seek to improve the generated solution (e.g., $S = (20, 40, 40)$) by evaluating its neighborhood (Algorithm 3). The efficiency of a local search method depends on many aspects, such as the neighborhood structure of the considered solution, the neighbors search technique and the starting solution itself. Two methods can be used for the neighborhood search: the *best-improving* strategy

Algorithm 2 Construction phase

function *Greedy_ Randomized_ Construction*(λ)
  for $i = 1, \ldots, M$ **do**
    $x_i \leftarrow 0$; // Cache allocated for each node v_i
  end for
  $S \leftarrow (x_1, \ldots, x_M)$; // Current solution
  $CS \leftarrow \emptyset$; // Initial candidate set CS
  while $T_c \neq 0$ **do** // Construction of the solution S
    $T_c \leftarrow T_c - P_c$
    for $i = 1, \ldots, M$ **do** // Create candidate set CS
      $x_i \leftarrow x_i + P_c$;
      $X \leftarrow (x_1, \ldots, x_M)$;
      $CS \leftarrow CS \cup \{X\}$;
      $x_i \leftarrow x_i - P_c$;
    end for
    Evaluate the incremental costs $f(X)\ \forall X \in CS$;
    $f^{\min} \leftarrow \min\{f(X) \mid X \in CS\}$;
    $f^{\max} \leftarrow \max\{f(X) \mid X \in CS\}$;
    $RCL \leftarrow \{X \in CS | f(X) \leq f^{\min} + \lambda(f^{\max} - f^{\min})\}$;
    Select an element X^* from the RCL at random;
    $S \leftarrow \{X^* = (x_1^*, \ldots, x_M^*)\}$;
    for $i = 1, \ldots, M$ **do**
      $x_i \leftarrow x_i^*$;
    end for
  end while
  return S;
end function

or *the first-improving* one. The best-improving strategy consists on investigating all the neighbors and the current solution will then be replaced by the best neighbor found. In the case when a first-improving method is used, the current solution will be replaced by the first neighbor whose evaluation value is better. In our case, we used for the neighborhood search a best-improving strategy as follows: starting from the solution generated by the greedy randomized construction, we transfer an amount of cache P_c from one node to another and explore all the possible cases looking for a cache configuration whose evaluation function value is better than the current one (e.g., $Neighbor(S) = \{(10, 50, 40), (30, 30, 40), \text{etc.}\}$). We repeat these steps until the current solution can no longer be improved, which then will be returned as the output of the local search procedure of GRASP.

As for the complexity of GRASP algorithms (i.e. construction and local search) applied to our formulation of the cache allocation problem, we have a complexity of $O(T_c\ M)$ for the construction phase and $O(M^2)$ for the local search phase.

3.4.2 Multi-objective GRASP

In single-objective GRASP, only one greedy function is used to evaluate the candidate solutions during the construction and local search phases. In multi-objective GRASP [34], we have in the general case k greedy functions

Algorithm 3 Local search phase

```
1: function Local_Search(S)
2:     do
3:         (x_1, ..., x_M) ← S;
4:         for i = 1, ..., M do
5:             x_i ← x_i − P_c;
6:             for j = 1, ..., i − 1 do // Cache transfer
7:                 x_j ← x_j + P_c;
8:                 if f(x_1, ..., x_M) < f(S) then
9:                     S ← (x_1, ..., x_M)
10:                end if
11:                x_j ← x_j − P_c;
12:            end for
13:            for j = i + 1, ..., M do // Cache transfer
14:                x_j ← x_j + P_c;
15:                if f(x_1, ..., x_M) < f(S) then
16:                    S ← (x_1, ..., x_M)
17:                end if
18:                x_j ← x_j − P_c;
19:            end for
20:        end for
21:    while Solution S is not locally optimal
22:    return S;
23: end function
```

$(f_1(X), f_2(X), \ldots, f_k(X))$. The evaluation of a candidate element X and the different objective functions can be achieved using the following two ways:

- *Pure construction/local search:* this method consists on using one single objective during each phase of the solution generation. The selection of which evaluation function to be used in each iteration can be done using either a pure-random approach or a pure-ordered approach. In the first method, the entire construction or local search is guided by only one single evaluation function that we select randomly from the set of the objective functions $(f_1(X), \ldots, f_k(X))$. In the second method, we select a greedy function in an ordered way and one at a time and use it in all the steps of the construction phase or the local search. In other words, the candidate elements are evaluated with $f_1(X)$ during the first iteration. Then, we use $f_2(X)$ in the second iteration, and so on until we reach the $k+1^{th}$ iteration, in which we go back again to $f_1(X)$.
- *Combined construction/local search:* this method consists on using more than one greedy function in each iteration following either a sequential combined approach or weighted combined approach. Using a sequential combined method means that each step of the construction or local search is guided by a different evaluation function, which can be chosen randomly or in an ordered way. When the weighted combined method is applied, a weighted combination of the objective functions is used in each step of the construction or local search[1]. We can either change the weights between

[1] Weighted combined construction: $f(X) = \sum_{i=1}^{k} w_i f_i(X)$, where w_i is the weight of the evaluation function $f_i(X)$.

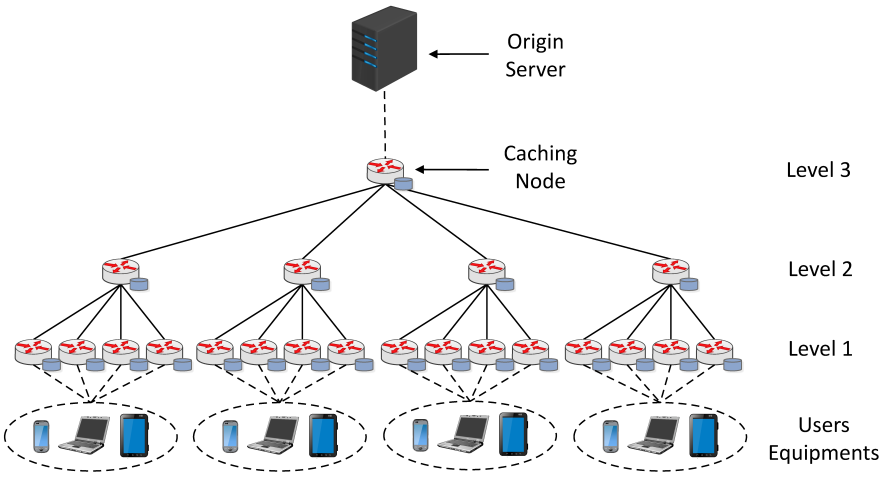

Fig. 1: An example of a network architecture for content delivery with cache-enabled nodes.

the steps of each phase or keep them the same during the whole process. It has to be noted that the weights can help us scale evaluations functions having significantly varied magnitudes into similar and comparable ones. In addition, some greedy functions in multi-objective optimization can be minimized while others maximized. In this case, the weights can take positive and negative values in order to take into account this fact.

In our case, we have two evaluation functions to be used in the multi-objective GRASP: $f_1(X)$ and $f_2(X)$ and there are many methods that can be used to compute the outcome of each configuration in the construction and local search phases. For example, one can use $f_1(X)$ in the construction phase and $f_2(X)$ during the local search and vice versa, or choose randomly between the objective functions during each GRASP iteration. We can also consider a weighted combined method ($f(X) = w_1 f_1(X) + w_2 f_2(X)$). The choice of which method to be used to evaluate a candidate solution (or partial solution) can depend on some desired results that should be achieved or some constraints that should be respected.

In our work, we considered two methods to generate the solutions based on the defined objective functions $f_1(X)$ and $f_2(X)$. In the first method, which matches a pure-ordered approach, we use in each iteration one objective function ($f_1(X)$ or $f_2(X)$) during the construction phase. The other one is then used during the local search. Then, we alternate between the selected functions for each phase in the next iteration. For example, if we chose in the first iteration respectively $f_1(X)$ and $f_2(X)$ for the construction and local search phases then, in the second iteration, we use $f_1(X)$ in the local search and

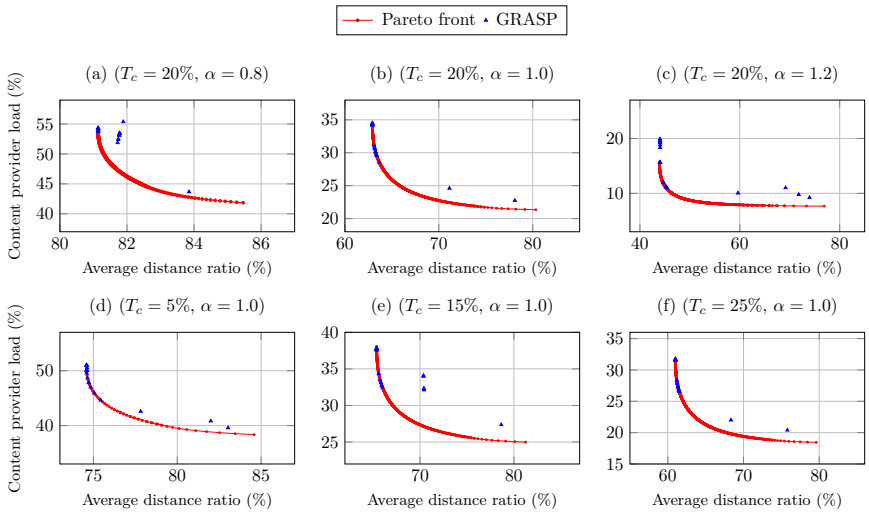

Fig. 2: Comparison between the Pareto front and GRASP solutions under the LCE scheme.

$f_2(X)$ in the construction phase, etc. This approach will allow us to produce diversified solutions of good quality. In the second method, we used a weighted combination of the evaluation functions as follows:

$$f(X) = 0.5 \times \left(\frac{|f_1(X) - 100|}{|Opt(f_1(X)) - 100|} + \frac{|f_2(X) - 100|}{|Opt(f_2(X)) - 100|}\right) \times 100. \quad (4)$$

The expressions $Opt(f_1(X))$ and $Opt(f_2(X))$ represent respectively the theoretical optimal values of the metrics $f_1(X)$ and $f_2(X)$ for a fixed cache resources budget T_c. More specifically, $Opt(f_1(X))$ is obtained by allocating all the available cache resources at the root node and then, computing the content provider load. As for $Opt(f_2(X))$, it is obtained by computing the average distance ratio metric where all the total cache resources are placed at one-hop from the clients. Since here we want to minimize $f_1(X)$ and $f_2(X)$ (expressed as percentages), then the range of values taken by these functions is: $[Opt(f_i(X)), 100]$ for i in $\{1, 2\}$. The value 100 represents the worst case, i.e., where no cache resources were allocated and thus, all the contents are retrieved from the main source and the average distance to get objects is not reduced. So, what we did is to compute how much improvement can be achieved in each of these performance metrics relatively to the optimal values that can be reached. For example, and for a certain configuration $X = (x_1, \ldots, x_M)$, if we have $f_1(X) = 40\%$ and $Opt(f_1(X)) = 20\%$, then $(|40-100|/|20-100|)\times 100 = 75\%$. This value means that we have achieved 75% of the optimal outcome of the content provider load.

The evaluation function $f(X)$ can be seen as a ratio of efficiency where the aim is to improve as much as possible the network's overall performance. It is also a mean to scale the evaluation functions $f_1(X)$ and $f_2(X)$ into similar and comparable magnitudes. In addition, the weighting coefficients (fixed to 0.5 in

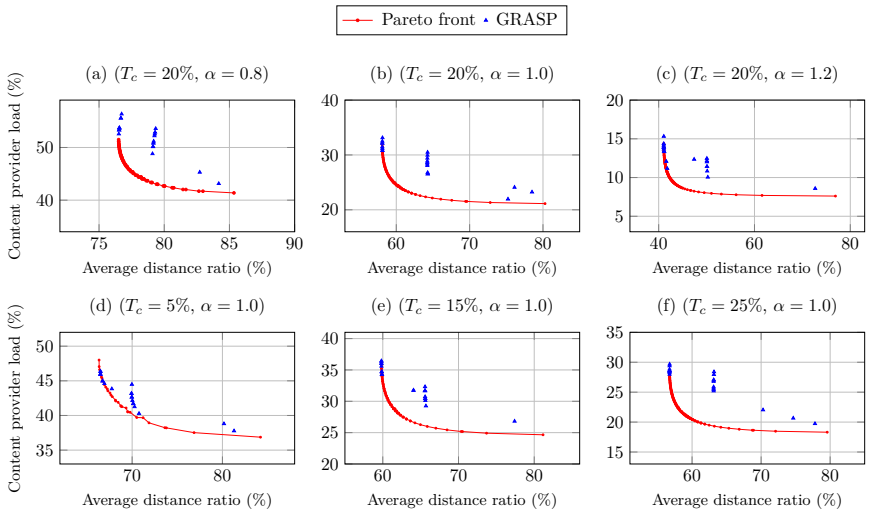

Fig. 3: Comparison between the Pareto front and GRASP solutions under the 2Q scheme.

our case) can be tuned in order to give more importance to one metric over the other. In the case where the weighted combined approach defined by $f(X)$ is adopted, the cache placement is formulated as the following optimization problem:

$$\begin{aligned} \max \quad & f(X) \\ \text{s.t.} \quad & \sum_{i=1}^{M} x_i \leq T_c, x_i \in \mathbb{N} \text{ for all } i. \end{aligned} \tag{5}$$

4 Performance evaluation

In this section, we present the evaluation results of the cache allocation problem using the methods presented previously. We will expose at first different results comparing the outcomes of the metaheuristic to the optimal solutions (i.e. obtained using an exhaustive search approach). Then, we will display some of the results obtained using our application of GRASP to the problem that we have formulated, to have an idea about the solutions that can be proposed by our approaches.

4.1 Simulation configuration

The different evaluations were conducted on a typical three-level network that contains 21 nodes forming a perfect 4-ary tree topology where the distance and the latency between each two adjacent nodes are the same (see Figure 1). We considered in our experiments a catalog of contents containing $20,000$ 1-chunk

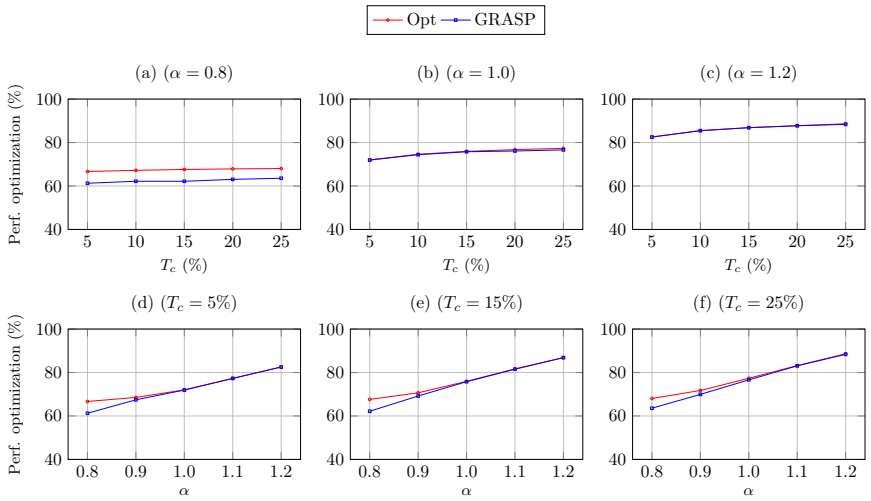

Fig. 4: Performance optimization vs total cache size using GRASP compared to the optimal values under the LCE scheme.

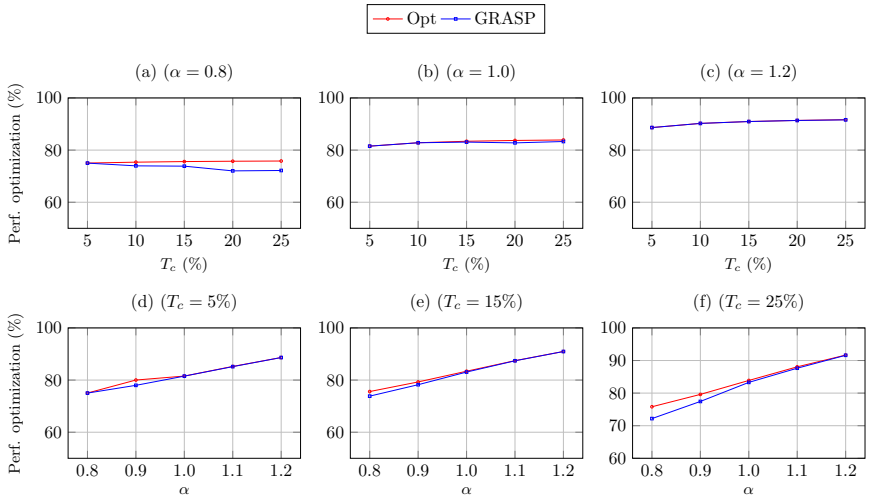

Fig. 5: Performance optimization vs total cache size using GRASP compared to the optimal values under the 2Q scheme.

sized objects whose popularity distribution follows the Zipf's law. Permanent copies of the available contents are hosted on one repository attached to the root node of the network, and the users are attached to the network's leaves (i.e. level 1 of the network).

Following common practice [26]-[35], we assume in this work that after a cache miss and when a content is decided to be cached by a node, it will be downloaded instantaneously. The contents requested by the different clients are

Table 2: Cache allocation solutions using GRASP with two separate evaluation functions when LCE is used.

$T_c = 20\%$				$\alpha = 1.0$			
α	X	$f_1(X)$	$f_2(X)$	T_c	X	$f_1(X)$	$f_2(X)$
0.8	(320,1696,2080)	53.47%	81.78%	5%	(224,208,592)	44.63%	75.42%
	(768,1376,1952)	54.80%	81.15%		(288,208,528)	45.94%	75.04%
	(736,1280,2080)	53.84%	81.14%		(352,224,448)	47.81%	74.72%
	(16,2288,1792)	55.38%	81.89%		(400,240,384)	49.56%	74.59%
1.0	(1536,1168,1392)	34.39%	62.92%	15%	(736,688,1648)	32.50%	65.99%
	(944,880,2272)	28.57%	63.65%		(864,640,1568)	33.09%	65.83%
	(1136,864,2096)	29.54%	63.38%		(992,672,1408)	34.36%	65.59%
	(1280,928,1888)	30.80%	63.14%		(1168,832,1072)	37.54%	65.38%
1.2	(1952,1040,1104)	15.62%	44.00%	25%	(1264,1184,2672)	26.54%	61.52%
	(1760,2176,160)	19.88%	44.00%		(1392,1168,2560)	27.06%	61.39%
	(1456,720,1920)	12.02%	44.67%		(1568,1152,2400)	27.84%	61.25%
	(1088,800,2208)	11.17%	45.24%		(1904,1488,1728)	31.75%	61.00%

forwarded according to the Shortest Path Routing (SPR) algorithm [36]. With SPR, when a client's interest cannot be satisfied by a node, it is forwarded along the shortest path to the closest permanent copy of the requested content. We tested different values of the Zipf law's skew parameter α going from 0.8, corresponding to User Generated Contents (UGC) like Youtube, to 1.2, corresponding to a Video on Demand (VoD) service [37].

The parameter λ that controls the amounts of greediness and randomness in GRASP was set to 0.5. Indeed, by choosing a greedy approach ($\lambda = 0$), there is a high chance to stick at a local minimum. Besides, by choosing a completely random approach ($\lambda = 1$) one risks to not converge towards the good solution. A λ in between allows to correctly explore the space of solutions.

The total cache T_c is expressed as a ratio of the catalog size. Two caching strategies were used during the experiments: LCE and 2Q. The 2Q policy is, of course, more efficient than LCE as it was shown in [8] but the aim here is to see the behavior of our model where different caching strategies are used. As we mentioned previously, the network performance is evaluated using the content provider load and average distance ratio metrics. For the sake of simplicity, the cache resources are allocated in a way that the nodes located on the same level of the network have the same cache size. Since there are 16 nodes on the first level of the network, P_c should be a multiple of 16 and in our case, it was set to 16. Thus, a cache resource placement can be described as $X = (x_1, x_2, x_3)$, where x_i represents the total cache allocated at level i of the network.

4.2 Model results and analysis

In Figures 2-3, we compare the network's performance metrics of the different cache distribution solutions generated by GRASP (20 iterations) with

Table 3: Cache allocation solutions using GRASP with two separate evaluation functions when 2Q is used.

$T_c = 20\%$				$\alpha = 1.0$			
α	X	$f_1(X)$	$f_2(X)$	T_c	X	$f_1(X)$	$f_2(X)$
0.8	(64,2240,1792)	52.90%	79.29%	5%	(320,384,320)	44.59%	66.92%
	(1216,832,2048)	52.56%	76.51%		(512,256,256)	46.35%	66.49%
	(1344,896,1856)	53.72%	76.55%		(192,512,320)	43.84%	67.75%
	(1472,1216,1408)	56.35%	76.73%		(64,448,512)	41.70%	70.12%
1.0	(64,2048,1984)	28.60%	64.23%	15%	(64,1280,1728)	30.42%	65.60%
	(64,1984,2048)	30.44%	58.13%		(1152,704,1216)	34.27%	59.88%
	(1600,768,1728)	30.83%	58.11%		(1216,704,1152)	34.70%	59.85%
	(1792,1024,1280)	33.13%	58.12%		(1408,768,896)	36.40%	59.81%
1.2	(448,2432,1216)	12.34%	47.39%	25%	(64,3200,1856)	28.40%	63.17%
	(1728,960,1408)	13.41%	41.78%		(1920,960,2240)	27.99%	56.74%
	(1984,896,1216)	14.13%	41.05%		(1984,1024,2112)	28.53%	56.73%
	(2176,1024,896)	15.31%	41.07%		(2112,1152,1856)	29.66%	56.75%

the Pareto front using the separate evaluation functions approach (content provider load and average distance ratio metrics). We can see from the plots how the solutions produced by GRASP, independently from the used caching scheme, are very close to the set of dominant points in the different tested scenarios, which reflects the good quality of the metaheuristic outcomes.

In Figure 4-5, we compare the solutions obtained using the weighted combined approach with the optimal values (only the best solution obtained with GRASP is shown). The results in these graphs also show the efficiency of the metaheuristic in giving solutions close to the optimal ones. When the Zipf's parameter α is set to 0.8, we observe that the solutions generated by GRASP are not as close to the optimal values as the results obtained in the other cases. Having a low value of α ($\alpha < 1$) means a lower difference in popularity between the different contents, which makes it more difficult to construct cache distribution solutions approaching the optimal performance.

In Tables 2-3, we display the cache allocation solutions generated by GRASP (only the results of four iterations of GRASP are shown) using separate evaluation functions approach and testing two different caching schemes: LCE (Table 2) and 2Q (Table 3). Depending on the Zipf law's parameter α, the total cache resources available T_c and the adopted caching strategy, different solutions are generated using the proposed GRASP-based algorithms. The choice of the final solutions between the different proposed ones can be based on some constraints that should be respected. For example, and under the LCE policy, if $T_c = 20\%$ and $\alpha = 1.2$ (Table 2) and if the priority is to minimize the content provider load ($f_1(X)$), then the allocation $X = (1088, 800, 2208)$ is the best one. In case when the 2Q scheme is used, if $\alpha = 1.0$ and $T_c = 15\%$ (Table 3) and one wants to minimize first the average distance ratio ($f_2(X)$), then the allocation $X = (1408, 768, 896)$ is the one to choose.

Table 4: Cache allocation solutions using GRASP with weighted combined evaluation function when LCE is used.

$T_c = 20\%$			$\alpha = 1.0$		
α	X	$f(X)$	T_c	X	$f(X)$
0.8	(592,1248,2256)	62.41%	5%	(176,176,672)	71.92%
	(512,1600,1984)	60.70%		(208,192,624)	71.69%
	(464,1824,1808)	59.55%		(272,176,576)	71.31%
	(432,1904,1760)	59.23%		(352,208,464)	69.91%
1.0	(1056,816,2224)	76.12%	15%	(672,512,1888)	75.76%
	(1152,864,2080)	75.77%		(768,592,1712)	75.40%
	(1232,896,1968)	75.46%		(912,624,1536)	74.85%
	(1328,912,1856)	75.10%		(1040,688,1344)	74.02%
1.2	(1712,176,2208)	87.83%	25%	(1328,1056,2736)	76.61%
	(1824,96,2176)	87.72%		(1408,1088,2624)	76.40%
	(1968,80,2048)	87.70%		(1520,1136,2464)	76.06%
	(2240,16,1840)	87.59%		(1648,1184,2288)	75.62%

Table 5: Cache allocation solutions using GRASP with weighted combined evaluation function when 2Q is used.

$T_c = 20\%$			$\alpha = 1.0$		
α	X	$f(X)$	T_c	X	$f(X)$
0.8	(704,832,2560)	72.03%	5%	(192,128,704)	81.51%
	(768,1088,2240)	70.36%		(192,256,576)	81.44%
	(768,1344,1984)	68.92%		(192,448,384)	80.49%
	(704,1600,1792)	67.79%		(256,448,320)	80.21%
1.0	(1024,576,2496)	82.77%	15%	(640,384,2048)	83.06%
	(1088,896,2112)	81.79%		(704,512,1856)	82.60%
	(1216,1216,1664)	80.35%		(768,640,1664)	82.05%
	(1152,1408,1536)	79.88%		(832,896,1344)	80.89%
1.2	(1152,384,2560)	91.36%	25%	(1152,576,3392)	83.29 %
	(1280,512,2304)	91.21%		(1280,768,3072)	82.72%
	(1408,832,1856)	90.72%		(1344,1088,2688)	81.88%
	(1536,1152,1408)	89.99%		(1472,1344,2304)	80.88%

In Tables 4-5, we expose the cache allocation solutions generated by GRASP using a weighted combined evaluation approach. In this scenario, the objective is to optimize the overall network performance without considering separately the evaluation metrics. The proposed solution will be the cache allocation that gives the best outcome. For example, and under the LCE strategy, if $T_c = 20\%$ and $\alpha = 1.0$ (Table 4), the solution $X = (1056, 816, 2224)$ is the best one. In case when the 2Q scheme is used and if we have for example $\alpha = 1.0$ and $T_c = 25\%$ then, the cache distribution $X = (1328, 1056, 2736)$ is the best solution generated by GRASP.

If we compare between LCE and 2Q in terms of network performance when different cache allocation solutions are used, the 2Q policy generally has better outcomes than LCE. However, in some cases the LCE strategy presents better results. For example, if we look at Tables 2-3 and in the case where $\alpha = 1.0$ and $T_c = 25\%$, the cache allocation $X = (1264, 1184, 2672)$ when LCE is used achieves better performance than the cache allocation $X = (64, 3200, 1856)$ when 2Q is applied. This result reinforces the importance of the cache's resources placement in distributed networks and how it can affect drastically the network performance.

We can see from the different presented results that there is no absolute solution for the cache allocation problem and it is not a question about whether to cache at the edge or in the core. Depending on the network's configuration and the objectives that one wants to achieve, multiple solutions can be adopted. Other metrics can be used as evaluation functions during the building of the solutions instead of those that we used in this work. The aim of our proposal is to give a tool capable of efficiently allocating distributed caching resources and versatile enough to adapt to specific desired network performance and constraints.

5 Conclusion

In this paper, the optimal placement of caching resources is investigated. The cache allocation issue is one of the most important problems to address when studying multi-cache networks because of the expensive cost of deploying distributed storage resources along the network. This issue is all the more important in the context of NFV, where placement is undoubtedly one of the most important keys for network optimization. In this regard, we propose an approach that solves the trade-off between minimizing the number of requests served at the origin server and minimizing the distance to retrieve contents. To do so, we modelled the cache allocation problem as a multi-objective integer nonlinear program, where our analytical tool MACS is used to evaluate the objective functions. Our formulation of the problem turned out to be an NP-hard problem. Thus, we proposed an adaptation of the metaheuristic GRASP to solve the problem using different evaluation functions to generate the solutions. The tests carried out were able to show the closeness of the results obtained compared to an optimal approach, in the case of different traffic patterns and cache sizes. By analyzing the different obtained results, we have shown that there is no absolute solution for the cache allocation problem and in contrast to the conclusions made in previous works, it is not a question about whether to cache at the edge or in the core of the network. The distribution of cache resources will depend on the network's configuration and the objectives that one wants to achieve. The versatility of our proposal allows it to be applied to any arbitrary network topology, and it can include other cost functions to be used as objective functions like, for example, the financial cost of cache

resources deployment and management. This will allow us to study caching from an economic point of view, which will be the scope of our future work.

References

1. H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc)," *IEEE Network*, vol. 28, pp. 18–26, Nov 2014.
2. X. Li, X. Wang, K. Li, and V. C. M. Leung, "Caas: Caching as a service for 5g networks," *IEEE Access*, vol. 5, pp. 5982–5993, Mar. 2017.
3. Cisco, "Cisco visual networking index: forecast and trends, 2017–2022," White paper, Nov. 2018.
4. E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, pp. 82–89, Aug. 2014.
5. A. Araldo, D. Rossi, and F. Martignon, "Cost-aware caching: Caching more (costly items) for less (isps operational expenditures)," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 1316–1330, May 2016.
6. D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," in *IEEE INFOCOM Workshops*, Mar. 2012, pp. 280–285.
7. K. S. Reddy, S. Moharir, and N. Karamchandani, "Effects of storage heterogeneity in distributed cache systems," in *16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, May 2018, pp. 1–8.
8. H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and S. Ait-Chellouche, "On the performance analysis of distributed caching systems using a customizable markov chain model," *Journal of Network and Computer Applications*, vol. 130, pp. 39–51, Mar. 2019.
9. T. A. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, pp. 109–133, Mar. 1995.
10. H. Ben-Ammar, Y. Hadjadj-Aoul, and S. Ait-Chellouche, "Efficiently allocating distributed caching resources in future smart networks," in *IEEE Annual Consumer Communications Networking Conference*, Jan. 2019, pp. 1–4.
11. J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "A survey on replica server placement algorithms for content delivery networks," *IEEE Communications Surveys Tutorials*, vol. 19, pp. 1002–1026, Nov. 2017.
12. P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, pp. 568–582, Oct. 2000.
13. N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "On the optimization of storage capacity allocation for content distribution," *Computer Networks*, vol. 47, pp. 409–428, Oct. 2004.
14. N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "Joint object placement and node dimensioning for internet content distribution," *Information Processing Letters*, vol. 89, pp. 273–279, Mar. 2004.
15. S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, Apr. 2001, pp. 31–40.
16. H. Yin, X. Zhang, T. Zhan, Y. Zhang, G. Min, and D. O. Wu, "Netclust: A framework for scalable and pareto-optimal media server placement," *IEEE Transactions on Multimedia*, vol. 15, pp. 2114–2124, Dec. 2013.
17. G. Rodolakis, S. Siachalou, and L. Georgiadis, "Replicated server placement with qos constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, pp. 1151–1162, Oct. 2006.
18. F. Chen, K. Guo, J. Lin, and T. L. Porta, "Intra-cloud lightning: Building cdns in the cloud," in *IEEE INFOCOM*, Mar. 2012, pp. 433–441.
19. I. Benkacem, T. Taleb, M. Bagaa, and H. Flinck, "Optimal vnfs placement in cdn slicing over multi-cloud environment," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 616–627, Mar. 2018.

20. I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2920–2931, Nov. 2014.
21. S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 147–158, Aug. 2013.
22. Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Transactions on Computers*, vol. 65, pp. 95–107, Jan. 2016.
23. J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an sdn-enabled nfv architecture," *IEEE Communications Magazine*, vol. 53, pp. 187–193, Apr. 2015.
24. B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, pp. 90–97, Feb. 2015.
25. J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1065–1082, Jun. 2014.
26. A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," *SIGMETRICS Performance Evaluation Review*, vol. 18, pp. 143–152, Apr. 1990.
27. C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *Proceedings IEEE INFOCOM Workshops*, Mar. 2012, pp. 310–315.
28. C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proceedings of the 24th International Teletraffic Congress*, Sep. 2012, pp. 1–8.
29. V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, Dec. 2009, pp. 1–12.
30. H. Ben-Ammar, S. Ait-Chellouche, and Y. Hadjadj-Aoul, "A Markov chain-based Approximation of CCN caching Systems," in *IEEE Symposium on Computers and Communications*, Jul. 2017, pp. 327–332.
31. T. Johnson and D. Shasha, "2q: A low overhead high performance buffer management replacement algorithm," in *Proceedings of the 20th International Conference on Very Large Data Bases*, Apr. 1994, pp. 439–450.
32. H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and S. Ait-Chellouche, "A versatile markov chain model for the performance analysis of CCN caching systems," in *IEEE Global Communications Conference*, Dec. 2018, pp. 1–6.
33. D. R. Page, "Generalized algorithm for restricted weak composition generation," *Journal of Mathematical Modelling and Algorithms in Operations Research*, vol. 12, pp. 345–372, Dec. 2013.
34. R. Marti, V. Campos, M. Resende, and A. Duarte, "Multiobjective grasp with path relinking," *European Journal of Operational Research*, vol. 240, pp. 54–71, Jul. 2014.
35. H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1305–1314, Sep. 2002.
36. L. Wang, A. Hoque, C. Yi, A. Alyyan, and B. Zhang, "Ospfn: An ospf based routing protocol for named data networking," Technical Report NDN-0003, Jul. 2012.
37. F. Guillemin, T. Houdoin, and S. Moteau, "Volatility of youtube content in orange networks and consequences," in *2013 IEEE International Conference on Communications (ICC)*, Jun. 2013, pp. 2381–2385.