Video-on-Demand Server Selection and Placement

Frederic Thouin and Mark Coates
McGill University
Department Electrical and Computer Engineering
3480 University, Montreal, Quebec, Canada H3A 2A7
{fthoui,coates}@ece.mcgill.ca

***Abstract*— Large-scale Video-on-Demand (VoD) systems with high storage and high bandwidth requirements need a substantial amount of resources to store, distribute and transport all of the content and deliver it to the clients. We define an extension to the *VoD equipment allocation problem* as determining the number and model of VoD servers to install at each potential replica location and the origin such that the deployment cost is minimized for a given set of distributed demand and available VoD server models. We propose three novel heuristics (GS, IRH and IGS) that generate near-optimal solutions (within 10% of the optimal solution). Simulations show that the number of replica sites for networks where the load is unevenly distributed is low $(35-45\%)$, but that the hit ratios at deployed replicas are high $(> 85\%)$.**

I. INTRODUCTION

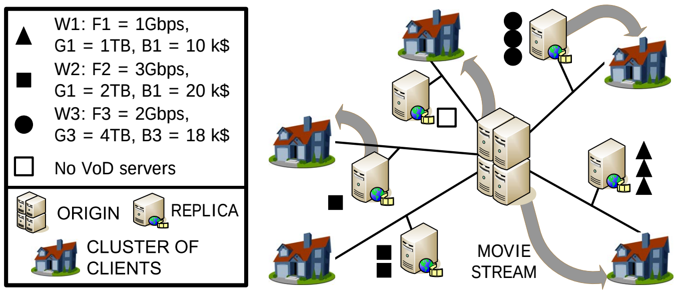
As the number of available titles and usage of video-on-demand services is expected to grow dramatically in the next years, many providers are planning the deployment of large-scale video-on-demand (VoD) systems. These systems require significant resources (bandwidth and storage) to store the videos, distribute them to caches, and deliver them to clients. An important and complicated task part of the network planning phase is resource allocation. It consists of determining the location and number of resources to deploy such that user demand is satisfied, cost is minimized, and any quality of experience (QoE) constraints (delay, packet loss, frame loss, or packet jitter) are respected. This operation is important because it is often very difficult (or even impossible) to substantially alter the chosen solution after the deployment. The main challenge is to build sufficiently accurate models for all of the factors involved: the available infrastructure, the network topology, the peak/average usage of the system, the popularity of each title, and bandwidth and storage requirements.

In the case of a distributed video-on-demand network deployment, the resources to consider are the equipment required at the origin and proxy video servers and for the actual transport between each location. We assume an existing topology with a high bandwidth capacity and focus on the equipment required at each location to store and stream the content. A video server consists of storage devices to cache the desired content and streaming devices to deliver the videos to the users. In [1], Thouin et al. defined the *VoD equipment allocation problem* that consists of determining the number of streaming and storage devices at each location in the topology such that the demand is satisfied and the deployment cost is minimized. They showed that the nature of the equipment installed at each location has a major impact on the design and on whether it is beneficial to even cache content. A natural extension of the problem thus involves identifying the best type of equipment to install at each location when many models are available and there is flexibility for variation from site to site. Therefore, in this paper, we address the problem of determining not only the number, but also the model of the VoD servers at each potential replica location.

This paper is organized as follows. In Section II, we formulate the VoD equipment allocation problem such that the solution includes both the number and model of the VoD servers. In Section III, we present two simple algorithms (Full Search and Centralized or Fully Distributed Heuristic) and three novel heuristics (Greedy Search, Integer Relaxation Heuristic and Improved Greedy Search) to solve the problem. In Section IV, we perform a complexity analysis of our algorithms. In Section V, we show and discuss the results of simulation experiments performed on randomly generated topologies. Finally, in Section VI, we present our conclusions and suggest future extensions to our work.

A. Related Work and Contribution

Researchers have tackled the problem of generating cost-efficient VoD network designs using different optimization techniques: placement of replica servers, video objects or allocation of available resources to minimize cost. Solving the replica placement [2], [3] or video placement [4] problems independently of the resource allocation problem usually leads to suboptimal solutions because the location of the replicas has a direct impact on the amount of resources required. Laoutaris et al. defined the *storage capacity allocation problem* as determining the location of each object from a set to achieve minimal cost whilst enforcing a capacity constraint [5]. Although they determine the actual storage requirements at each node with their solution, the authors do not explicitly determine the equipment required. In [6], Wauters et al. define an Integer Linear Programming (ILP) model built on viewing behavior, grooming strategies, statistical multiplexing and Erlang modeling to specify the equipment required for transport (the number of ports, multiplexers and switch ports) at each of the candidate network nodes [6]. Thouin et al. defined the *VoD equipment allocation problem* in [1] as the task of determining the number of VoD servers (which include both a storage and streaming device) to deploy at each potential location in a network topology such that the total demand is satisfied and the deployment cost is minimized. Solving the



W1: F1 = 1Gbps, G1 = 1TB, B1 = 10 k$
W2: F2 = 3Gbps, G1 = 2TB, B1 = 20 k$
W3: F3 = 2Gbps, G3 = 4TB, B3 = 18 k$
No VoD servers
ORIGIN
REPLICA
CLUSTER OF CLIENTS
MOVIE STREAM

Fig. 1. Video-on-Demand equipment allocation problem. Logical connectivity between origin, $N = 5$ replica servers and clients. Each replica represents a potential location to install VoD equipment in the network. Clients' requests (shown as movie stream arrows) are served by replicas when possible (if content is available) or by origin. Key shows the specifications (streaming and storage capacity and price) of $W = 3$ different VoD server models. We show the number and type of VoD servers installed at each potential location. The optimal solution can include locations with no equipment (empty square).

VoD equipment allocation problem determines the location of the replicas, the amount of storage available to cache content, the streaming capacity available to serve clients and the explicit specification of the equipment installed at each location. However, the approach in [1] assumed that a fixed, single and predetermined type of VoD server was available at each location. This constraint rarely holds in practice and enforcing it leads to suboptimal designs if the nature of the equipment is not a good fit to the streaming (user demand) and storage (library size) requirements.

In this paper, we re-formulate the VoD equipment allocation problem to determine both the number and model of the servers to install at each location. Instead of fixing the streaming and storage capacity per VoD server at each site (the approach used in [1]), we require the pre-specification of a set of available VoD servers and select the model at each location that minimizes total network cost. This leads to the faster generation of lower-cost solutions because the network designer does not need to manually try all models for each potential site. To solve the problem, we develop a network cost model solely in terms of the numbers and models of servers and propose three novel heuristics: the Integer Relaxation Heuristic and two greedy-search based algorithms (Greedy Search and Improved Greedy Search).

II. Problem Statement

We consider a metropolitan area network with one origin server and N potential replica locations such as the one depicted in Fig. 1. Each cluster of clients has worst-case demand M_i (peak usage demand) and is assigned to a potential replica location with hit ratio h_i. The hit ratio represents an estimate of the fraction of the demand M_i served at the replica, the other portion is served directly by the origin server.

We address the *VoD equipment allocation problem* of determining not only the number, but also the model of the VoD servers at each potential replica location. To solve this problem, we require the specification of a set of available VoD server models $\mathcal{W} = \{w_j : j = 1, \ldots, W\}$ where w_j is a VoD server with streaming capacity F_j Gbps, storage capacity G_j TB and unit cost B_j k\$. We define the sets $\mathcal{N} = \{n_i : i = 1, \ldots, N\}$ and $\mathcal{V} = \{v_o, v_i \in \mathcal{W} : i = 1, \ldots, N\}$ where n_i is the number and v_i is the model of the servers installed at location i. The optimization problem is expressed as follows:

$$\{\mathcal{N}^*, \mathcal{V}^*\} = \arg \min_{\mathcal{N},\mathcal{V}} C_{\text{TOT}_\mathcal{V}}(\mathcal{N}) \tag{1}$$

where $C_{\text{TOT}_\mathcal{V}}(\mathcal{N})$ is the total cost of the network C_{TOT} for a fixed set $\mathcal{V}$. We impose the following constraints:

$$\begin{aligned}
n_o \cdot G_o &\geq Y \cdot \text{file size} && (2)\\
n_i \cdot F_i &\geq h_i \cdot M_i, \quad \forall i \in \{1 \ldots N\} && (3)\\
n_o \cdot F_o &\geq \sum_{i=1}^{N} (1 - h_i) M_i && (4)\\
n_o \cdot F_o + \sum_{i=1}^{N} n_i \cdot F_i &\geq \sum_{i=1}^{N} M_i && (5)\\
\widehat{H}\left(\frac{n_i \cdot G_i}{Y \cdot \text{file size}}, Y, Z\right) &\geq h_i, \quad \forall i \in \{1 \ldots N\} && (6)
\end{aligned}$$

The first constraint states that the storage capacity at the origin must be large enough to host the entire initial library. The constraints in (3) and (4) ensure that the streaming capacity at each location and the origin is large enough to serve the fraction of the demand routed to this site. The constraint in (5) imposes the constraint that the total network streaming capacity is larger than the total demand from all locations. (6) states that the storage capacity at each location should be large enough to ensure the estimated hit ratio.

A. Network Cost Model

In order to perform a direct optimization, we derive an expression for the deployment cost solely in terms of $\mathcal{N}$ and $\mathcal{V}$. We begin by expressing the total cost C_{TOT} as the sum of the cost of infrastructure, C_T, and the cost of transport, C_S:

$$\begin{aligned}
C_T &= f_1(n_o, v_o) + \sum_{i=1}^{N} f_1(n_i, v_i)\\
C_S &= \sum_{i=1}^{N} f_2(h_i, M_i)\\
C_{\text{TOT}} &= f_1(n_o, v_o) + \sum_{i=1}^{N} f_1(n_i, v_i) + f_2(h_i, M_i) && (7)
\end{aligned}$$

The cost of infrastructure at each potential location and the origin includes a start-up cost for installation and software (A_i) and increases linearly with the number of VoD servers installed (n_i). Note that f_1 is also a function of v_i which defines B_i, F_i and G_i.

$$f_1(n_i, v_i) = A_i + B_i n_i \tag{8}$$

The cost of transport for each location includes transport from the replica to the clients ($C_{S_{RC_i}}$) and from the origin to the replica ($C_{S_{OR_i}}$). The cost of transport from the replica to the clients includes the cost of network interfaces (C_{IF}) and fiber (C_f). The number of network interfaces (n_{RC_i}) required is proportional to the demand M_i and the fiber capacity (c). The cost of transport from the origin to the replica location includes the cost of DWDM with w_{max}-ports multiplexers (C_{DWDM}), network interfaces (C_IF), fiber (C_f) and line amplifiers (C_{LA}). The number of network interfaces (n_{OR_i}) required is a function of the demand M_i and the hit ratio h_i: the amount of traffic on this link is equal to the fraction of the demand un-served by the replica. For more details on the cost functions f_1 and f_2, the reader is referred to [1].

$$\begin{aligned} f_2(h_i, M_i) &= C_{S_{OR_i}} + C_{S_{RC_i}} &(9)\\ C_{S_{RC_i}} &= n_{RC_i} \cdot (2 \cdot C_{IF} + d_{RC_i} \cdot C_f) &(10)\\ C_{S_{OR_i}} &= n_{OR_i}(2 \cdot C_{IF}) + \frac{n_{OR_i}}{w_{max}} \Big[2C_{DWDM} + \\ & \qquad d_{OR_i} \cdot C_f + \left(\frac{d_{OR_i}}{max_{amp}}\right) \cdot C_{LA}\Big] &(11) \end{aligned}$$

$$n_{OR_i} = \frac{(1-h_i) \cdot M_i}{c} \qquad n_{RC_i} = \frac{M_i}{c} \tag{12}$$

To derive an expression for C_{TOT} solely in terms of n_i for $i = 1, \ldots, N$, we resolve the hit ratio h_i and number of servers at the origin n_o as functions of n_i (M_i is a fixed parameter). We develop expressions to calculate h_i and n_o for a fixed $\mathcal{N}$. To estimate the hit ratio $\widehat{H}$, we use (13), a function of the cache size ratio X_i (number of files in the cache / number of total files in the library), the library size Y and the number of files added to the library every week Z. We designed the function and determined best-fit constants K_1 - K_8 using a discrete-time simulator based on the file access model proposed by Gummadi et al. in [7] (refer to [1] for more details).

$$\begin{aligned} \widehat{H} &= A(Y,Z) + B(Y,Z) \cdot \log(X) &(13)\\ A &= K_1 + K_2 Z + K_3 \log(Y) + K_4 Z \log(Y) &(14)\\ B &= K_5 + K_6 Z + K_7 Y + K_8 ZY &(15) \end{aligned}$$

The hit ratio at a location is limited by either the streaming or the storage capacity represented by constraints shown in (3) and (6). We isolate h_i in both expressions and define $f_3(n_i, v_i)$ as the minimum (worst-case) hit ratio:

$$f_3(n_i, v_i) = \min\left[\frac{n_i \cdot F_i}{M_i}, \widehat{H}\left(\frac{n_i \cdot G_i}{Y \cdot \text{file size}}, Y, Z\right)\right] \tag{16}$$

The number of servers required at the origin, n_o, is also constrained by either streaming or storage (shown in (4) and (2)). In (17), we define n_o as $f_4(\mathcal{N}, \mathcal{V})$ by substituting h_i with the expression in (16).

$$f_4(\mathcal{N}, \mathcal{V}) = \max\left[\frac{\sum_{i=1}^{N}(1-h_i) \cdot M_i}{F_o}, \frac{Y \cdot \text{file size}}{G_o}\right] \tag{17}$$

By replacing the equations for n_o and h_i in (7), we derive a new expression solely in terms of n_i:

$$C_{\text{TOT}} = f_1(f_4(\mathcal{N}, \mathcal{V})) + \sum_{i=1}^{N} f_1(n_i) + f_2(f_3(n_i, v_i)) \tag{18}$$

III. Description of Heuristics

A. Full Search (FS)

The Full Search is a very straightforward approach that consists of trying all the possible points in the solution space. We reduce this space by calculating the maximum number of servers it is worth installing at a given location using (19). We define **ub** $= \{ub_i : i = 1, \ldots, N\}$ where ub_i is the number of servers required to store the entire library and handle 100% of the requests ($h_i = 1.0$).

$$ub_i = \max\left(\frac{M_i}{F_i}, \frac{Y \cdot \text{file size}}{G_i}\right) \tag{19}$$

For a given $\mathcal{V}$, the boundaries of the solution space are $\mathcal{N} = \mathbf{0}$ to $\mathbf{ub}$ where $\mathbf{0} = \{n_i = 0 : i = 1, \ldots, N\}$. To complete the full search, all the possible combinations of $\mathcal{V}$ must also be tried. Although this procedure is guaranteed to find the optimal solution, it is very computationally expensive and the amount of time to search the entire space grows exponentially with the size of the network.

B. Central or Fully Distributed Heuristic (CoFDH)

```
1 $C_{central} = \infty$;
2 forall locations $i$ do /* centralized design, $\mathcal{N}_{central} = \mathbf{0}$ */
3     $n_i = 0$;
4 end
5 forall models $w_j \in \mathcal{W}$ do /* pick model at origin */
6     Set $\mathcal{V}'$: $v_i' = w_j$ for $i = 1, \ldots, N$;
7     calculate cost $C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}_{central})$;
8     if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}_{central}) < C_{central}$ then
      $C_{central} = C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}_{central})$ and $\mathcal{V}_{central} = \mathcal{V}'$ ;
9 end
```

Algorithm 1: Central Heuristic

```
1 $C_{FD} = \infty$;
2 forall models $w_j \in \mathcal{W}$ do
3     forall locations $i$ do
4         $v_i' = w_j$;
5         $n_i' = ub_i$ /* fully distributed, $\mathcal{N}' = \mathbf{ub}$ */;
6     end
7     calculate cost $C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}')$;
8     if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_{FD}$ then $C_{FD} = C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}')$,
      $\mathcal{N}_{FD} = \mathcal{N}'$, $\mathcal{V}_{FD} = \mathcal{V}'$ ;
9 end
```

Algorithm 2: Fully Distributed Heuristic

The Central or Fully Distributed Heuristic simply calculates the cost of a centralized design ($\forall i : n_i = 0$) and a fully distributed design ($\forall i : n_i = ub_i$) for each available VoD server model in $\mathcal{W}$ and picks the cheapest design. The Cental part of the heuristic is described in Algorithm 1; the Fully Distributed in Algorithm 2. This heuristic is straight-forward and highly suboptimal, but it provides an upper-bound that can be used as a comparison base for other approaches.

C. Greedy Search (GS)

```
Set $C_{GS} = \infty$, $\mathcal{N}_{GS}$: $n_i = 0$ and $\mathcal{V}_{GS}$: $v_i = w_1$ for $i = 1, \ldots, N$;
Set $C_0 = C_{GS}$, $\mathcal{N}_0 = \mathcal{N}_{GS}$, $\mathcal{V}_0 = \mathcal{V}_{GS}$ and $k = 0$;
repeat /* cost has not decreased for I iterations */
    k++;
    Set $C_k = \infty$, $\mathcal{N} = \mathcal{N}_{k-1}$ and $\mathcal{V} = \mathcal{V}_{k-1}$;
    forall locations $i$ do
        $\mathcal{N}' = \mathcal{N}$, $\mathcal{V}' = \mathcal{V}$;
        $n_i' = n_i + 1$ /* add one server at $i$ */;
        forall models $w_j \in \mathcal{W}$ do
            $v_o' = w_j$ /* model at origin */;
            forall models $w_k \in \mathcal{W}$ do
                $v_i' = w_k$ /* model at location $i$ */;
                calculate cost $C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}')$;
                if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_{GS}$ then $C_{GS} = C_{\text{TOT}_{\mathcal{V}'}}$, $\mathcal{N}_{GS} = \mathcal{N}'$, $\mathcal{V}_{GS} = \mathcal{V}'$ ;
                if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_k$ then $C_k = C_{\text{TOT}_{\mathcal{V}'}}$, $\mathcal{N}_k = \mathcal{N}'$, $\mathcal{V}_k = \mathcal{V}'$ ;
            end
        end
    end
until $C_j \geq C_{j-1}\ \forall j \in k - I + 1 \ldots k$;
```

Algorithm 3: Greedy Search (GS)

We define a search topology in the discrete solution space where each solution is connected to its neighbouring solutions. In this case, a neighbour consists of adding one server at one of the locations (and then potentially changing the server models of the origin and all other locations). Greedy Search (GS) is a searching heuristic that explores all neighbouring nodes and selects the one that yields the best solution at every iteration without considering the subsequent steps [8]. For each node, we try each of the N locations (lines 5-7 of Algorithm 3) and the different server models for both the origin server (lines 8-9) and the current location (lines 10-11). Therefore, each solution has NW^2 neighbours; we select the origin model v_o, the location i and the model at that location v_i that yield the lowest cost at each iteration. Note that with this procedure, the value of v_o and v_i can change at every iteration. We define $\mathcal{N} = \mathbf{0}$ as our initial solution, i.e., no servers installed at any of the locations and continue the search until it reaches a local maximum (or minimum); no neighbours offer a better solution than the current one.

To perform a more thorough search, we wait for more than one ($I = 3, 5, 10, 20, etc.$) iteration over which the cost does not decrease before stopping the search. Let C_k be the minimum cost after placing k servers (k iterations), then the search stops when $C_j \geq C_{j-1}\ \forall j \in k - I + 1 \ldots k$. To explore a larger part of the solution space, we perform two different greedy searches: one where servers are added to an initial solution $\mathcal{N} = \mathbf{0}$ and a second one that removes servers from an initial solution $\mathcal{N} = \mathbf{ub}$). For the second search, line 5 of Algorithm 3 becomes $n_i' = n_i - 1$. We then select the solution that produces the lowest cost.

D. Integer Relaxation Heuristic (IRH)

```
forall models $w_j \in \mathcal{W}$ do
    Set $\mathcal{V}_j$: $v_i' = w_j$ for $i = 1, \ldots, N$;
    Obtain $\mathcal{N}_j$ by performing a constrained nonlinear optimization on $C_{\text{TOT}_{\mathcal{V}_j}}$;
end
forall locations $i$ do
    Set $v_i = w_j$ and $n_i = n_j$ such that $C_{T_i} + C_{S_{OR_i}}$ is minimized;
end
Set $C_{IRH} = \infty$, $\mathcal{N}_{IRH} = \mathcal{N}$;
forall models $w_j \in \mathcal{W}$ do
    Set $v_o = w_j$;
    Calculate cost for $C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N})$;
    if $C_{TOT_{\mathcal{V}}} < C_{IRH}$ then $C_{IRH} = C_{\text{TOT}_{\mathcal{V}}}$, $\mathcal{V}_{IRH} = \mathcal{V}$
end
Set $C_0 = C_{IRH}$ and $k = 0$;
repeat
    $k++$;
    Set $\mathcal{N} = \mathcal{N}_{IRH}$;
    forall locations $i$ do
        Set $\mathcal{N}' = \mathcal{N}$ and $n_i' = 0$;
        Calculate cost $C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N}')$;
        if $C_{TOT_{\mathcal{V}}}(\mathcal{N}') < C_{IRH}$ then $C_{IRH} = C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N}')$, $\mathcal{N}_{IRH} = \mathcal{N}'$ ;
    end
    $C_k = C_{IRH}$;
until $C_k \geq C_{k-1}$ ;
Set $C_0 = C_{IRH}$ and $k = 0$;
repeat
    $k++$;
    Set $\mathcal{N} = \mathcal{N}_{IRH}$;
    forall locations $i$ do
        Set $\mathcal{N}' = \mathcal{N}$;
        for $k = n_i \pm 2$ do
            Set $n_i' = k$;
            Calculate cost $C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N}')$;
            if $C_{TOT_{\mathcal{V}}}(\mathcal{N}') < C_{IRH}$ then $C_{IRH} = C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N}')$, $\mathcal{N}_{IRH} = \mathcal{N}'$ ;
        end
    end
    $C_k = C_{IRH}$;
until $C_k \geq C_{k-1}$ ;
```

Algorithm 4: Integer Relaxation Heuristic (IRH)

The first step of the Integer Relaxation Heuristic presented in Algorithm 4 is to find an initial non-integer solution and the second step is to search its neighborhood for a near-optimal integer solution. In the first step (lines 1-13), we start by finding a non-integer solution for each server model using a constrained nonlinear optimization. Then, we calculate the cost associated with each replica ($C_{T_i} + C_{S_{OR_i}}$) and determine the model that minimizes this cost for each location. We complete the initial solution by determining the best server model to install at the origin (lines 9-13). In the second step (lines 14-42), we perform two different searches to find a near-optimal integer solution. In the first one (lines 14-26), we iteratively set $n_i = 0$ at each location to make sure it is profitable to setup a replica. The second search (lines 27-42) consists of iteratively trying to remove or add up to two servers at each location until we find a local minimum.

E. Improved Greedy Search (IGS)

```
1  Set $C_{IGS} = \infty$;
2  forall models $w_j \in \mathcal{W}$ do
3      Set $\mathcal{V}'$: $v'_i = w_j$ and calculate upper bounds $ub_i$ for $i = 1, \ldots, N$ ;
4      forall models $w_k \in \mathcal{W}$ do
5          Set $v'_o = w_k$;
6          Set $C_0 = \infty$ and $l = 0$;
7          repeat
8              $l{+}{+}$;
9              Set $\mathcal{N}' = \mathcal{N}$;
10             forall locations $i$ do
11                 Set $n'_i = ub_i$ and calculate $C_{\text{TOT}_{\mathcal{V}'}}(\mathcal{N}')$;
12                 if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_{IGS}$ then
                       $C_{IGS} = C_{\text{TOT}_{\mathcal{V}'}}$, $\mathcal{N}_{IGS} = \mathcal{N}'$,
                       $\mathcal{V}_{IGS} = \mathcal{V}'$ ;
13             end
14             Set $C_l = C_{IGS}$;
15         until $C_l \geq C_{l-1}$ ;
16     end
17 end
18 Set $C_0 = C_{IGS}$, $\mathcal{N}_0 = \mathcal{N}_{IGS}$ and $\mathcal{V}_0 = \mathcal{V}_{IGS}$;
19 repeat /* cost has not decreased for I
   iterations */
20     k++;
21     Set $C_k = \infty$, $\mathcal{N} = \mathcal{N}_{k-1}$ and $\mathcal{V} = \mathcal{V}_{k-1}$;
22     forall locations $i$ do
23         for $m = -1$ and $m = 1$ do
24             Set $\mathcal{N}' = \mathcal{N}$ and $n'_i = n_i + m$;
25             calculate cost $C_{\text{TOT}_{\mathcal{V}}}(\mathcal{N}')$;
26             if $C_{TOT_{\mathcal{V}}}(\mathcal{N}') < C_{IGS}$ then $C_{IGS} = C_{\text{TOT}_{\mathcal{V}'}}$,
               $\mathcal{N}_{IGS} = \mathcal{N}'$, $\mathcal{V}_{IGS} = \mathcal{V}'$ ;
27             if $C_{TOT_{\mathcal{V}}}(\mathcal{N}') < C_k$ then $C_k = C_{\text{TOT}_{\mathcal{V}'}}$,
               $\mathcal{N}_k = \mathcal{N}'$, $\mathcal{V}_k = \mathcal{V}'$ ;
28         end
29     end
30 until $C_j \geq C_{j-1}$ $\forall j \in k - I + 1 \ldots k$ or $C_j \geq C_{IGS}$
   $\forall j \in k - 2I + 1 \ldots k$;
```

Algorithm 5: Improved Greedy Search (IGS)

As in the Integer Relaxation Heuristic, the Improved Greedy Search is divided into two steps: determining an initial solution and searching its surroundings for a better one. In IGS, both steps are inspired by the greedy search. During the first step of the heuristic (lines 7-17 of Algorithm 5), we iteratively add servers in a greedy-fashion starting from a centralized design by setting $n_i = ub_i$ at the location that achieves the lowest cost. We repeat this process of adding ub_i servers at a chosen location such that cost is minimized after each iteration, until it is no longer possible to decrease the cost. This first step is repeated for each VoD server model at the origin and the other locations (lines 1-6) and at that point, we have determined an initial integer solution and the first step is complete. The second step (lines 20-36), just like in the Integer Relaxation Heuristic, is an exploration procedure in the neighbourhood of the initial solution. In a greedy-type approach, at iteration k we add or remove one server to the initial solution at the location that minimizes the cost C_k. We stop the search when $C_j \geq C_{j-1}$ $\forall j \in k - I + 1 \ldots k$ or when $C_j \geq C_{IGS}$ $\forall j \in k - 2I + 1 \ldots k$ (minimum cost has not decreased for 2I iterations). Because we increase and decrease the number of servers, some solutions can be revisited during the searching procedure. For that reason, we add the second termination condition to guarantee the convergence of the heuristic (to avoid a loop in the solution space topology).

IV. Complexity Analysis

In this section, we analyze the worst-case complexity, WCC, of each of the heuristics presented in the previous section. We define the worst-case complexity as the maximum number of operations that the heuristics can perform before terminating. The expressions presented are functions of the number of locations N, number of VoD server models W and the maximum of all upper bounds ub_i, $U_{max} = \max(\mathbf{ub})$. To further simplify these expressions, we assume that $ub_i = U_{max}$ for all locations; this is reasonable for a network where the demand is distributed evenly among all the locations.

A. FS

In the full search, all models must be evaluated at all locations (W^N) for all the possible number of servers ($\prod_i^N ub_i$). When we assume $ub_i = U_{max}$ for all locations, the maximum number of iterations for FS is:

$$\begin{aligned} WCC_{FS} &= W^N \prod_i^N ub_i = W^N \cdot {U_{max}}^N \\ &= (W \cdot U_{max})^N \end{aligned} \tag{20}$$

This expression is not the worst-case scenario, but the actual number of iterations for every search. It is exponential in the size of the network, N, indicating that it is impractical to use this method for most scenarios.

B. CoFDH

CoFDH was developed to generate an upper-bound and a baseline solution for assessing the performance of the other heuristics. It is trivial and has low complexity; running either the central or the fully distributed heuristic only requires a number of iterations equal to W because the value of $\mathcal{N}$ is either $\mathbf{0}$ (centralized) or $\mathbf{ub}$ (fully distributed).

$$WCC_{CoFDH} = 2W \tag{21}$$

C. GS

One iteration of Algorithm 3 consists of trying each model at each location and the origin: $N \cdot W^2$ operations. The worst-case scenario is that the best solution is a fully distributed design ($n_i = ub_i$ for all locations) which requires $\sum_i^N ub_i$ iterations if the algorithm reaches that solution.

$$\begin{aligned} WCC_{GS} &= \sum_i^N ub_i \cdot (N \cdot W^2) = (N \cdot U_{max}) \cdot (N \cdot W^2) \\ &= N^2 W^2 U_{max} \end{aligned} \tag{22}$$

Under our simplifying assumptions, the complexity of GS is a second degree polynomial in N and W and linear in U_{max}.

D. IRH

The first step of IRH consists of performing a constrained nonlinear optimization for each VoD server model. This type of optimization is performed using a sequential quadratic programming (SQP) [9], [10] algorithm which has a complexity of $\mathcal{O}(N^2)$. With W more operations, we determine the model at the origin. The first part of the searching step (lines 11-18 of Algorithm 4) requires going through each location once until the cost does not decrease. The worst-case scenario is starting from a solution $\mathcal{N}$ with $n_i \neq 0$ for all locations and finishing with $\mathcal{N} = \mathbf{0}$; this requires up to N iterations. In the second part, each iteration requires five operations (trying solutions in the range of $n_i \pm 2$) for each location. The worst-case number of iterations is $\sum_i^N ub_i$ if we start from $\mathcal{N} = \mathbf{0}$ and terminate the search with $\mathcal{N} = \mathbf{ub}$ or vice-versa.

$$\begin{aligned} WCC_{IRH} &= (W \cdot N^2 + W) + (N^2) + 5N \sum_i^N ub_i \\ &= N^2(W + 1 + 5U_{max}) + W \end{aligned} \tag{23}$$

E. IGS

Each iteration of the first step of IGS has the same complexity as a GS iteration, but the maximum number of iterations is N because we add ub_i servers at a time instead of one. In one iteration of the searching phase, two operations are performed for each location. The worst-case is the same as that described in IRH.

$$\begin{aligned} WCC_{IGS} &= (W^2 N^2) + 2N \sum_i^N ub_i \\ &= (W^2 N^2) + 2N^2 U_{max} \end{aligned} \tag{24}$$

F. Worst-case complexity comparison

We complete our analysis by comparing the WCC of all the heuristics described in this section. The full search is unsuitable for our problem; even small problems such as $N = 5$, $W = 6$ and $U_{max} = 20$ take on the order of 10^9 operations. For complex problems like $N = 100$, $W = 6$ and $U_{max} = 20$, the WCC of IRH (270k oper.) and IGS (760k oper.) is lower than GS (7.2M oper.), but the actual computational times differ due to the running time of each operation. In the next section, we measure the actual CPU time used by each heuristic during our simulations.

V. Simulation Experiments

In this section, we present our simulation results obtained by applying our heuristics to different networks. Each test network is defined by the constant variables in Table I and choosing values for the other network parameters from uniform distributions with the ranges specified in Table I. Simulations were executed on a AMD Athlon 3000+ with 1 GB of OCZ Premier Series 400 MHz Dual Channel memory.

TABLE I
Values and range of variables used for the simulations.

Variable	Value	Variable	Min	Max
C_{IF}	10 k\$	d_{OR} (km)	0	50
C_{DWDM}	25 k\$	d_{RC} (km)	0	5
C_{LA}	10 k\$	Y (files)	1000	10000
C_f	0.006 k\$/km	Z (files/week)	0	100
w_{max}	16	priceGbps (k\$/Gbps)	0	4
c	10 Gbps	priceTB (k\$/TB)	0	3
max_{amp}	75 km	A (k\$)	6	36
bit rate	3.75 Mbps	F (Gbps)	1	5
duration	5400 s	G (TB)	1	11
file size	2.53 GB	M (Gbps)	1	20

In our first set of tests, we generated networks with the number of locations $N \in \{1, \ldots, 5\}$ and the number server model $W = 1$ and another series with $N = 3$ and $W \in \{1, 2, 3\}$. We choose small networks to allow comparison with the full search, which cannot produce a solution for larger networks in a reasonable time frame.

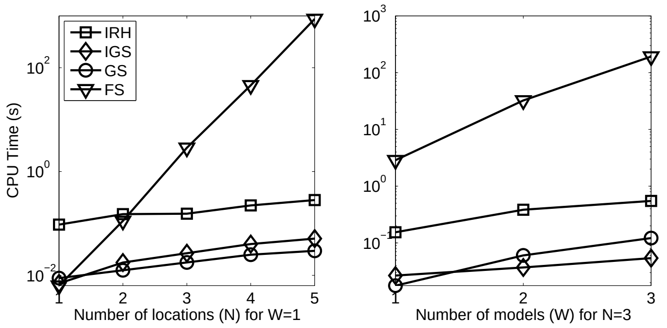


Fig. 2. Computational time in seconds required to find a solution by each of the heuristics averaged over 30 runs shown on a log-scale. Computational time of Full Search (FS) grows exponentially with the size of the network. Greedy Search (GS), Integer Relaxation Heuristic (IRH) and Improved Greedy Search (IGS) all provide solutions within 0.1 seconds.

In Fig. 2, we show the computational time in seconds on a log-scale averaged for 30 different networks with the same N and W. In both plots, we see the exponential behavior of the full search whereas the other heuristics show a very small increase in CPU time. We note that the computational time of the greedy-based heuristics (GS and IGS) is one order of magnitude lower than the integer relaxation approach, but both are nevertheless below 0.1 seconds for the simulated networks.

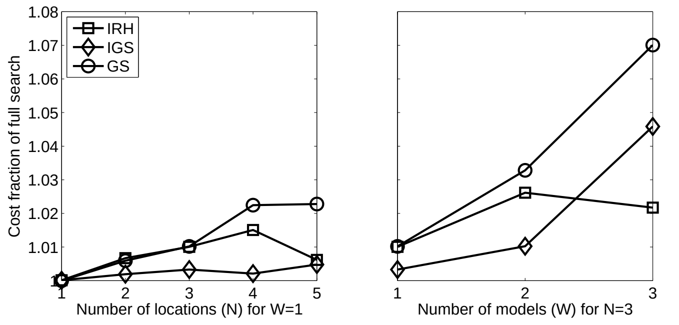


Fig. 3. Cost ratio between heuristics and the full search averaged over 30 runs.

In Fig. 3, we show the performance of our heuristics by dividing the cost of the solution by the optimal solution provided by the full search. For these small networks, Integer Relaxation Heuristic and Improved Greedy Search perform within 4% of the optimal solution. For all values of N and W, both IRH and IGS perform better than the Greedy Search, which is within 8% of the Full Search solution.

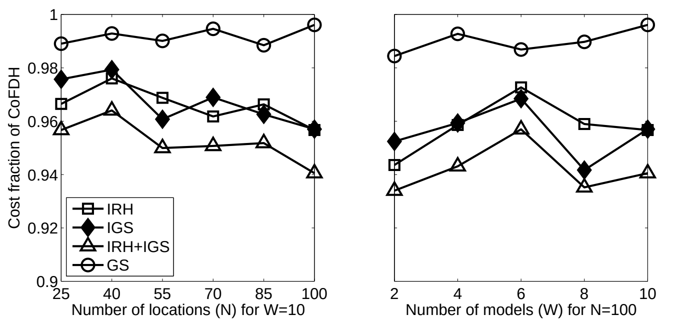


Fig. 4. Cost ratio between the heuristics solution and CoFDH averaged over 25 runs. IRH+IGS is the average of the minimum value between IRH and IGS for all runs.

In this next set of tests, we compare the complexity and the performance for networks with $N = 25$ to 100 potential replica locations and $W = 2$ to 10 server models. We use Central or Fully Distributed Heuristic (CoFDH) to measure the performance of our heuristics because it is impossible to determine the optimal solution with the Full Search.

In Fig. 4, we show values (averaged over 25 runs) of the ratio between the cost of Integer Relaxation Heuristic, Improved Greedy Search and Greedy Search and the cost of CoFDH. Whereas Greedy Search is actually very close to the cost produced by CoFDH, the other two heuristics generate solutions that cost 2-5% less. It is not clear from those plots whether Integer Relaxation Heuristic or Improved Greedy Search performs better. By combining both (choosing the best solution of the two), we obtain a better heuristic (IRH+IGS), which achieves a 4-6% cost reduction compared to CoFDH. In the left panel, we notice the downward trend of the cost fraction as the number of locations in the network increases because more modifications to the CoFDH design can be made to reduce cost.

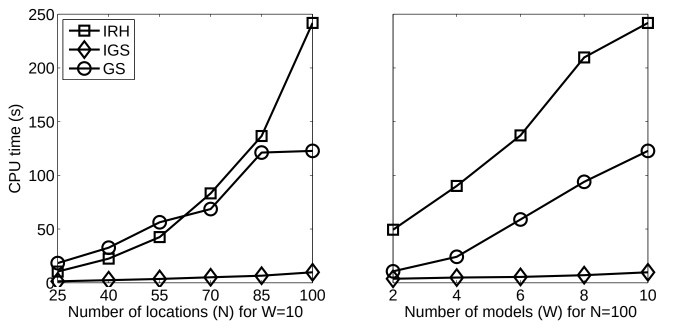


Fig. 5. CPU time in seconds for all heuristics averaged over 25 runs. IRH+IGS is the average of the sum of the time taken to perform both searches.

For the same set of tests, we also show the complexity expressed as the computational time in seconds in Fig. 5 and the number of iterations (cost function evaluations) to obtain a solution in Fig. 6. As suggested by the Worst-Case Complexity analysis in section IV, the greedy search (GS) takes many more iterations to find a solution than our other two heuristics Integer Relaxation Heuristic and Improved Greedy Search. However, although the number of iterations for Greedy Search is much larger than for IRH, their computational times are comparable in the left panel of Fig. 5. This is an indication that IRH's iterations take more time to execute than those in the greedy approaches (GS and IGS).

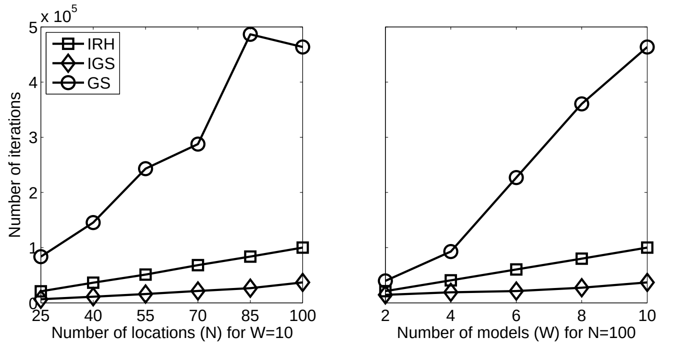


Fig. 6. Number of iterations averaged over 25 runs. IRH+IGS is the average of the total number of iterations performed in both searches.

The Integer Relaxation Heuristic was the slowest of the tested heuristics, but it still converges in a reasonable amount of time. Since the computation time of Improved Greedy Search is so low, we can combine IRH and IGS and obtain a solution in a timely fashion.

Finally, we focus on the networks with six server models (similar behaviour was observed for other values of W) to analyze the hit ratio, ratio of locations with replicas, average demand at replica locations and load on the origin server. Fig. 7 and Fig. 8 show the results and provide interesting insights on the solution generated by the heuristics. The left panel of Fig. 7 shows that for networks of any size where demand is not uniformly distributed among all locations (i.e,

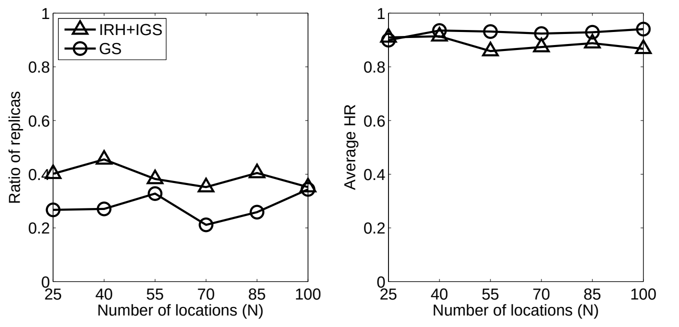


Fig. 7. LEFT: Ratio between the number of replicas (location with cached content) and potential locations. RIGHT: Average hit ratio at locations with cached content. Values are averages of 25 runs with $W = 6$.

the demand at each location is different), the percentage of locations where a replica will be deployed is below 40% for both heuristics. Although a case where the demand load is evenly shared among all the locations (all M_i are approximately equal) is more plausible, this result indicates that it is not always advantageous to cache content. If the demand is too low or the site is too close to the origin, then it can be more cost-effective to assume the entire load from a group of clients directly at the origin. An impact of this low percentage is shown in Table II where we show the number of servers installed at the origin. Because the fraction of locations where replicas are installed remains approximately constant for any value of N, the total number of sites for which the origin must assume the demand grows as the network becomes larger.

TABLE II
AVERAGE NUMBER OF VOD SERVERS AT THE ORIGIN FOR DIFFERENT NUMBER OF LOCATIONS N WITH GREEDY SEARCH (GS) AND INTEGER RELAXATION HEURISTIC + IMPROVED GREEDY SEARCH (IRH+IGS)

N	25	40	55	70	85	100
IRH+IGS	56	85	131	156	175	192
GS	81	109	133	190	244	223

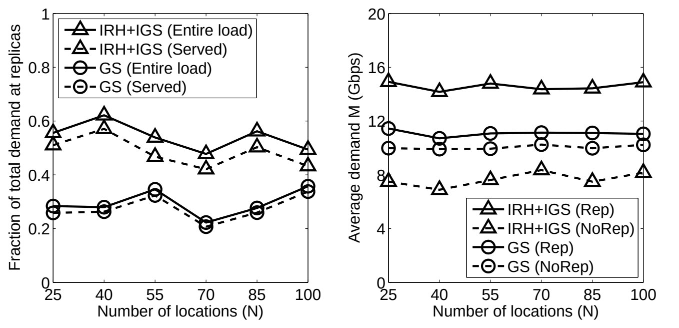
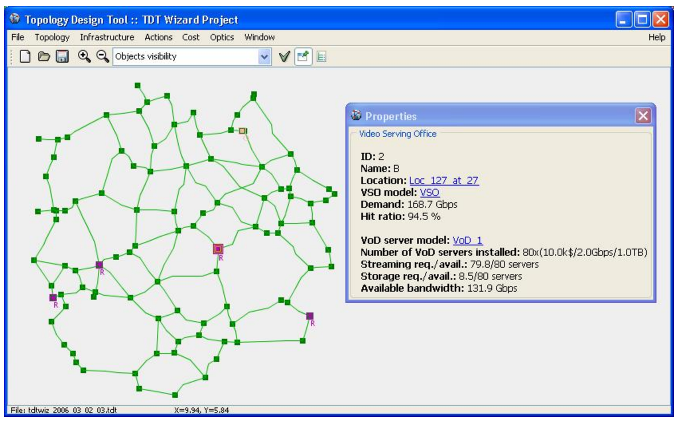


Fig. 8. LEFT: Fraction of the total network demand supported by replica locations. Total is the sum of the demand M_i at each location where a replica is installed and Real is the actual part of the demand that the replica handles $(M_i \cdot h_i)$. RIGHT: Average load on the locations where replicas are installed (Rep) and where no replicas are installed (NoRep). The values shown are averages of 25 runs with $W = 6$.

In the right panel of Fig. 7, we display the averaged hit ratio at all the locations where content was cached. The average hit ratio of 90% suggests that the optimal number of servers to install at a replica is often very close to ub_i. This is explained by both our popularity model and the ratio between the startup cost of a location (A) and the cost incurred in transportation to the origin. From our popularity model, we know that it is possible to achieve a high hit ratio with a relatively small amount of storage. Depending on the actual demand and the type of server installed, the streaming capacity is usually the limiting factor, which means that storage is often available to increase the hit ratio to the values we observe in this plot.

We display the fraction of the total network demand at the replica locations in the left panel of Fig. 8. We show two lines for each heuristic: the sum of the demands M_i at each location where a replica is installed (Entire Load) and the actual part of the load $(M_i \cdot h_i)$ handled by the replica (Served). For both Greedy Search and the best of Integer Relaxation Heuristic and Improved Greedy Search (IRH+IGS), the performance is very similar as a result of the high average hit ratio ($\approx 90\%$). We compare this ratio with the fraction of replicas in the network (left panel of Fig. 7). For GS, the difference is not significant, but in the case of IRH+IGS the percentage of the network load handled at replicas is approximately ten-twenty percent higher. This signifies that the locations chosen by IRH+IGS to host replicas generally have a high demand. This interpretation is confirmed in the right panel of Fig. 8 in which we depict the difference between the average demand at replica locations and locations where no caching is performed. Whereas there is only a marginal difference in the GS case, the average demand at replica sites in the IRH+IGS solutions is almost twice the average demand of the other locations. The solutions generated by combining Integer Relaxation Heuristic and Improved Greedy Search have a much lower total cost than the GS solutions, indicating that it is more cost-efficient to install replicas at locations where demand is high and transport the entire load of locations with low demand to the origin.

VI. CONCLUDING REMARKS



Video Serving Office

ID: 2
Name: B
Location: Loc_127_at_27
VSO model: VSO
Demand: 168.7 Gbps
Hit ratio: 94.5 %

VoD server model: VoD_1
Number of VoD servers installed: 80x(10.0k$/2.0Gbps/1.0TB)
Streaming req./avail.: 79.8/80 servers
Storage req./avail.: 8.5/80 servers
Available bandwidth: 131.9 Gbps

Fig. 9. Screenshot of the design tool that implements our heuristic (IRH+IGS) to solve the VoD equipment allocation problem. A sample topology of potential replica locations and the properties window of a selected replica location is shown in the figure.

In this paper, we defined an extension of the *VoD equipment allocation problem* described in [1]. Instead of considering fixed and pre-determined streaming and storage capacity at each location, we require the specification of a set of available VoD servers models. The optimization problem consists of choosing the number and type of VoD servers to install at each potential location in the network such that cost is minimized. We modified the total cost expression to make it a function of the number of servers n_i instead of the cache size ratio X_i. Solving this problem with a complete search is possible, but for networks of more than five locations and a set of available models larger than three the computational requirements render the approach impractical. We described three heuristics to find a near-optimal solution including two greedy-type approaches (GS and IGS) and an integer relaxation method (IRH) that we implemented in an interactive design tool shown in Fig. 9. The Improved Greedy Search has very low complexity in practice (less than half a minute and 50,000 iterations for large networks), but does not always provide a better solution than the Integer Relaxation Heuristic. We showed that it is possible to combine both by choosing the best of the two to obtain a better solution while maintaining the computational time reasonably low (slightly more than four minutes and 150,000 iterations on average for large networks). Depending on the context, two heuristics are available: Improved Greedy Search for a very quick solution (few seconds) or combining IRH and IGS for a better solution that takes more time to produce.

For all our simulations, we generated network topologies where the load was different at each location. For such networks, we observed that the fraction of locations where it was cost-efficient to install replicas was small (35-45% depending on network size). In the optimal solutions produced by our heuristic IRH+IGS, the average worst-case demand at replica locations is approximately 15 Gbps and 8 Gbps at locations where the entire load is transported to the origin server. For networks with 100 locations, the replica sites assume less than 45% of the total network load which results in a very large number (almost 200) of required servers at the origin that might be impossible to deploy in practice. Our simulations indicate that the average hit ratio at the replica sites is above 85% for all network sizes. This suggests that it is possible to have a cost-efficient solution with a higher fraction of the network load handled at replicas and much reduced load at the origin. A way to obtain such a solution is by using equipment (VoD server model) that satisfies the streaming and storage requirements of most of the locations in the topology. Alternatively, the network designer could strive to divide the demand evenly among all locations such that it is optimal to deploy replicas at most locations using the same model of equipment. A sensible extension to the resource allocation problem we addressed in this paper is the problem of jointly designing the VoD network and the logical topology. It consists of choosing a topology that allows an allocation of resources which minimizes the deployment cost of the network.

We have considered only the scenario where the service provider does not own any network equipment or infrastructures prior to the deployment. However, this is not always the case because some provider might be able to transport data for free, i.e., no need to install fiber, network interfaces, switches, or amplifier. Even if there is no installation cost, there are still fees incurred by the usage and maintenance of the equipment and the resources, which have to be considered when generating solutions for this scenario. Also, we focused on large-scale deployments, but there is also the issue of scalability of such deployments. As the library reaches tens of thousands of movies, the access model we assumed changes as a larger portion of requests are located in the heavy tail of the popularity distribution ('long-tail' of content). It is unclear if this simply shifts the hit ratio curve down (more storage needed to achieve the same hit ratio) or the function would be completely different. The growth in usage also affects the design. During our simulations for the hit ratio function, we determined that the impact of the varying number of users on the hit ratio is not significant. Even if the storage requirements are not affected, the higher loads at each location and on the origin server require more streaming capacity. In that case, it is sensible to impose a constraint on the maximum number of servers at the origin to avoid a high load on one location (or alternatively impose a minimum hit ratio at each replica). The reason we chose not to include these constraints in our initial problem statement was to allow a maximum number of valid solutions. Producing the most cost-efficient solution, whether it is feasible in practice or not, provides important feedback on the design choices of the network planner. From our results, an infeasible design is an indication that the equipment was a mismatch for the given topology or, alternatively, the chosen topology was not optimal for the available equipment.

References

[1] F. Thouin, M. Coates, and D. Goodwill, "Video-on-demand equipment allocation," in *Proc. IEEE Int. Conf. Network Computing Applications (NCA)*, Cambridge, MA, July 2006.

[2] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Networking*, vol. 8, pp. 568–582, Oct. 2000.

[3] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Trans. Multimedia*, vol. 6, pp. 356–365, April 2004.

[4] W. Tang, E. Wong, S. Chan, and K. Ko, "Optimal video placement scheme for batching vod services," *IEEE Trans. on Broadcasting*, vol. 50, pp. 16–25, Mar. 2004.

[5] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis, "On the optimization of storage capacity allocation for content distribution," *Computer Networks Journal*, vol. 47, pp. 409–428, Feb. 2005.

[6] T. Wauters, D. Colle, M. Pickavet, B. Dhoedt, and P. Demeester, "Optical network design for video on demand services," in *Proc. Conf. Optical Network Design and Modelling*, Milan, Italy, Feb. 2005.

[7] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. ACM Symp. Operating Systems Principles (SOSP)*, Bolton Landing, NY, Oct. 2003.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1990.

[9] R. Fletcher, *Practical Methods of Optimization*. New York, NY: John Wiley and Sons, 1987.

[10] W. Hock and K. Schittkowski, "A comparative performance evaluation of 27 nonlinear programming codes," *Computing*, vol. 30, pp. 335–358, 1983.