Video-on-demand Equipment Allocation

Frédéric Thouin

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

July 2006

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Masters of Engineering.

Abstract

Network-based video-on-demand (VoD) deployments are today very limited in scope. The largest deployed libraries are just 0.7% of the global movie and TV-series catalog and peak utilization of VoD targets are 10-15% of broadcast TV peak viewing numbers. Recognizing that libraries and usage may grow, service providers are intensely interested in large-scale content delivery networks that provide content propagation, storage, streaming, and transport. We focus on one of the challenges of VoD network design: resource planning. We describe a method and design tool for the planning of large-scale VoD systems and address the resource allocation problem of determining the number and model of VoD servers to install in a topology such that the deployment cost is minimized. Our general design tool provides important feedback and insights on VoD network design; we observed that the available equipment and the topology had a significant impact on the resulting design [1].

Abrégé

Le contenu disponible des services de vidéo à la demande (VoD) en place représente 0.7% de la totalité des films et séries télévisées et le nombre d'utilisateurs aux heures de pointe représente 10-15% de l'auditoire d'émissions télévisées. Prenant conscience de l'expansion imminente de l'utilisation et de la bibliothèque, les fournisseurs de service s'intéressent aux réseaux de diffusion de contenu (CDN) de grande échelle qui offre la propagation, le stockage, la lecture en transit et le transport du contenu. Nous développons une méthode et un outil pour la planification de systèmes de VoD de grande échelle et s'attaquons au problème d'attribution de ressources suivant: déterminer le nombre et le modèle de serveurs VoD à installer à chaque localisation pour minimiser le cout de déploiement. Nos résultats de simulation démontrent que le type d'équipement disponible ainsi que la topologie du réseau ont une grande influence sur le design final.

Acknowledgments

First and foremost, I would like to thank my supervisor Mark Coates for his help and guidance throughout my two years as a Masters student. I also like to thank Dominic Goodwill for his contribution through a Partnered Research Project with Nortel Networks. The feedback from the industry world he provided was very helpful. I thank Lorne Mason and Ning Zhao for their help designing the function for the cost of transport. I would like to thank Anton Vinokurov, the developer of the Topology Design Tool (TDT), who helped me integrate the various components of my solution into his tool. I would like to thank my fellow lab-mate Garrick Ing for letting me benefit from his previous experiences as a Graduate student.

Je tiens à remercier mes parents pour leur support et encouragements durant mes deux années à la maîtrise.

Contents

List of Figures

List of Tables

List of Acronyms

CDN	Content Delivery (Distribution) Network
VoD	Video-on-demand
QoS	Quality of Service
QoE	Quality of Experience
MAN	Metropolitan Area Network

Chapter 1

Introduction

Network-based video-on-demand (VoD) deployments are today very limited in scope. The largest deployed libraries are just 0.7% (5,000 hours) of the global movie and TV-series catalog and peak utilization of VoD targets are 10-15% of broadcast TV peak viewing numbers. Recognizing that libraries and usage may grow, service providers are intensely interested in large-scale content delivery networks that provide content propagation, storage, streaming, and transport. Content delivery networks (CDNs) are designed to distribute content to a set of clients, as streams or as files [2–7]. We focus primarily on the case of a streaming CDN, in that the client is assumed to have buffering capability but not caching capability. Nonetheless, the mathematics and the models we adopt are readily extended to the client-cached scenario. Through approaches such as replication of content at multiple servers (known as replicas, proxies or caches), CDNs attempt to minimize latency at the end-user while reducing bandwidth consumption and load at the origin server. The CDN delivery of streaming media causes new problems that did not apply to the distribution of HTTP objects: streaming objects are much larger than web objects and hence create much more traffic [8]. Furthermore, it is no longer possible to assume infinite storage size at the replica locations, which makes calculations more complicated [2]. The design of a VoD network consists of two tasks: (i) making resource planning decisions and (ii) developing in-service intelligent request routing, resource control policies, and performance monitoring. We focus on the first challenge, i.e., the allocation of resources during network planning, generally performed when planning greenfield and incremental deployments. Of particular interest is VoD delivery across metropolitan area networks (MANs).

1.1 Motivation

In this thesis, we describe a method and design tool for the planning of large-scale VoD systems. Before describing the method, it is useful to underline the scale of the problem. Today's libraries of consumer-accessible media vary widely in size. We compare the total stock of media content (listed in Table 1.1) against the library sizes as advertised by service providers. Music services (e.g., Apple iTunes) offer around 10% of existing music. Similarly, services of movies on DVD by mail (e.g., NetFlix) offer around 10% of movies and TV series. In contrast, the largest existing network VoD systems (e.g., Comcast) offer only 0.7% (several thousand) of the movies and TV series genres, and only 0.002% of the back catalog if broadcast TV is considered to be within the scope of VoD libraries.

Table 1.1 Amount of unique stock of media content produced annually, and the accumulated unique stock. [9].

Type of content	Unique stock per year	Accumulated unique stock
Movies except TV movies	7350 titles (19TB)	250,000 titles(720 TB)
TV movies & series	3040 series titles (38TB)	62,000 series titles (950TB)
All forms of TV	31M hrs (70,000TB)	≫100M hrs (≫200,000TB)
rf broadcast radio	70M hrs (3,500TB)	≫200M hrs (≫10,000TB)
Professional music recording	90,000 albums (59TB)	1,5M albums (975TB)

The amount of video content produced annually (over 10,000 movie and series titles) and size of the libraries of other media providers indicate that the amount of content available to video-on-demand users will probably grow in the future. Even if this growth is only a few percent, considering the accumulated amount of unique video titles, we expect that an expansion of the library would make video-on-demand a real alternative to services like DVD by mail and attract more users. Large-scale VoD systems with high storage and high bandwidth requirements require a substantial amount of resources to store, distribute and transport all the content and deliver it to all the clients. At a time where many companies are considering deploying such large-scale systems, there is a real need for a design tool used during the network planning.

Resources allocation is an important and complicated task that consists of determining the location and number of resources to deploy such that cost is minimized whilst certain conditions are respected. This operation is important because it is often very difficult or impossible to adjust the chosen solution based on observations made after the deployment.

The main challenge is to build accurate models for all the factors involved: the available infrastructure, the network topology, the peak/average usage of the system, the popularity of each title, bandwidth and storage requirements, etc.

1.2 Thesis Problem Statement

In the case of a video-on-demand network deployment, the resources to consider are the equipment required at the origin and proxy video servers and the equipment required for the actual transport between each location (switching). 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. For this thesis, we define and tackle 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 deployment cost is minimized.

1.3 Thesis Contribution and Organization

In **Chapter 2**, we review the different aspects related to the delivery of multimedia objects: architecture and topology, caching scheme and file popularity model, delivery mechanism and traffic modeling. Also, we present a summary of the solutions proposed in the literature: problem statements, parameters and constraints considered for the cost function and heuristics proposed to solve the replica placement problem.

In **Chapter 3**, we address a simplified *VoD equipment allocation problem*, which focuses on identifying the optimal number of VoD servers at a set of locations with fixed and predetermined streaming and storage capacity per VoD server, such that the deployment cost of the VoD system is minimized. Our main contributions to solving this problem are the following. We design a parametric function for estimating the worst-case hit ratio for given system parameters (cache size, library size and file arrival rate). We determine an appropriate functional form and train parameters using discrete-time simulations based on an extension of the file access model proposed in [10]. Such a parametric function is essential for the *interactive* design tool we develop (see Section 3.2.1). We propose a cost function based on the hit ratio, the distributed demand and the number of VoD servers at each location. This differs from previous work that only takes into account transport and storage costs without explicitly considering the cost and type of equipment installed (see Section 3.2.2). We develop the Integer Relaxation Heuristic to generate a solution to the problem. The heuristic relaxes the integer constraint on the number of devices in order to identify an optimal non-integer solution, and then finds a near-optimal integer solution by searching in the neighborhood of the non-integer solution (see Section 3.2.3). We develop an *interactive* design tool that implements our cost function, hit ratio function and heuristic. This tool allows a user to modify system parameters easily and generate new design solutions quickly.

In **Chapter 4**, we extend the work of Chapter 3 by addressing the problem of determining not only the number, but also the model of the VoD servers at each potential replica location. Instead of fixing the streaming and storage capacity per VoD server at each site, 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 new problem being of greater complexity, we adapt the Integer Relaxation Heuristic and present another algorithm: the Improved Greedy Search. Finally, we briefly analyze the feasibility and implications of a VoD deployment over an agile all-photonic network (AAPN) [11]

In **Chapter 5**, we summarize our work and discuss in more detail the results presented in the previous chapters and conclude with proposed future work.

1.4 Published Work

Some parts of this thesis have been published or have been accepted for publication. Parts of the literature review presented in Chapter 2 has been published as a technical report and a summarize version of Chapter 3 on our solution to the *VoD equipment allocation problem* has been accepted for presentation at the Symposium on Network Computing and Applications (IEEE NCA).

- F. Thouin and M.J. Coates, A review on content delivery network, Technical report, McGill University, Montreal, Canada. June 2005.
- F. Thouin, M.J. Coates and D. Goodwill, Video-on-demand Equipment Allocation, to appear in Proc. IEEE Network Computing and Applications (IEEE NCA), Boston, MA. July 2006.

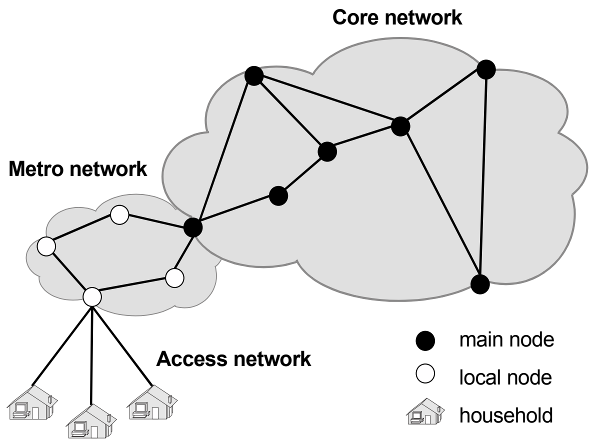
Chapter 2

Literature Review

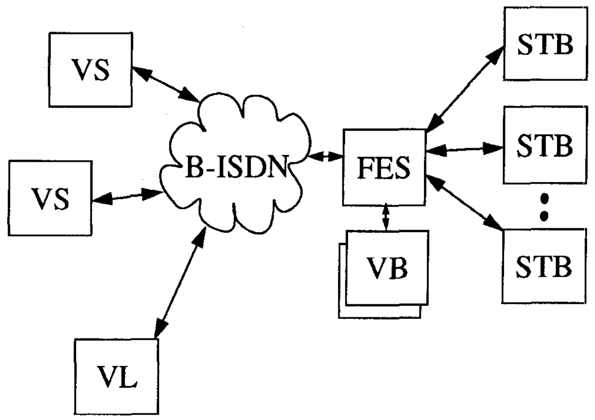
Services with low system cost, like near video-on-demand (nVoD), have been available for many years in hotels and offered by cable providers as pay-per-view television. In nVoD, at fixed scheduled times, a video is broadcast on a single channel shared by all the users who wish to see it. This solution is simple and cost-effective, but it is not flexible and does not allow the user to interact with the system [12, 13]. True or Interactive video-on-demand (iVoD) dedicates a single channel to each user and allows the video to be started at any time with VCR-like controls (pause, rewind, fast-forward, etc.) [14]. While being very user-friendly, this type of service has high bandwidth requirements and high deployment cost.

There is currently on-going research into optical core networks (e.g. agile all-photonic networks (AAPN) [11]) that should be able to support applications, like iVoD, that require substantial bandwidth. We address the challenge of minimizing the deployment cost of a network offering iVoD. As a first step, it is valuable to review the previously proposed solutions for the delivery of multimedia objects. This chapter serves that purpose and is organized as follows. Section 2.1 presents architectures and topologies that have been considered for content distribution (delivery) networks (CDNs) and VoD deployments. Section 2.2 surveys the different caching schemes and the strategies to determine file popularity. Section 2.3 covers the techniques used for content delivery, traffic modeling and routing users' requests. Section 2.4 describes optimization problems related to the deployment of content delivery networks such as replica location, content allocation, storage capacity allocation and resource allocation. We summarize the factors to consider, cost functions and heuristics proposed to solve these problems.

2.1 Architecture and Topology



(a) The transport network is divided into a core network and local metro networks. (Reproduced from [15])



(b) Generic VoD network architecture. (Reproduced from [16])

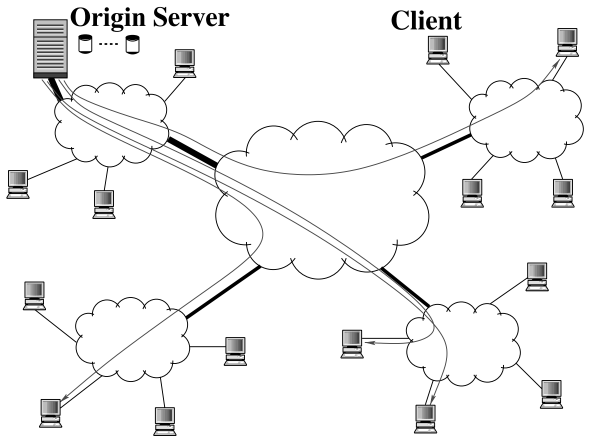
Fig. 2.1 (a) In this architecture, video servers are installed in the metro network to reduce the load on the core network. (b) Video library (VL) and video servers (VS) deliver the videos to the user's set-top box (STB). The front-end server (FES) is responsible for making and maintaining the connection between the user's set-top box and the video server. The front-end servers can also have a smaller video buffer (VB) to serve a fraction of the requests.

In this section, we present architectures and topologies that have been considered for VoD networks. In general, the available network infrastructure is divided into a core network and local/metro networks such as that depicted in Fig. 2.1(a) [15–22]. The core network is typically where substantial bandwidth is both available and needed due to aggregated transit traffic between the origin and replicas or clients. The local/metro network is responsible for the delivery to the users and is usually organized in a tree-hierarchy [16, 18, 21–23], but Wauters et al. have proposed to interconnect nodes as a ring [15].

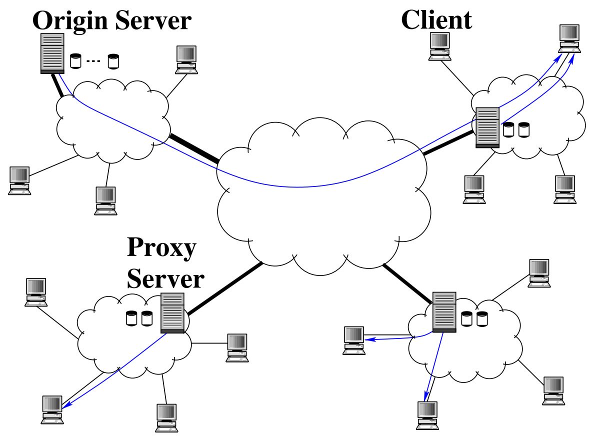
The deployment of a VoD network consists of placing and connecting a few elements shown in Fig. 2.1(b). The set-top box (STB) installed at the users' household is used to uncompress and display streams on a standard television. The video library (VL) and servers (VS) are responsible for storing and streaming the video objects. The front-end servers (FES) are responsible for the establishment and management of the connections between the set-top box and the video server. With the presence of a video buffer, a front-end server effectively becomes a small video server. Referring to Fig. 2.1(a), the main nodes represent the video servers and the local nodes front-end servers.

The location and presence of these elements in the network varies between each design. In a centralized architecture, the origin server is responsible for serving all the clients (Fig. 2.2(a)). Although it is very simple, this approach has serious weaknesses: a single point of failure and high load on both the origin server and the backbone network. Due to these shortcomings, authors have focused on distributed approaches with proxy servers installed at strategic location in the network (closer to the clients). The proxy servers cache content to reduce the load on the origin, as shown in Fig. 2.2(b) [15, 16, 20, 22, 23].

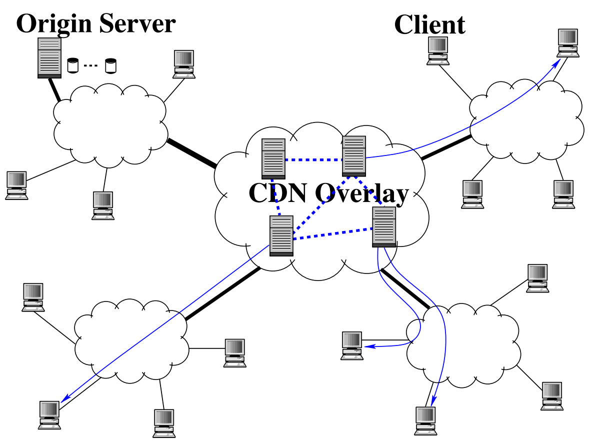
Instead of deploying proxy servers in the client domain, content distribution networks (CDNs) use proxy servers (called replicas or surrogates) on the edge of the core of the network, as close to the user-end as possible (Fig. 2.2(c)). The purpose of a CDN is to transmit to users the content they requested in the most efficient manner, that is, meeting the quality of service (QoS) requirements at the lowest cost possible. Content distribution networks, such as Akamai, achieve this by re-routing clients' requests to their replica servers [24]. Placing copies of objects at edge proxy servers closer to the user minimizes the delay at the user-end while reducing the bandwidth requirements at the origin server by serving a fraction of the requests at the proxies [25]. Furthermore, Barnett shows that distributed approaches can solve the main problems associated with centralized design without increasing cost [16]. However, Hefeeda et al. argue that proxy-based approaches shift the bottleneck from the origin to the proxy servers without reducing the cost and that CDNs are not cost-effective solutions for streaming media [17]. As an alternative, they propose a hybrid architecture based on the peer-to-peer (P2P) paradigm to distribute the files to the users (Fig. 2.2(d)). Ditze et al. have also considered collaborative transfers between peers to improve the scalability of media delivery networks [21]. In P2P-based architectures, network coding eases the scheduling and makes distribution more efficient [26, 27]. Finally, other solutions include placing proxy servers with different functionality both inside and outside the core [18] and assigning one server for each movie [19].



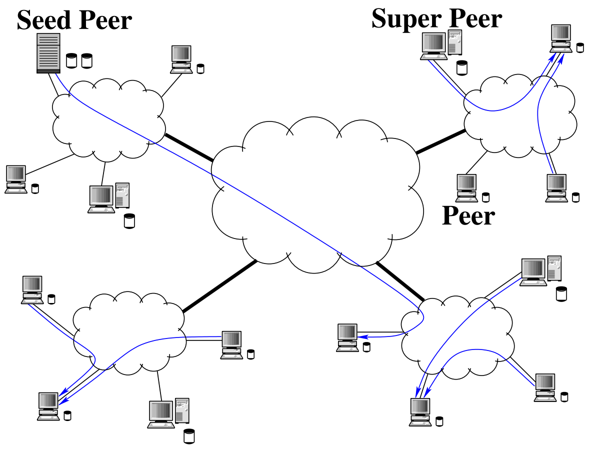
(a) Centralized architecture



(b) Proxy-based architecture



(c) CDN approach



(d) Hybrid architecture

Fig. 2.2 Architectures used for media delivery. (a) In the centralized architecture, all the requests from the clients are handled at the origin server. (b) Proxy servers located close to the user-end reduce the load on the origin server by caching content to serve a fraction of the clients' requests. (c) Content delivery network (CDN) is a third-party solution that deploys proxy servers in the core of the network (close to the edge) that serve a fraction of the clients' requests. (d) In a peer-to-peer (P2P) based approach, the peers share their resources to distribute the media. Powerful peers help in routing the requests and searching for content. (Reproduced from [17])

2.2 Content Allocation

Deciding upon the location of the proxy servers is not the only task, because the determination of the optimal content to store at each of these locations is non-trivial. The choice of content has an impact on the total cost (amount of storage required) and on the user perceived quality. If the selection is poorly made, users are forced to retrieve the data from the origin server which increases latency and the load on the origin server.

A content allocation strategy is server replication, which consists of placing copies of the origin server at strategic places in the network. Server replication partitions the network, resulting in lower bandwidth requirements at the expense of server cost. When using such a strategy, the placement of the servers that minimizes total cost is above the switches that connect the users to the network; between 70% and 90% of the path length between the origin server and the user-end [28]. However, according to Lu, content distribution networks using edge delivery (files are transmitted to users via servers placed on the edge of the Internet) cannot be scaled to deliver high-quality broadband video because there are no sufficient and affordable bandwidth and QoS on the last mile. [29]. He proposes the inclusion of "leaf servers" in local-area networks (LANs); these serve as second-tier surrogate and support a relatively small number of clients. The motivation behind this approach is that heavy traffic does not go beyond the edge servers of the core network and LANs have abundant and stable bandwidth, are less dependent on a sophisticated direction system and have a higher degree of personalization.

It is not always possible to have complete replicas of the origin server because the large size of multimedia objects leads to a high storage cost. An alternative is to store only specific objects from the origin at the surrogate servers; upstream bandwidth is reduced at the cost of increasing the storage for caching the most popular programs. Using the cost model he developed, Schaffa found that overall minimum cost is achieved when 15% of the programs are cached at 80% of the path length between origin and client [28]. The popularity of an object changes through time and a hot (popular) file might become cold (the number of requests falls below a given threshold) after some time. To maintain request coverage stable for long periods, it is important to replace objects that become cold with hot objects, a procedure called *incremental clustering* in [30].

Schaffa suggests that program caching be performed at more than one level in the network hierarchy [28]. The idea is to use a main cache to reduce overall system cost and a secondary cache at a higher level for fine-tuning the performance. When the main cache is close to the root, the cost of the system is mainly driven by the bandwidth component which makes the secondary cache almost useless. As the main cache is placed closer to the user, storage starts being the dominant factor and splitting the cache becomes advantageous. If the client request rate is high and/or proxy storage is limited, storing file prefixes rather than full files significantly reduces delivery cost [31,32]. It prevents clients from experiencing delays and jitter and reduces traffic on the origin-proxy path. Despite these advantages, Almeida et al. argue that storage at proxies is only effective if the origin is not multicast-enabled, the file request is low or the cost of a proxy is a small fraction of the origin server [33].

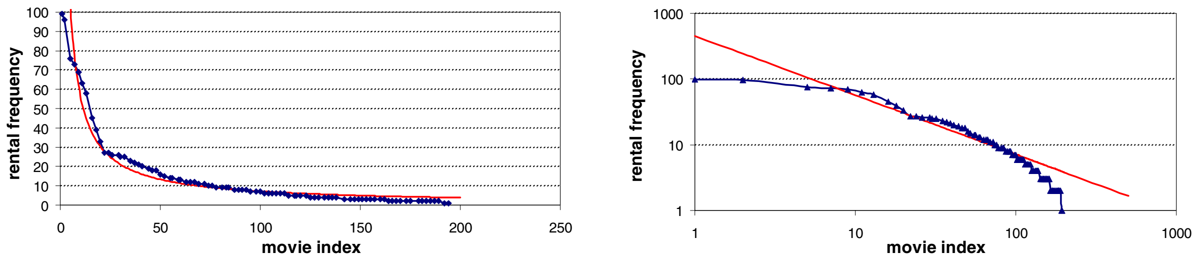
An efficient way to improve the performance is by sharing the content of the different surrogate servers by grouping them into clusters [20]. Clustering avoids the duplication of content at servers that are close to each other. In a hierarchical content routing scheme [34], the request are served by the local server (local hit), by another server in the same cluster (intra-cluster), by a server outside the cluster (inter-cluster), or by the original content server. Another approach is to cluster data using correlation distance (spatial, temporal, session clustering or popularity-based) [30].

Finally, it is worth mentioning that there are two different approaches to allocate content [30]. First, in the client-initiated approach, or pull-caching, the replica retrieves the copy of an object in the case of a cache miss. On the other hand, in a server-initiated approach, or push-caching, content is distributed to replicas before any requests for this data have been made [35]. If we anticipate that a specific object will be very popular (e.g. blockbuster movie release), it is advantageous to distribute the object prior to any requests in order to avoid cache misses and longer delays.

2.2.1 Choice of Content and File Popularity

When allocating content with a program caching scheme, only the most popular files are stored, with the aim of minimizing the storage and bandwidth needs. By using an appropriate popularity distribution, we can predict the hit ratio at a replica site given the set of files it is hosting. The hit ratio represents the probability that a user's request is served on a given path (or a given replica) [2].

Previous studies exploring the distribution of multimedia files in CDNs have used Zipf's Law to characterize the popularity of the different files [15, 28, 31, 33, 34]. In Zipf-like distributions, access frequency for file of rank i is equal to C/i^{α}, where C is a normalization constant and $\alpha > 0$ is the distribution parameter [36]. Such distributions generate a linear curve in a log-log plot of access frequency versus rank. In [30], the analysis of Chen indicates that 80% of all requests are for 10% of all Web objects. Based on a video store rental statistics ([37]), researchers also adopt the Zipf approach [38–40] to model popularity in video-on-demand applications. Although this data seems to fit a Zipf curve (Fig. 2.3(a)) on a linear scale, Fig. 2.3(b) shows that the part of the curve for the most popular files is flattened and does not fit the Zipf linear curve on a log-log graph.



(a) linear scale (b) log-log scale

Fig. 2.3 (a) The popularity distribution from a 1992 video rental data set used to justify Zipf's law in many video-on-demand proposals, along with a Zipf curve fit with $\alpha = 0.9$, and (b) the same data set and curve fit plotted on a log-log scale. Contrary to the assumption of many papers, video rental data does not appear to follow Zipf's law. (Reproduced from [10])

Gummadi et al. explain this behaviour by analyzing the characteristics of video object access [10]. VoD system users rarely access the same file twice because the files are not modified (fetch-at-most-once). However, new files are often added to the system. In contrast, Web objects are accessed more than once because they are updated regularly (fetch-repeatedly). Since the popularity of a movie diminishes in time, when new titles are added to the system, they become the most popular titles. Hence, popularity distributions need to be adjusted over time. To model the flattened part of the curve, Almeida et al. used a mixture of two Zipf distributions, after noticing the log-log graph is divided in two linear curves [41]. Although the mixture model fits the data reasonably well, there is no explanation of why the mixture is a realistic model. Gummadi et al. propose a model that is driven by Zipf's Law but takes into account the "fetch-at-most-once" and "new arrivals" factors [10]. When a client makes a second request, the previously fetched files are removed from the distribution and access probabilities are recalculated to have a total probability of 1. When an object is added to the system, its popularity rank is determined from a Zipf distribution, the rank of existing files which are less popular is decreased and file probabilities are recalculated to normalize the distribution to 1.

Gridwodz et al. propose a model for request generation based on the long-term life cycle of movies in the VoD context and varying user population sizes. They consider not only day-to-day changes when estimating the popularity of a file, but also the variations of the users' behaviour throughout the day: children's interests dominate during the afternoon whereas adults' interests dominate later in the day [23].

2.3 Content Delivery

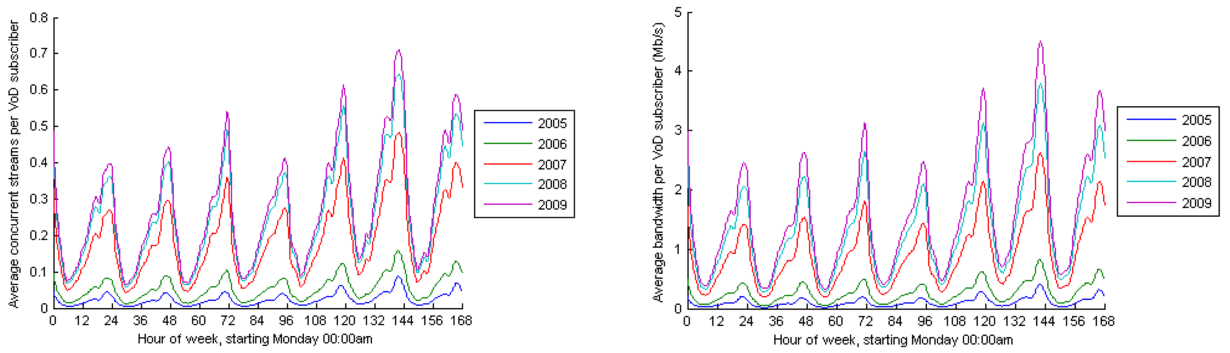
Content delivery consists of transmitting objects from the surrogate servers (or origin servers) to the clients. A popular technique to transmit large multimedia files over the Internet is called streaming. It allows clients to start displaying the data before the entire file has been transmitted which is useful if the user does not have fast access or the file to send is very large. VoD is unicast in nature (there is a dedicated stream to each user), which imposes significant bandwidth pressure on the network, but provides interactive VCR functions to the user. In broadcast schemes, the video is transmitted with a pre-defined schedule on a dedicated channel that supports any number of clients with a constant amount of bandwidth. As opposed to unicast connections, the client has no control on the stream (when it starts or stops) and bandwidth is wasted if the popularity of the video is low.

Another scheme is multicasting, which is a one-to-many connection where multiple clients receive the same stream from a server by monitoring (listening) to a specific multicast IP address [42]. Lichtenberg argues that multicasting can easily be implemented in existing client and server structures and provides a better Quality-of-Service (QoS) while saving a substantial amount of bandwidth [43]. An example of multicasting is batching, which collects requests that arrive within a given time interval and then multicasts the stream to the clients [44, 45]. In patching (stream tapping), if there is no stream for a video, then one is initiated when a client requests it [46–48]. If it already exists, then the client simultaneously listens to the multicast stream of the video and retrieves, from a proxy server, the part of the video that was streamed before he joined the broadcast. In [14], Lee proposes a trade-off between nVoD and iVoD called *unified video-on-demand* (uVoD). This approach first tries to serve a client by searching for a channel that is multicasting the requested movie, if none is available it assigns the first free unicast channel. Although this scheme shows performance close to that of iVoD, it requires additional buffering capabilities at the user-end.

The problem with aforementioned techniques is that they all require the path between the server and the client to be multicast-enabled (all the routers on the path must be able to interpret Class D IP addresses), but multicast capability is far from being fully deployed on the Internet due to its lack of support by Internet Service Providers (ISPs) [31, 49]. One solution when the end-to-end network provides only unicast service is to use proxy-assisted transmission schemes (one-to-one connection between the server and the client). By using patching in the unicast context (which is possible because proxies can forward one copy of the data to multiple clients), Wang et al. derived a transmission scheme that takes advantage of prefix caching at proxy servers [31]. The proxy transmits the prefix to the clients (if present locally) and schedules the transmission of the suffix from the origin server. If a request arrives within a given interval after the transmission of the suffix starts, the proxy can schedule a patch from the origin for the missing part of the suffix. Application-layer Multicast provides another alternative to IP multicast [49–51]. In this method, end hosts need to maintain a data forwarding path for nearby hosts. In [49], Milic et al. suggest an approach called *Multicast Middleware* that uses a virtual network device for capturing the traffic and forwarding it to a user application. Hsu et al. propose a mechanism called *Active Video Delivery* (AVD) that takes advantage of application-layer multicast [52]. Although AVD does not require all the routers on the transmission path to be multicast-enabled, it achieves the same efficiency as IP Multicast. Another way to transmit content without requiring multicast support on the delivery path is to use peer resources [17, 21, 26, 27].

2.3.1 Traffic Models

We are interested in modeling traffic generated by high quality video (determining the amount of bandwidth required for a stream) for applications like VoD. Without using any compression schemes, it would be difficult to transmit DVD-like quality videos over the Internet because of their large bandwidth requirements. For that reason, compression methods like MPEG, which can achieve high compression ratio while maintaining good quality, are used and MPEG-encoded sources are expected to generate a large part of the Internet traffic in the future.



(a) Concurrent streams per subscriber (b) Bandwidth per subscriber

Fig. 2.4 VoD usage projections by time of day and day of week. (Reproduced from [54])

MPEG videos are encoded using a variable bit rate (VBR) making traffic modeling a non-trivial task. The VBR is caused by the fact that compression is performed by encoding each frame using one of three different schemes: intra(I), predicted(P) and bidirectional(B). I-frames are encoded with a low compression ratio, but are independent and act as reference points. P-frames provide a higher compression by using motion-compensated prediction based on the previous I or P frame. Finally, B-frames achieve the highest level of compression by using both the previous and next frame in the sequence for its prediction. These different levels of compression produce frames with different sizes and hence a variable bit rate. MPEG movies use a group-of-picture (GOP) structure based on a (N,M) cyclic format; each sequence contains N frames (6, 8, 10, etc.) with the first one being an I-frame and every Mth one a P-frame [53]. A full-length movie is usually encoded with one GOP structure even though the MPEG standard allows the use of many different structures.

The variable bit rate (VBR) and high burstiness of these movies make it difficult to predict the required resources. By reserving resources based on average rates, long delay are experienced in case of bursts or when the source is transmitting at peak rates. Fig. 2.4 depicts projections of the usage (in terms of concurrent streams and bandwidth per subscriber) of VoD during the next five years. Although these expectations are not based on actual data, the presence of peak hours, during which bandwidth requirements are substantially greater than at other times, is highly likely. If the system is designed to support the peak rates, it will be under-utilized outside the high-usage periods [54]. On the other hand, if it is not, the customers will experience poor service during the busy hours. One way to eliminate the peaks and maintain a constant rate is to request content ahead of time. However, this procedure requires users to have a device that can store the content at home. A solution to this problem is using a stochastic process to model the dynamics of VBR video traffic. Models of this nature take advantage of the statistical properties of the source to achieve higher utilization of the bandwidth. However, Wrege et al. argue that using these models has several drawbacks, mainly arising from difficulty of implementation and complexity [55]. Therefore, as an alternative, they suggest using deterministic models, which provide an absolute upper bound (worst case) on the source's arrival traffic. Empirical evidence indicates that peak-rate allocation, which leads to low-utilization of the network resources for bursty traffic, is not required for deterministic models [56–59]. These models are parameterized to establish an upper-bound on the arrival rate from the source . As an example, the token-bucket approach uses two parameters: average rate and bucket depth. As this solution is not suitable for variable bit rate sources, Lee et al. present an improved version of the leaky-bucket scheme by updating the parameter pair after every group-of-picture [60]. They take advantage of the fact that I-frames and P-frames can tolerate one extra frame delay compared to B-frames to reduce the bandwidth requirements [61]. Their simulations show better accuracy and higher utilization than previous leaky-bucket models or peak rate models.

The deterministic models are called data-rate models (DRMs) because they only consider the rate at which data is arriving. While these models are good for predicting average packet-loss probability, they fail to identify such details as percentage of frames lost or incomplete [53]. Alternatively, there are frame-size models (FSMs) which generate the size of individual MPEG frames that can afterwards be used to deduce the data-rate. Sarkar et al. show, through model simulation, that even a small loss rate can decrease the video quality substantially because loss of an I-frame (or part of it) affects an entire GOP [53]. They propose two FSMs that generate frame sizes for full-length VBR videos preserving both GOP periodicity and size-based video-segment transitions, which previously proposed FSMs failed to do. These transitions are modeled with a Markov renewal process, an approach also adopted in [62, 63]. Zhang et al. add that it is important to consider the entire auto-correlation structure (many models deal with I, B and P frames sub-sequences separately) [64]. Finally, Janakiraman et al. propose a proactive multicast scheme and demonstrated that it was able to deliver VBR content over constant-rate channels with minimal performance loss or complexity overhead [65].

Another consideration with video on demand (VoD) when predicting the required bandwidth is that several videos can be transmitted simultaneously on the same link. In that case, effective bandwidth per video (measure of the amount of bandwidth that a given source will use over a given time period) is in fact much lower because the average frame-size of a VBR video is usually different in different segments; this is known as multiplexing gain. Zhou et al. have developed a Markov-modulated gamma (MMG)-based model to predict the value of this multiplexing gain [63].

2.3.2 Request Routing

Request routing is a function performed by a content distribution network which consists of directing the client requests to the best surrogate server. The objective of a request routing algorithm is to exclude the surrogate servers that provide low performance while avoiding overloading the others.

For a replicated server system, one of the simplest approaches is Round-Robin (RR) [66]. This algorithm selects which surrogate serves a specific request in a cyclic mode without considering the state of the network. It means that a RR scheme can assign a surrogate that is overloaded or out of service to handle a specific request. On the other hand, there are many schemes which use various metrics to make a better decision than the RR algorithm. For example, the Response Time (RT) algorithm selects the surrogate based on the response time the user previously experienced with a particular server [67]. Although this scheme distributes requests among the different surrogates more efficiently than the RR scheme and provides users with low delay, it does not necessarily prevent overloading. On the other hand, the Load scheme assigns a probability to each surrogate in inverse proportion to the client-replica path's current utilization [68]. So, the Load algorithm prevents overloading by reducing the chance of a request being served by a busy server. In [66], Masa proposes an algorithm that takes full advantage of the CDN architecture by considering latency, cluster request rate and link load and capacity. *Worst Surrogate Exclusion* (WSE) is based on three concepts: the exclusion of surrogates with latency higher than the estimated average system response time, the equalization of the average response time and the prevention of overloading the surrogate servers. Based on simulation results, Masa shows how WSE performs better than the other schemes which either consider only one metric (Load and RT) or do not consider the network at all (RR).

When program caching is preferred to server replication, the request routing algorithms are different than those just described. Because surrogate servers are hosting sets of different objects, requests cannot be simply routed according to some metric. A simple method is the query-based scheme [34], in which a proxy broadcasts a query to other nodes in its cluster if it does not have the requested content locally. If a node in its cluster responds positively, the request is routed to that server. The downside of this approach is that the queries and replies generate a significant amount of traffic. An alternative is a digest-based scheme where each proxy maintains a list of the information stored on all others [69]. Although there is no “query traffic”, these lists need to be kept up-to-date date which, again, can produce significant traffic. One way to reduce this “update-traffic” is to centralize the list of files hosted by each proxy on a directory server [70]. Even if this approach helps to reduce undesired traffic, it has the disadvantage of having a single point of failure. Jian et al. propose a solution called the semi-hashing based approach which has small routing overhead and high efficiency [34]. Their scheme is a modified version of the hashing method ([71,72]) which uses the content’s URL, the address of the proxies and a hashing function to redirect the request to a designated proxy. Their enhancement consists of reserving a portion of storage at each proxy for local popular content. They show that even if the amount of storage dedicated is very small (smaller than 20%), there is a significant improvement in performance (higher hit-ratio). The only constraint is that cooperating proxies must be close to one another because requests are often redirected.

2.4 Optimization Problems

In section 2.1, we presented distributed architectures in which replicas are placed in strategic locations. Placing replicas very close to the clients, in order to achieve very small delay, is not a viable solution because of the storage costs it incurs. On the other hand, placing the replicas too close to the origin requires far too much bandwidth to handle all the traffic. The *replica placement problem* consists of determining the location of replicas in the network such that the performance is maximized given an infrastructure or that the infrastructure cost is minimized for a given quality of experience (QoE) impairment, such as delay, packet loss, frame loss, or packet jitter. Content delivery networks are usually modeled as read-only (or read-mostly) workloads using classic network problems like the k-median problem or the facility location problem [73]. In the k-median problem, the objective is to select k

locations for replicas among m potential sites for a fixed k. The choice for this value of k is not obvious; if the value is too small, clients are forced to take a longer route (long response time and high load on the network) whereas if the value is too large, the hit ratio becomes smaller and hence cost of delivery is shared by fewer requests. A high number of replicas results in a considerable traffic load to distribute the objects to the replicas. Therefore, contrary to intuition, deploying as many replicas as possible is not always good. A solution, to avoid this tedious task of determining a value for k, is to find the subset of the m locations that minimizes the cost over all possible values of k, which is known as the facility location problem [2].

Determining the location of the replicas is only one of the problem involved in the design on content distribution networks. The placement of the objects (which objects to cache at each replica) and the allocation of streaming and storage capacity at each location are other problems that affect the final design. In [45], the *video placement problem* is defined as identifying the number of copies of each video and their location such that capacity usage is minimized and a specified quality-of-service (QoS) is guaranteed. Laoutaris et al. argue that the replica and video placement problems should not be solved independently of the resource allocation problem to avoid a suboptimal solution [74]. They define the *storage capacity allocation problem* as the distribution of an available storage capacity budget to the nodes of a hierarchical content distribution system, given known access costs and client demand patterns. In [15], Wauters et al. address the *resource allocation problem* of determining the equipment required for transport (number of ports at each server and number of multiplexers and switch ports at each node). In this thesis, we focus on a different resource allocation problem: determining equipment required to store and stream the content at each location (number of streaming and storage devices).

2.4.1 Parameters and constraints

To solve these problems, it is crucial to first determine a good cost function which is minimized whilst respecting appropriate constraints. An important factor to consider when determining the cost function is the internodal distance between clients, replicas and origin servers. Many metrics are used to represent distance such as network latency, number of hops, or link cost (also called bandwidth cost). Another way to express distance is transmission cost, i.e., the cost to transmit a bit on a specific path [31, 75]. Bartolini et al. propose a scheme where requests are served by the closest replica and use distance as a means to measure the users' perceived quality by summing the user-replica distance over all requests [5]. In the streaming case, finding the multicast tree that minimizes the bandwidth cost is a trade-off between minimizing distance and maximizing the number of clients sharing a path segment (streaming and multicast are discussed in section 2.3). Almeida et al. argue that closest server and shortest path routing do not necessarily lead to lowest cost [76]. Instead, to calculate the delivery cost, they use the total network bandwidth, which is expressed as the sum (possibly weighted) of the bandwidth required for each hop on the delivery path.

Another key parameter is the storage server cost, or replication cost, of keeping a copy of an object at a given location [5, 28, 75, 77]. In addition to storage cost, a fixed start-up cost or a server installation cost has been considered in [75, 77]. Bartolini et al. propose an algorithm where the location of the replica changes dynamically [5]. The start-up cost is expressed as the addition or removal of a replica site. The server must also be able to serve all the incoming requests for this specific file. The server cost therefore includes the cost of the required bandwidth, which is proportional to the popularity of the file it stores. A server that hosts very large files (high storage cost) which are not popular (like archives) has low bandwidth requirements.

In multimedia applications, due to the size of the objects, it is not always possible to have complete replicas of the origin server, because unacceptably large storage costs incur. Therefore, a selection of the objects is stored at proxy servers; the choice is based on popularity and hit ratio. The decision of whether to place a file at a replica is based on its size and its popularity: is the object popular enough (able to maintain a given hit ratio) to deserve the storage space it requires? As the popularity of an object can change through time, it might be necessary to replace objects or update them. In HTTP applications, objects are small and the transmission cost from the origin server to the replica is negligible. However, video objects are much larger and the distribution of a document is only compensated by a finite number of requests from the client. Therefore, the number of updates or replacements required is another factor to consider.

In defining the optimization problem, the constraints to impose on the possible solutions must be considered. Depending on the given infrastructure, it might be necessary to upper-bound the storage capacity of the servers [7, 75]. Nguyen et al. add constraints on the load capacity of the server (number of requests it can handle) and a quality of service (QoS) threshold (maximum delay) for each request [77]. Other researchers impose requirements of the availability of any object in the system, e.g., all requests must be handled and all objects must be available [75, 77].

2.4.2 Cost functions

We divide cost functions into categories according to whether they consider a single or multiple objects and whether they take storage into account [73]. In a single object cost function, only the aggregate user demand is considered; the specific objects requested are unimportant. In the case of streaming media applications, a common choice for the delivery cost model is one that considers the bandwidth required by the servers and network as the only factor [33, 76]. An alternative choice is a cost function based simply on distance and hit ratio [2] (Table 2.1). In a paper by Bartolini et al. the storage, or hosting cost, is part of the so-called maintenance cost, which also includes the cost of updating the copies at the different locations (Table 2.1) [5].

A more complicated case is one where there are many different objects in the system, each with different popularity (user demand). A proposed solution by Wang et al. is to minimize the transmission cost (similar to what is done in the examples above with delivery cost) by finding the position for each object that results in the largest savings in transmission cost (Table 2.1) [31]. However, it is often impossible to minimize the cost while maximizing the performance because these are two conflicting objectives. Buchholz maps the quality of the service into the cost domain by determining the amount the customers are willing to pay for maximal performance [7]. Finally, in the case where both multiple objects and storage are considered, the cost function is the sum of the start-up cost, storage cost and transmission cost, as shown in Table 2.1 [28, 75, 77].

d_j	demand from client j
$hitratio$	hit ratio of replica i
c_{ij}	cost (distance) from client j to replica i
c_i	cost (distance) from the origin to replica i
$A(x)$	user-perceived quality in network configuration x
$M(x)$	maintenance cost per unit of time of a network configuration x
$\tau(x, d)$	dwell time of a network configuration x
v_i	prefix size

Table 2.1 Categories of cost functions

Objects	Storage	Example	Other
Single	No	$\sum_{\forall j} d_j \cdot hitratio \cdot c_{ij} + d_j \cdot (1 - hitratio) \cdot (c_i + c_{ij})$ [2]	[15,33,76]
	Yes	$[A(x) + M(x)] \cdot \tau(x,d) + \sum_{j=1}^{\lvert V_R \rvert} \sum_{k=1}^{C} (d_j^{k+} C^+ + d_j^{k-} C^-)$ [5]	
Multiple	No	$\sum_{\forall i} saving(m_i) = \sum_{\forall i} C_i(0) - C_i(m_i u / b_i)$ [31]	[7]
	Yes	$\sum_{i=1}^{n} C_F \cdot y(i) + \sum_{i=1}^{n} \sum_{k=1}^{K} C_s \cdot [\sum_{j=1}^{n} x_k(i,j)/M]^+ + \sum_{i=1}^{n} \sum_{k}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{n} C_{ij} \cdot x_k(i,j)$ [75]	[28,77]

$C_i(v_i)$	transmission cost for video i if a prefix v_i is stored
u	smallest unit of cache allocation
m_i	size of video i
b_i	mean bandwidth of video i
$y(i)$	1 if a server is installed at location i
$x_k(i,j)$	transmission cost of program k from location i to j
C_F	installation cost of a server
C_S	storage cost
M	number of multiple accesses
C_{ij}	transmission cost per program from location i to j

2.4.3 Heuristics

The cost functions presented in section 2.4.2 are often complex and obtaining the optimal solution is impractical; solving the *replica placement problem* is considered NP-hard[1]. The heuristics (algorithms with no guarantee of finding a solution) presented in this section offer near-optimal performance as a trade-off to reducing the complexity. In simpler scenarios, it

[1]The complexity class of decision problems that are intrinsically harder than those that can be solved by a nondeterministic Turing machine in polynomial time. [78]

is sometimes possible to calculate the optimal solution and use it as a reference to evaluate the performance of heuristics.

A popular heuristic, considered by many authors [2,7,30,74,77], is greedy selection [79]. It first chooses the replica that minimizes the total cost and then selects a second replica among the remaining sites such that the total is minimized when combined with the first choice. Replica sites are added either until a predetermined number of sites is reached (k-median problem) or when adding more replicas increases the total cost (facility location problem) [2]. Genetic algorithms are another approach to solve the optimization problems described at the beginning of this section [45, 75].

Although these methods are known to perform very closely to the optimal solution (within a factor of 1.1-1.5), they require knowledge about the client locations in the network and internodal distances [80]. Among the alternatives to greedy algorithms are hot-spot [2] and max fan-out [2, 80]. In the hot-spot algorithm, the traffic generated near each site is used as the metric for selection and is expressed as the total number of requests from clients within a given range. At each step, the algorithm selects the hottest (maximum number of requests) site available. This is different from the greedy scheme as the latest choice does not depend on the combined cost with previous selections. The max fan-out algorithm behaves similarly with the difference that the metric used is the number of input/output terminals at each site. In both cases, sites are added until a local minimum is reached. As routers with high fan-out are usually busy, the solution is to build a cluster of replicas as close as possible to high fan-out routers [80]. By using the sum of distances between each client and its replica as performance metric, these strategies usually work very well (within 1.1-1.2 of greedy placement). However, performance decreases when the number of clients is small.

The system state might change through time and the quality of an originally near-optimal configuration can deteriorate substantially. In order to adapt to system variations, we can periodically execute any of the aforementioned static algorithms to reposition replicas such that cost is minimized. However, if the period between two executions is not chosen carefully, the replica placement determined by the last algorithm execution can become poorly matched to the current network state. The dynamic algorithm proposed by Bartolini et al. analyzes the current configuration and removes unnecessary replica(s) (if possible) if it can support an increase in user demand [5]. If the current configuration cannot, the algorithm adds one or more replica(s) while taking the cost of these changes into account. When considering as performance measures (i) the average number of replicas, (ii) user-replica average distance and (iii) the number of requests that cannot be served, the heuristic performs within 2-4% of the optimal strategy as computed by solving the Markov decision model.

2.5 Concluding remarks

In this chapter, we reviewed the many aspects to consider when planning a VoD network: the architecture, the content allocation and delivery and the location of the replicas. In the first section, we discussed three different distributed architectures that reduce the load on the origin server by placing replicas (proxy-based and content distribution networks) or using peer resources (peer-to-peer) to deliver media. In section 2.2, we addressed content allocation. The size of the media objects and the amount of storage it takes to store entire libraries (server replication) motivates program caching (cache only the most popular objects) and prefix caching. Gummadi et al. showed evidence that the Zipf distribution, which is a good model for the popularity of Web objects and has used by many authors in the context of VoD, is not appropriate for media objects because of the access pattern of users for movies (fetch-at-most-once and new arrivals) [10]. In section 2.3, we reviewed delivery protocols, traffic models for video and request routing mechanisms. Although iVoD is unicast in nature, multicast is more efficient, but it is much harder to implement because it is not guaranteed to be supported along all delivery paths. In section 2.4, we presented the parameters and constraints to consider when solving optimization problems related to the design of content delivery networks. In particular, the replica placement problem consists of finding the location of replicas that minimizes a function that includes storage and/or transport cost. Because minimizing the cost function is a NP-hard problem, heuristics that produce near-optimal results have been proposed. Greedy algorithms perform the best when client locations, internodal distances and demand are known, but algorithms such as hot-spot and max fan-out have been suggested when this information is unknown.

In [15], Wauters et al. have presented a design tool to determine the network equipment (number of ports at each server and number of multiplexers and switch ports at each node) needed for a VoD deployment. In this thesis, we address the resource allocation problem of finding the number of VoD servers required at each location. Our main objective is to determine not only the location of the replicas, but the details of the equipment required (number of storage and streaming devices) at each site of the VoD network. We consider an architecture in which replicas are organized in a star topology with the origin server in the middle. This simple architecture requires no complex request routing; requests are first handled at the replica and, only if necessary, forwarded to the origin. The delivery to the client is performed using unicast from the replica or the origin and each stream is dedicated a predetermined and constant amount of bandwidth. We estimate the total load at each location by calculating the worst-case demand (during peak hours) based on the population size and use the bandwidth available during off-peak hours for content distribution (content update) from the origin to the replicas. In the following chapter, we construct a cost function based on the hit ratio that we optimize by choosing the fraction of the library to cache at at each location. We assume that the VoD software installed at each replica is capable of determining the popularity of each movie and properly fill the cache with the most popular ones. However, to map the size of a cache to its hit ratio, we build a function based on data generated by simulator that implements the “fetch-at-most-once” and “new arrivals” factors introduced by Gummadi et al. in [10].

Chapter 3

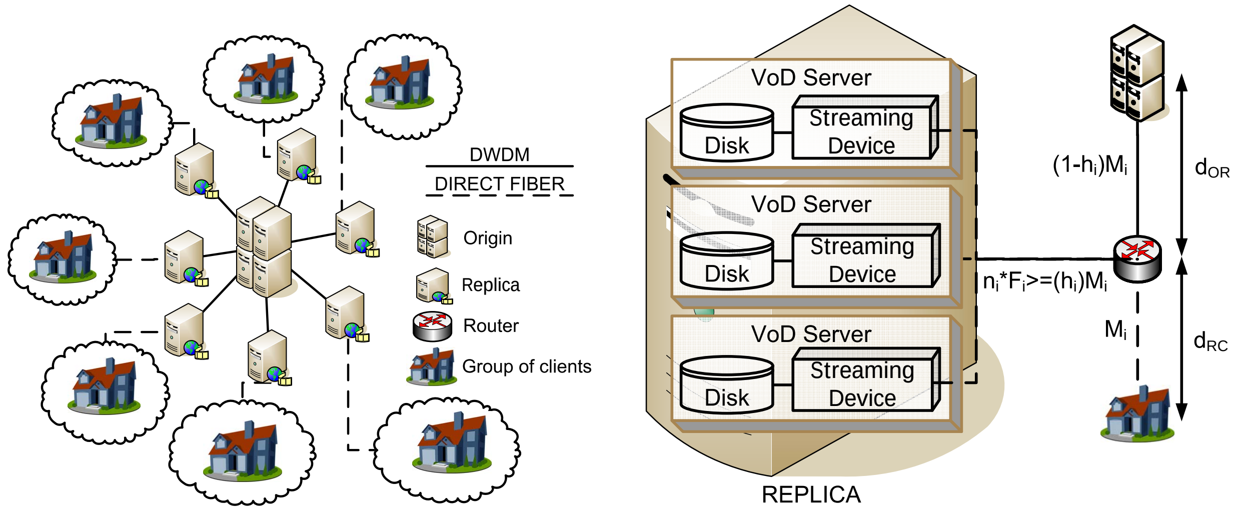
Video-on-demand equipment allocation

In this chapter, we focus on resource allocation in a VoD network deployed in a metropolitan area network such as the one depicted in Fig. 3.1(a). We define the *VoD equipment allocation problem* as choosing the number of streaming and storage devices (depicted as VoD servers in Fig. 3.1(b)) for each potential replica locations, such that the deployment cost of the VoD system is minimized. We solve this problem by determining the fraction of the total library that should optimally be stored at each location.

The remainder of this chapter is organized as follows. In Section 3.1, we express the equipment allocation problem as an optimization problem, and we state our assumptions. In Section 3.2, we present our solution to this problem, developing a novel cost function, hit ratio estimation function and heuristic. In Section 3.3, we present the VoD Equipment Allocation Tool: an interactive design tool that implements our solution. In Section 3.4, we apply our heuristic to three scenarios with different demand, equipment capabilities, and topologies (or geographies). We compare the cost generated by our heuristic to a centralized design and illustrate our method of determining when a centralized VoD deployment should be modified to a hierarchically-distributed VoD deployment.

3.1 Problem statement

We address the problem of determining the number of storage and streaming devices needed at each potential replica location. We require the specification of the topology of a metro-



(a) Logical connectivity (b) Replica level

Fig. 3.1 (a) The topology shows the logical connectivity between the clients, replicas and the origin. All requests originating from a group of clients are routed to the associated replica through direct fiber. If the request cannot be served by the replica (file not present), it is re-routed to the origin and served through DWDM equipment. (b) A replica with $n_i = 3$ VoD servers each with storage capacity of G_i TB and streaming capacity of F_i Gbps. The total streaming capacity $n_i \cdot F_i$ must be greater or equal to $h_i \cdot M_i$ where h_i is the hit ratio at site i given the storage capacity $n_i \cdot G_i$ and M_i is the worst-case demand from the attached group of clients.

area network (MAN) indicating the set of inter-nodal distances and the specifications (cost and capacity) of network elements and available equipment. We consider the case where only one type of equipment (VoD server) is installed at each site but allow this equipment type to vary from site to site. However, in Chapter 4, we relax this assumption and assume a case where we are given a set of available VoD server models and must determine the number and the model of servers to install at each location. We define the *VoD equipment allocation problem* as choosing the equipment for each of these replicas such that the deployment cost of the network is minimized.

As illustrated in Fig. 3.1(a), this topology contains one origin server and a maximum of N replicas. Each replica is responsible for a group of clients representing a fraction of the population; any request made by a client in that group is routed to that replica. The origin server hosting the entire library (the complete set of objects) can be located anywhere and serves all the requests that replicas are unable to fill.

This thesis does not consider the management of the content at the replicas. We suppose that there exists an external mechanism (e.g., VoD software installed at each replica) to maintain the most popular files at the replicas, which can be executed during off-peak hours when more bandwidth is available. Because content delivery itself is also out of the scope of this thesis, we are assuming unicast delivery to the user-end.

3.1.1 Mathematical formulation

Let $\mathcal{S} = \{s_i : i = 1, \ldots, N\}$ and $\mathcal{T} = \{t_i : i = 1, \ldots, N\}$ where s_i is the number of streaming devices with capacity F_i (Gbps) and t_i the number of storage devices with capacity G_i (TB) of replica site i. Let $C_{\mathrm{TOT}}(\mathcal{S}, \mathcal{T})$ be a strictly positive function that maps the number of devices installed at each location to the total network cost. The objective is to determine $\mathcal{S}$ and $\mathcal{T}$ to minimize total system cost:

$$\{\mathcal{S}^*, \mathcal{T}^*\} = \arg \min_{\mathcal{S},\mathcal{T}} C_{\mathrm{TOT}}(\mathcal{S}, \mathcal{T}) \tag{3.1}$$

This formulation is only valid when the streaming and storage devices can be deployed independently (s_i does not need to be equal to t_i). However, in practice, the two devices are often deployed as a joint unit called a VoD server, so that $s_i = t_i$ (Fig. 3.1(b)). To address this scenario, we define $\mathcal{N} = \{n_i : i = 1, \ldots, N\}$, where n_i is the number of VoD servers with streaming capacity F_i and storage capacity G_i at location i. The objective in this second formulation (used in the rest of this chapter) is to choose the number of VoD servers at each replica in order to minimize the total cost:

$$\mathcal{N}_{opt} = \arg \min_{\mathcal{N}} C_{\mathrm{TOT}}(\mathcal{N}) \tag{3.2}$$

We denote the worst-case demand M_i at replica i as the total bandwidth required to serve all client requests using unicast streaming during the peak utilization hours. We assume that we either know M_i or can approximate it from a given population size and peak usage ratio[1]. We define the hit ratio h_i as the smallest fraction of requests satisfied by replica i at any given time (worst-case). If the desired object is not present at the replica or the replica does not have enough streaming capacity, the request is unsatisfied (cache

[1]Worst-case demand M_i = population size $\times$ ratio of subscribers (clients/house) $\times$ peak usage rate (stream/client) $\times$ bitrate (Mbps/stream)

miss) and routed to the origin server. Although the hit ratio could be used as a measure of service quality, we do not to add this constraint to the optimization problem because of the imperceptible difference in quality of streaming video between the origin and a replica in a MAN.

3.2 Proposed solution

In this section, we present the three components of our solution: the hit ratio function, the cost function and the heuristic (Fig. 3.2). We call our heuristic the *Integer Relaxation Heuristic* (IRH). The first step of the heuristic produces an initial solution $\mathcal{X} = \{X_i : i = 1, \ldots, N\}$ where X_i is the fraction of the library stored at every replica i to minimize our cost function. This value is then used to form an estimate of the hit ratio, which allows us to calculate the number of servers needed, $\mathcal{N}_{ini}$. The second step of the heuristic consists of searching the neighborhood of the non-integer solution $\mathcal{N}_{ini}$ to determine the integer number of servers $\mathcal{N}_{IRH}$. We generate the infrastructure and transport cost for the entire network by calculating the output of the cost function for $\mathcal{N}_{IRH}$.

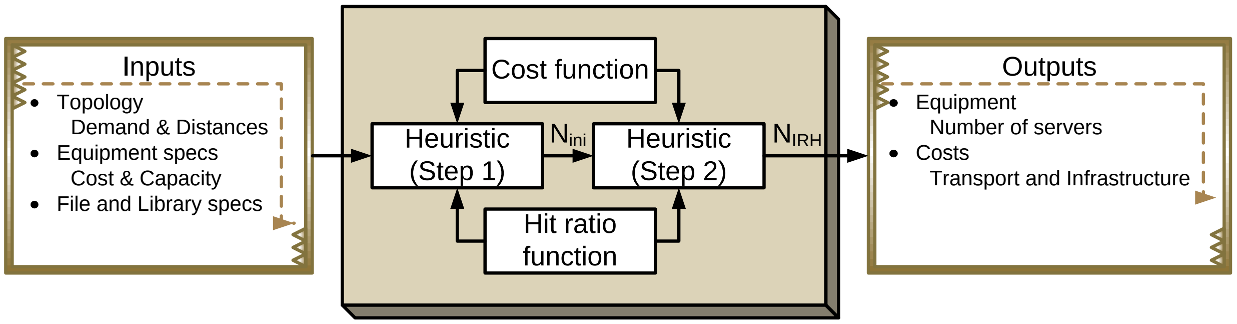


Fig. 3.2 High level overview of the proposed solution components: the heuristic, the cost function and the hit ratio function. The inputs consists of the worst-case distributed demand and internodal distances, the cost and capacity of the VoD servers, network interfaces and other network components and the number and type of objects (size and bandwidth requirements) stored at the origin and replicas. These values, the form of the cost function and hit ratio function are the inputs to the heuristic that produces an initial non-integer solution $\mathcal{N}_{ini}$. A near-optimal integer solution $\mathcal{N}_{IRH}$ is generated by searching the neighborhood of $\mathcal{N}_{ini}$ during the second step of the heuristic.

3.2.1 Hit ratio function

The purpose of a file popularity model is to predict the access frequency of a given file, which can be estimated by dividing the number of requests for this file by the total number of requests. In section 2.2.1, we reviewed popularity models and file access models for video-on-demand systems. Of particular interest are the “fetch-at-most-once” and “new arrivals” factors introduced by Gummadi et al. in [10]. Although it is possible to estimate the worst-case hit ratio through simulations, for the purpose of an interactive design process where we need to modify design choices repeatedly, it is impractical and time-consuming. Our objective is to train a parametric function that provides an estimate of the worst-case hit ratio based on specified system parameters in a few seconds compared to the tens of minutes required by simulations.

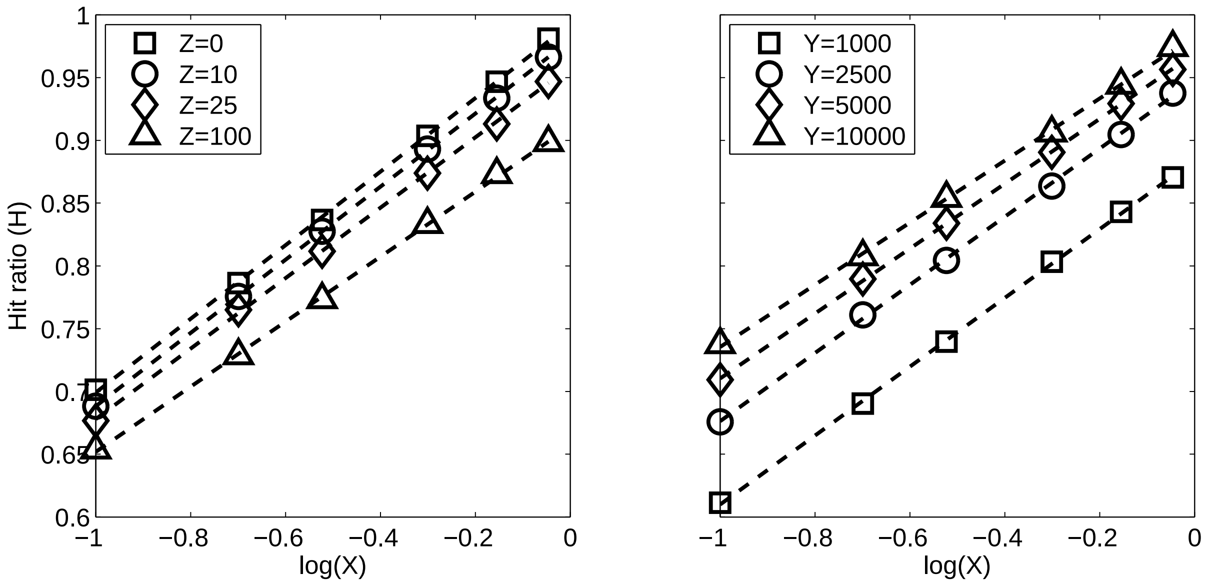


Fig. 3.3 Data fitting curves to construct the form of the hit ratio estimate $\widehat{H}$. The linear curves indicate that $\widehat{H} = A + B\log(X)$ achieves an adequate fit. Markers show values of H and the dashed lines (- -) show the linear fits. We plot the hit ratio H as a function of $\log(X)$ where X is the cache size ratio (Number of files in cache / Library Size). LEFT: We plot different values of file arrival rate Z for library size $Y = 2500$. RIGHT: We plot different values of library size Y for file arrival rate $Z = 50$.

We designed a simulation environment with a library of size Y and a cache of size $X \cdot Y$ where files are accessed according to the model described by Gummadi [10]. We calculate

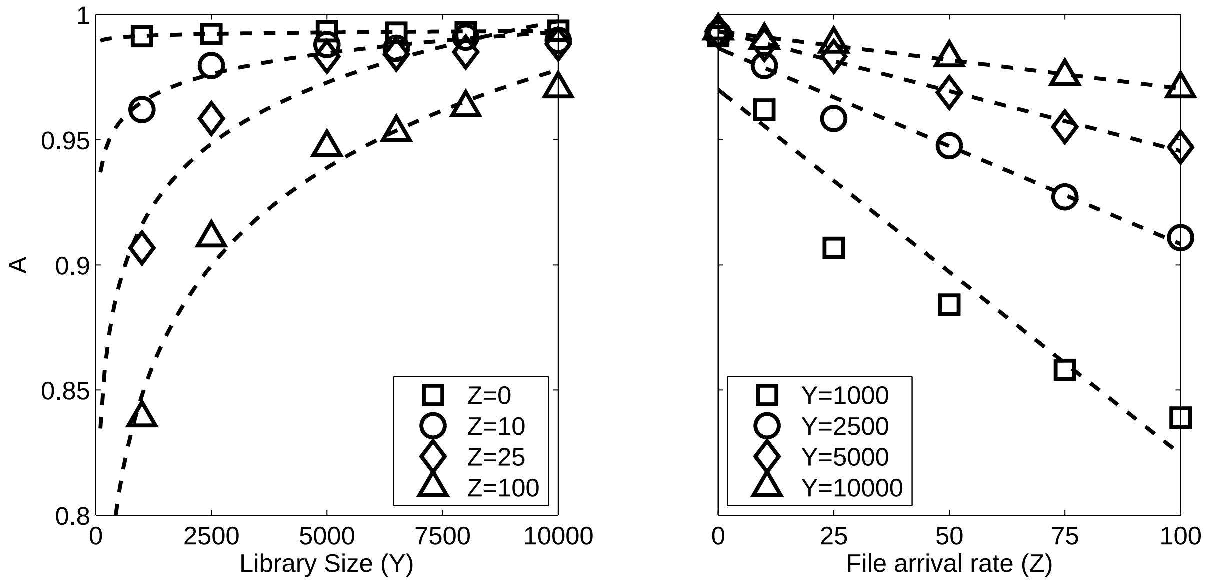


Fig. 3.4 Data fitting curves to construct the form of A in $\widehat{H} = A + B\log(X)$. Markers show values of A and the dashed lines (- -) show the our fit $A = K_1 + K_2 Z + K_3 \log(Y) + K_4 Z \log(Y)$ where Y is the library size and Z is the file arrival rate. LEFT: A as a function of the library size Y for different values of file arrival rate Z . RIGHT: A as a function of Z for different values of Y.

the hit ratio by dividing the number of requests for objects in the cache by the total number of requests. Let each client's library L_j be a subset of the complete library L that excludes all files client j has selected in the previous weeks. During each iteration (one week) of the discrete-time simulation, the following sequence of events occurs:

1. Clients are added to the population at a specified rate.
2. New files are added to the library L at a specified rate.
3. The cache is filled with the most popular files.
4. Each client j selects an object from his library L_j.
5. The weekly hit ratio is calculated.

The users' requests are generated using a Zipf distribution with coefficient $\alpha = 1$. The probability of selecting the file at rank i in library L_j is given by $p_j(i)$:

$$p_j(i) = \frac{i^{-\alpha}}{\sum_{i \in L_j} i^{-\alpha}}$$

Files that have already been fetched by the user cannot be selected again (fetch-at-

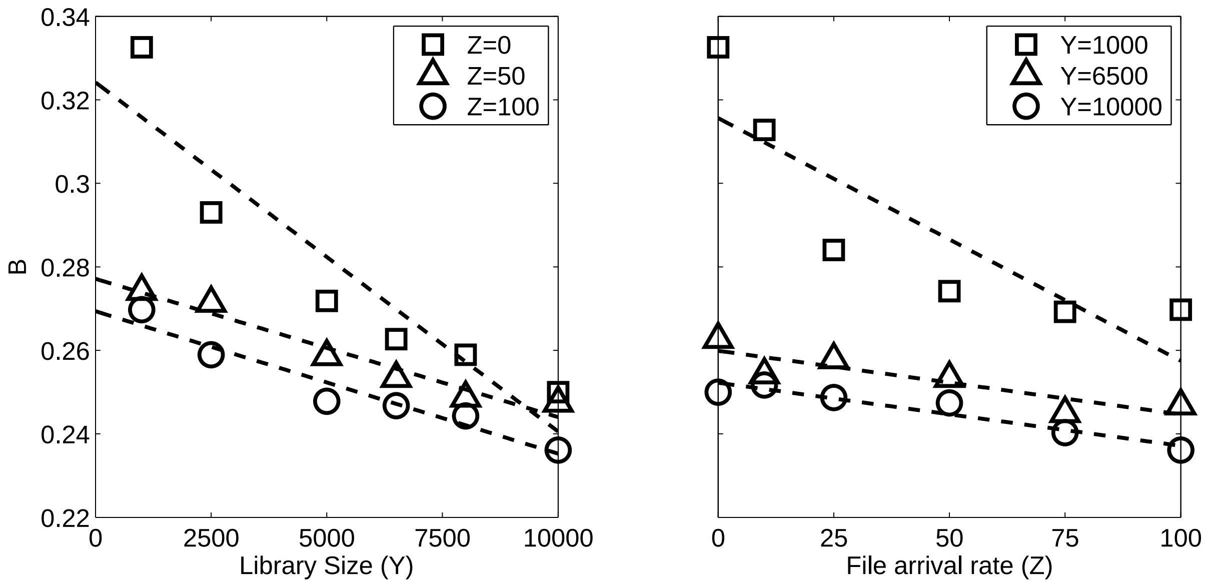


Fig. 3.5 Data fitting curves to construct the form of B in $\widehat{H} = A+B\log(X)$. Markers show values of B and the dashed lines (- -) show the our fit $B = K_5 + K_6Z + K_7Y + K_8ZY$ where Y is the library size and Z is the file arrival rate. LEFT: A as a function of the library size Y for different values of file arrival rate Z . RIGHT: A as a function of Z for different values of Y.

most-once-model). After every request a user makes, the selected file is removed from his library L_j and file selection probabilities are recalculated. New files are introduced in the library L and each library L_j at a specified rate. The insert position of a file is determined using a Zipf distribution (with $\alpha = 1$); the ranks of existing files which are less popular are decreased and selection probabilities are recalculated. We ran extensive simulations with different values for the following parameters:

1. Number of weeks (length of the simulation).
2. Size of the client population.
3. Number of new clients every week.
4. Size of the initial library of objects Y. [1000 2500 5000 6500 8000 10000]
5. Number of files added to the library every week Z. [0 10 25 50 75 100]
6. Size of the cache as a fraction (X) of the library size. [0.1 0.2 0.3 0.5 0.7 0.9]

From our simulation results, we determine that the only parameters that have a significant impact on the hit ratio are the library size Y, the number of files added every week Z and the cache size ratio X. We generated 864 points for the hit ratio H by running the simulation four times for 216 possible combinations of X, Y and Z. In Fig. 3.3, we observe the linear behavior of H as a function of $\log(X)$ for different values of Y and Z and propose the form in (3.3) for our estimate $\widehat{H}$, where $0 \leq X \leq 1$, $1000 \leq Y \leq 10000$ and $0 \leq Z \leq 100$. We construct the bilinear functional form for A and B represented respectively by (3.4) and (3.5). In Fig. 3.4 and Fig. 3.5, we show the fitting curves generated with those functional forms with the dashed lines (- -) and the actual values of A and B with the markers. The curves fit the markers for most of the sets depicted; the lines for the file arrival rate $Z = 0$ and library size $Y = 1000$ in both figures do not represent the actual value of A and B as accurately as the other curves. We consider that $Y = 1000$ represents a library size smaller than those of interests for large-scale deployment.

$$\begin{aligned}
\widehat{H} &= A + B \cdot \log(X) = f_3(X) && (3.3)\\
A &= K_1 + K_2 Z + K_3 \log(Y) + K_4 Z \log(Y) && (3.4)\\
B &= K_5 + K_6 Z + K_7 Y + K_8 Z Y && (3.5)
\end{aligned}$$

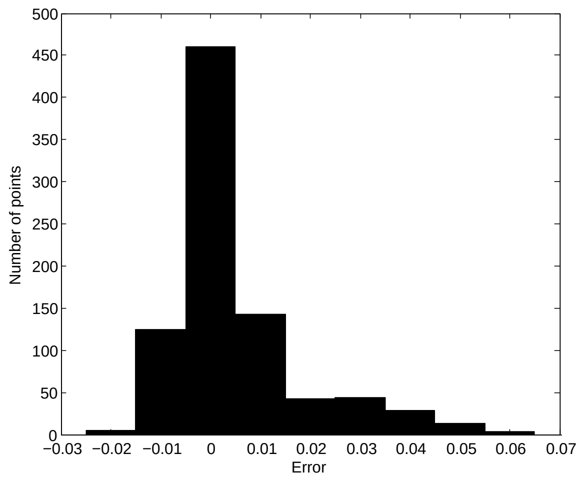
We determine the values of the coefficients K_1 to K_8 by solving in the least squares sense the system $KV(X, Y, Z) \cong H$ obtained by substituting (3.4) and (3.5) into (3.3). Our resulting function for h_i is accurate, showing less than a 0.02 error eighty-five percent of the time and less than a 0.05 error ninety-nine percent of time. In Fig. 3.6(a), we show the histogram of the error distribution for the entire dataset ($1000 \leq Y \leq 10000$) for our simulations, the error is less than 0.05 ninety-nine percent of the time. In Fig. 3.6(b), we show the histogram of a reduced dataset that focuses on the error for library sizes larger than 2500 files. The accuracy of the function estimate for this set is much higher: the error is less than 0.015 ninety-eight percent of the time.

3.2.2 Cost function

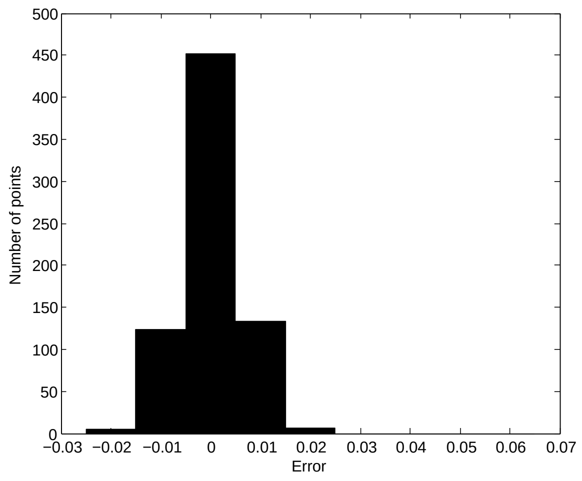
We can express the total cost, C_{TOT}, as the sum of the cost of infrastructure, C_T, and the cost of transport, C_S.

$$C_{\text{TOT}} = C_T + C_S \tag{3.6}$$

The cost of infrastructure, C_T, includes the software and start-up cost of a location (A_i) and the cost of VoD servers (B_i) for every replica site i and the origin server. In (3.7), we



(a) Entire dataset ($1000 \leq Y \leq 10000$)



(b) Reduced dataset ($2500 \leq Y \leq 10000$)

Fig. 3.6 Histograms of the error $\widehat{H} - H$ between our function estimate $\widehat{H}$ and the value observed during simulations H. (a) The error for entire dataset generated by our simulations. The error is less than 0.05 ninety-nine percent of the time. (b) The error of a reduced dataset where the values for library size below 2500 are discarded. In the case of $2500 \leq Y \leq 10000$, the error is less than 0.015 ninety-eight percent of the time.

express C_T as a function of the number of VoD servers installed at location i, n_i, and the origin, n_o.

$$C_T = \sum_{i=0}^{N} A_i + B_i n_i = f_1(n_o) + \sum_{i=1}^{N} f_1(n_i) \tag{3.7}$$

The cost of transport consists of two components: transport from the origin to replicas and clients, $C_{S_{OR_i}}$, and transport from replica i to client i, $C_{S_{RC_i}}$. It includes the cost of node interfaces (C_{IF}) and of fiber (C_f). The transport from replicas to the user-end (small distances) uses direct fiber whereas the transport from the origin to the replicas uses DWDM connections.

$$\begin{aligned}
C_S &= \sum_{i=1}^{N} C_{S_{OR_i}} + C_{S_{RC_i}} && (3.8)\\
C_{S_{RC_i}} &= n_{RC_i} \cdot (2 \cdot C_{IF} + d_{RC_i} \cdot C_f) && (3.9)\\
C_{S_{OR_i}} &= n_{OR_i}(2 \cdot C_{IF}) + \frac{n_{OR_i}}{w_{max}} \left[2C_{DWDM} + d_{OR_i} \cdot C_f + \left(\frac{d_{OR_i}}{d_{amp}} \right) \cdot C_{LA} \right] && (3.10)
\end{aligned}$$

n_{OR_i}:	Num. of interfaces (fibers) toward the origin.
n_{RC_i}:	Num. of interfaces (fibers) toward the user-end.
c:	Fiber capacity. (Gbps)
C_{IF}:	Node switch interface cost. ($)
C_f:	Cost of fiber. ($/km)
C_{DWDM}:	Cost of DWDM equipment ($)
w_{max}:	Number of fibers supported by DWDM equipment.
C_{LA}:	Cost of line amplifier. ($)
d_{amp}:	Max. distance between two amplifiers. (km)

The number of fibers at each node depends on the amount of traffic on the various links, the hit ratio at the replica and the fiber capacity. On the link between location i and the clients (RC_i), the traffic is equal to the demand from the user, M_i. On the link between the origin and a location i (OR_i), all the requests that cannot be served by the replica (cache misses) are handled by the origin server, generating a traffic equal to $(1 - h_i) \cdot M_i$. Notice that we are using non-integer values for the number of network equipment (number of fibers and ports) because we assume that the unused fraction can be used for other applications and does not need to be included in the cost.

$$n_{OR_i} = \frac{(1 - h_i) \cdot M_i}{c} \qquad n_{RC_i} = \frac{M_i}{c} \tag{3.11}$$

The worst-case demand between location i and the group of clients is fixed, so $C_{S_{RC_i}}$ does not depend on any of the optimization variables.However, $C_{S_{OR_i}}$ indirectly depends on n_i because the hit ratio h_i changes with the number of VoD servers installed. We express the cost of transport C_S as follows, where $f_2(h_i, M_i)$ is obtained by substituting (3.11) into (3.10) and (3.9):

$$\begin{aligned} C_S &= \sum_{i=1}^{N} C_{S_{OR_i}} + C_{S_{RC_i}} = \sum_{i=1}^{N} f(n_{OR_i}) + f(n_{RC_i}) \\ &= \sum_{i=1}^{N} f_2(h_i, M_i) \end{aligned} \tag{3.12}$$

By substituting (3.7) and (3.12) in (3.6), we can express the total cost as a function of the number of VoD servers installed (n_i), the hit ratio (h_i) and the demand (M_i) at each location:

$$C_{\text{TOT}} = f_1(n_o) + \sum_{i=1}^{N} f_1(n_i) + f_2(h_i, M_i) \tag{3.13}$$

The required number of VoD servers is determined by either the streaming or storage requirement ($n_i = \max(s_i, t_i)$), expressed as functions of X_i:

$$s_i = \frac{h_i \cdot M_i}{F_i} = \frac{f_3(X_i) \cdot M_i}{F_i} \qquad t_i = \frac{X_i \cdot Y}{G_i} \tag{3.14}$$

$$s_o = \frac{\sum_i (1 - f_3(X_i)) \cdot M_i}{F_o} \qquad t_o = \frac{Y}{G_o} \tag{3.15}$$

Define $f_4(\mathcal{X}) \triangleq \max(s_o, t_o)$ and $f_5(X_i) \triangleq \max(s_i, t_i)$. By substituting (3.3), (3.14) and (3.15) into (3.13) and assuming that the demand M_i is known, we express C_{TOT} as a function of the optimizing variable X_i:

$$C_{\text{TOT}} = f_1(f_4(\mathcal{X}) + \sum_{i=1}^{N} f_1(f_5(X_i)) + f_2(f_3(X_i)) \tag{3.16}$$

3.2.3 Integer Relaxation Heuristic (IRH)

The Integer Relaxation Heuristic (described in Algorithm 3.1) consists of two steps: (i) relaxing the integer constraint and (ii) searching the surroundings of the initial solution for a near-optimal integer solution.

Step 1: The first step of the heuristic is to provide an initial solution, $\mathcal{X}_{ini} = \{X_i : i = 1, \ldots, N\}$, representing the optimal fraction of the library to store at each replica. We obtain $\mathcal{X}_{ini}$ by performing a constrained nonlinear optimization on C_{TOT} (as expressed in (3.16)) where $0 \leq X_i \leq 1$, which is solved using a sequential quadratic programming (SQP) method [81,82]. From $\mathcal{X}_{ini}$, we calculate $\mathcal{N}_{ini}$, the set of fractional numbers of VoD servers, by expressing s_i, t_i, s_o and t_o as functions of X_i with (3.14) and (3.15).

```
1  Obtain $\mathcal{X}_{ini}$ by performing a constrained nonlinear optimization on $C_{\mathrm{TOT}}$;
2  Calculate the integer values of $\mathcal{N}_{IRH}$ with (3.14) and (3.15);
3  Calculate $\mathcal{X}_{IRH}$ from $\mathcal{N}_{IRH}$ using (3.17) and (3.18);
4  Set $C_0 = C_{IRH} = C_{\mathrm{TOT}}(\mathcal{X}_{IRH})$ and $k = 1$;
5  repeat
6      forall locations $i$ do
7          $\mathcal{X} = \mathcal{X}_{IRH}$ and $\mathcal{N} = \mathcal{N}_{IRH}$;
8          for $n_i \pm 2$ do
9              calculate $X_i$ with (3.17) and (3.18);
10             calculate cost $C_{\mathrm{TOT}}(\mathcal{X})$;
11             if $C_{TOT}(\mathcal{X}) < C_{IRH}$ then  $C_{IRH} = C_{\mathrm{TOT}}(\mathcal{X})$, $\mathcal{N}_{IRH} = \mathcal{N}$, $\mathcal{X}_{IRH} = \mathcal{X}$
12         end
13     end
14     $C_k = C_{IRH}$;
15     $k++$;
16 until $C_k \geq C_{k-1}$ ;
```

Algorithm 3.1: Integer Relaxation Heuristic (IRH)

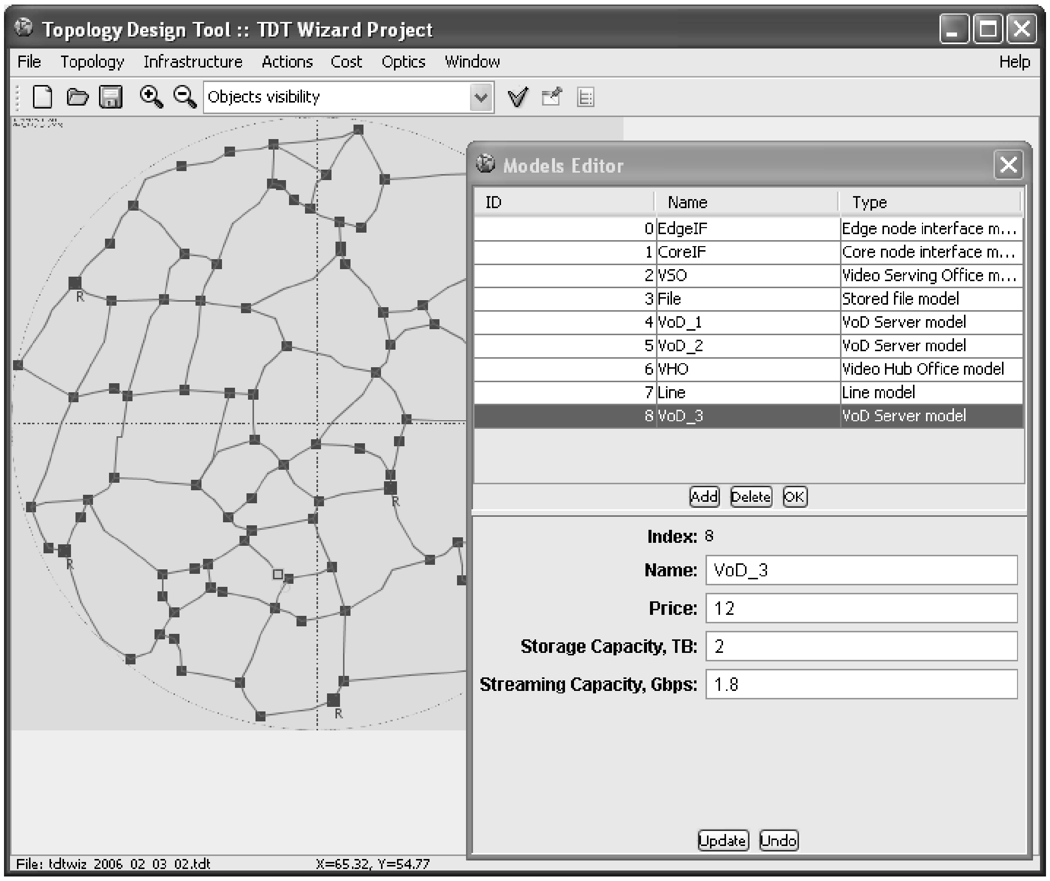
Step 2: The second step of the heuristic consists of searching the neighbourhood of $\mathcal{N}_{ini}$ for a near-optimal integer solution. One iteration consists of going through each location i and calculate the cost for integer values of n_i near the initial value ($n_i \pm 2$) by converting $\mathcal{N}$ to $\mathcal{X}$ using (3.17) and (3.18). At the end of each iteration k, we compare the lowest cost C_k with the lowest cost from the previous iteration C_{k-1}. We continue the search until $C_k \geq C_{k-1}$, which means that there was no improvement in the last iteration.

$$X_{storage} = \frac{n_i G}{Y} \qquad X_{streaming} = f_3^{-1}(h_i) = f_3^{-1}\left(\frac{n_i F}{M_i}\right) \tag{3.17}$$

$$X_i = \min(X_{storage}, X_{streaming}) \tag{3.18}$$

3.3 Interactive Design Tool

Our design tool, the VoD Equipment Allocation Tool, is an interactive Graphical User Interface (GUI) application used to plan the deployment phase of a video-on-demand (VoD) network. The tool includes two components: the Topology Design Tool (TDT) (developed by Vinokurov in [83]) and the optimization program. The TDT allows the user to (i) create topologies and models of network components and VoD infrastructures and (ii) visualize the design suggested by the optimization program (the solution we generate with our heuristic).



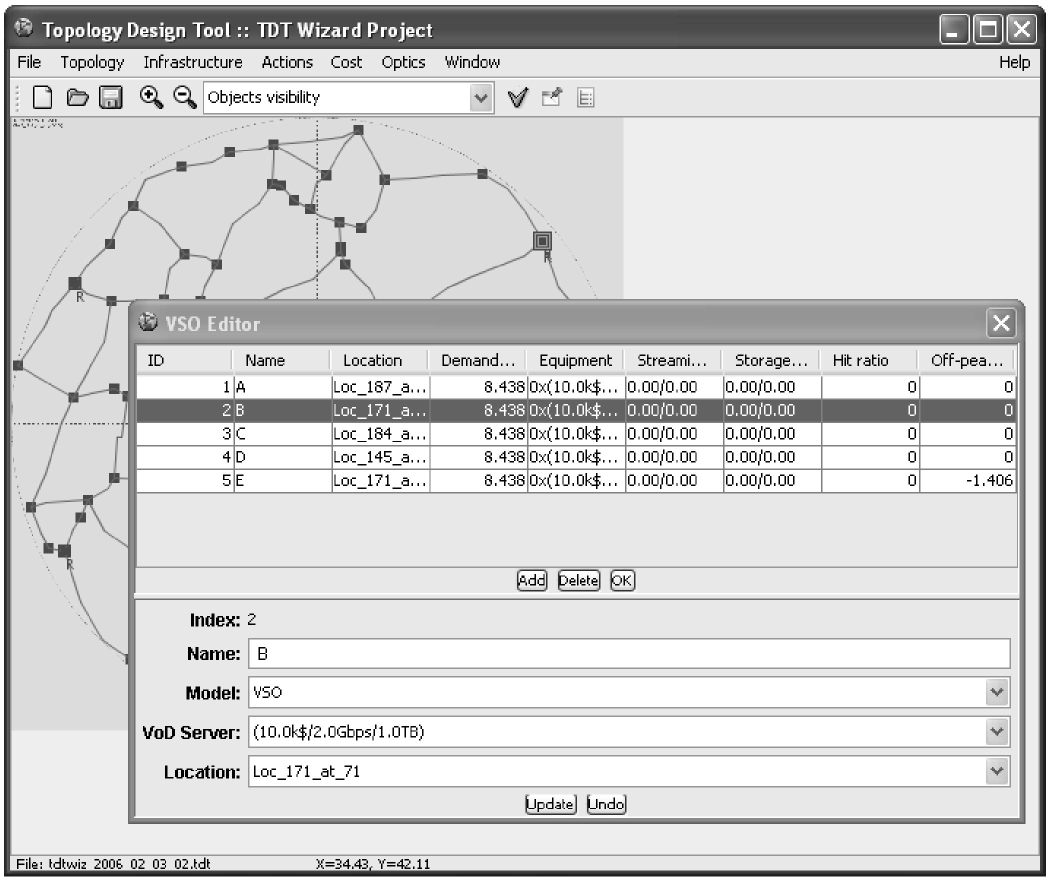
ID	Name	Type
0	EdgeIF	Edge node interface m...
1	CoreIF	Core node interface m...
2	VSO	Video Serving Office m...
3	File	Stored file model
4	VoD_1	VoD Server model
5	VoD_2	VoD Server model
6	VHO	Video Hub Office model
7	Line	Line model
8	VoD_3	VoD Server model

Index:	8
Name:	VoD_3
Price:	12
Storage Capacity, TB:	2
Streaming Capacity, Gbps:	1.8

File: tdtwiz_2006_02_03_02.tdt X=65.32, Y=54.77

Fig. 3.7 Model editor of the design tool. Topology of a MAN where the small squares represent inter-connected locations. The demand and infrastructures (replica) installed at each location can be customized by the user. The model editor allows the user to create different models for VoD Servers or other network components. The models are added to a library which is loaded every time a new project is created.

We describe the typical workflow to follow to design a VoD network with the tool. The


Topology Design Tool :: TDT Wizard Project
File Topology Infrastructure Actions Cost Optics Window Help
Objects visibility
VSO Editor
Add Delete OK
Update Undo

ID	Name	Location	Demand...	Equipment	Streami...	Storage...	Hit ratio	Off-pea...
1	A	Loc_187_a...	8.438	0x(10.0k$...	0.00/0.00	0.00/0.00	0	0
2	B	Loc_171_a...	8.438	0x(10.0k$...	0.00/0.00	0.00/0.00	0	0
3	C	Loc_184_a...	8.438	0x(10.0k$...	0.00/0.00	0.00/0.00	0	0
4	D	Loc_145_a...	8.438	0x(10.0k$...	0.00/0.00	0.00/0.00	0	0
5	E	Loc_171_a...	8.438	0x(10.0k$...	0.00/0.00	0.00/0.00	0	-1.406

Index: 2
Name: B
Model: VSO
VoD Server: (10.0k$/2.0Gbps/1.0TB)
Location: Loc_171_at_71

File: tdtwiz 2006 02 03 02.tdt X=34.43, Y=42.11

Fig. 3.8 Replica editor of the design tool. The object editing window allows the user to create and edit objects in the topology; for example, the user can edit a replica object by changing the type of VoD servers available. This windows also displays results from the optimization: s_i, t_i, n_i, h_i and bandwidth available during off-peak hours.

first step is to create the network topology with all the locations using the TDT Wizard or manually. The second step is to build models for network components, VoD equipment and the VoD network itself. At least one model (cost and specifications) needs to be craeted for each of the following components before adding infrastructures to the topology: network interface, DWDM switch, fiber, stored file, VoD server, library server (or origin) and replica server. In Fig. 3.7, we show the model editor that allows the creation and modification of all the components. When the topology and the models are created, the user can create replicas and origin objects using the Objects Editor (shown in Fig. 3.8). The editor allows the creation and modification of each replica and the origin server. A valid VoD network includes only one origin and any number of replicas (up to one per location). With a valid network setup, it is possible to run the optimization to determine the optimal equipment.

3.4 Results

In this section we examine the results obtained from applying our heuristic to three scenarios using our design tool.

Scenario1 20 different sets of inputs where the number of locations N in the topology is between 1 and 100, the number of files in the library between 1000 and 10000 and the file arrival rate per week is between 0 and 100. The system parameters are all uniformly distributed within the following specified ranges: demand M (1-20Gbps), startup cost A (6-37k\$), VoD server cost B (1-53k\$), streaming capacity F (1-5Gbps), storage capacity G (1-10TB), distance to the origin d_{OR} (0-50km), average distance to the client d_{RC} (0-5km), cost of bandwidth (0-4k\$/Gbps) and cost of storage (0-3k\$/TB);

Scenario2 Topology of 25 locations with the system parameters uniformly distributed within the same ranges as in Scenario1. For each trial, demand $M_i = M$ at each replica, where M varies from 2.5Gbps to 50Gbps;

Scenario3 Topology of 14 locations with the system parameters uniformly distributed within the same ranges as in Scenario1. The specifications of the equipment and demand at each node appear in Table 3.1.

Fig. 3.9(a) shows three different total network costs C_{TOT} for Scenario1: cost of a centralized design ($n_i = 0$ for all i) and cost after the first and second step of the Integer Relaxation Heuristic (IRH). Fig. 3.9(b) shows the percentage reduction of the cost achieved by the Integer Relaxation Heuristic. IRH yields average improvements of 17% over the a centralized design. The majority of the heuristic improvement comes from the first step.

In Scenario2 we illustrate the impact of the demand M on the deployment cost. In Fig. 3.10(a), as the demand increases, the cost differential between the design generated by our tool and a design in which no equipment is installed grows substantially. Below a certain demand ($\approx 7 - 8$ Gbps), both designs are of equal cost, which means that if the demand is too low, it is no longer cost-efficient to deploy equipment. Fig. 3.10(b) compares costs of transport and infrastructure for a single location i, which has a startup cost $A = 19k\$$ and

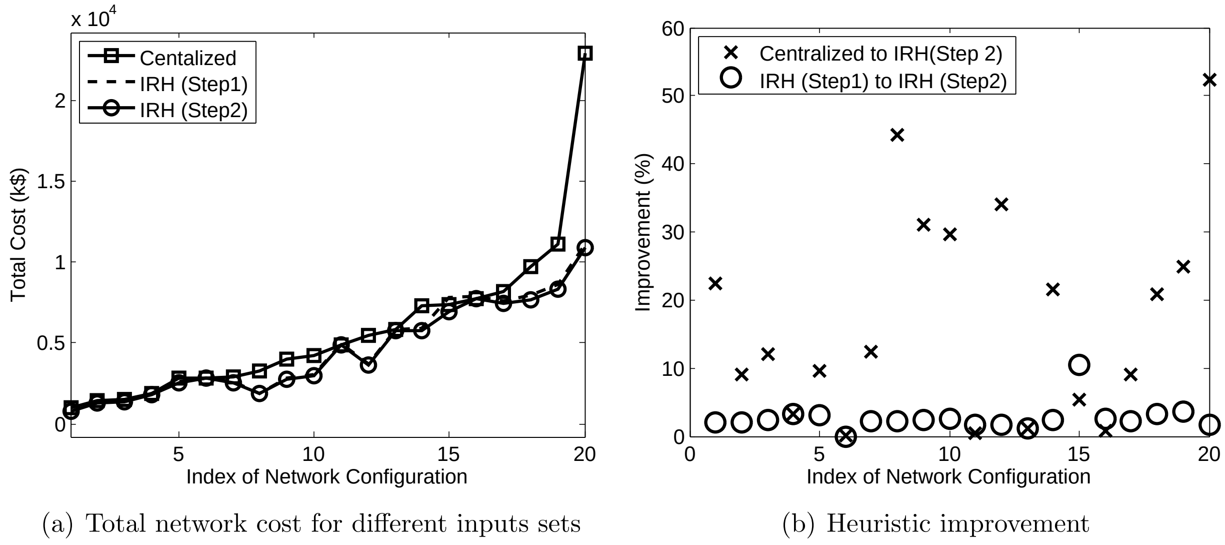
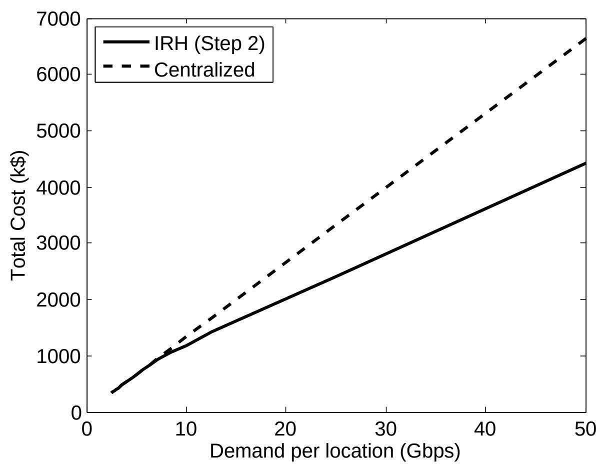


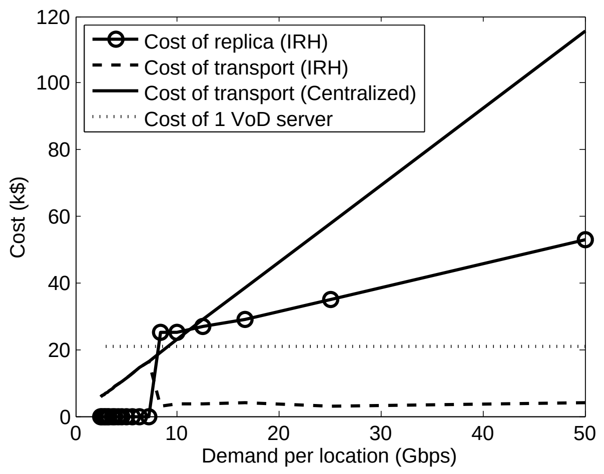
Fig. 3.9 Scenario 1. (a) Three values for the total network cost for each of the 20 different inputs sets are shown: placing no equipment (Centralized), after the first step of the heuristic (IRH (Step 1)) and after the Integer Relaxation Heuristic (IRH (Step 2)). (b) Cost improvement from a centralized approach and running the first step of the Integer Relaxation Heuristic (IRH (Step 1)) to running the entire IRH (IRH (Step 2)). Running IRH yields an average improvement of 2.5% on Step 1 and a 17% average improvement on a centralized design. In both (a) and (b), the 20 different cases are displayed in increasing order of cost of Centralized.

where VoD servers with $3Gbps$ and $2TB$ capacity are available at $2k$\$. Provided "Cost of 1 VoD server" is lower than "Cost of transport (Centrlized)", it is beneficial to cache content at i. If the equipment installed at the origin and i is identical and "Cost of transport (Centalized)" is smaller than "Cost of 1 VoD server", then it is cheaper not to install any replica and carry the entire demand up to the origin. In the analyzed scenario, equipment is cheaper at i than at the origin, so there is one point (M = 8.3Gbps) where the heuristic indicates that a replica should be installed even though the cost of transport is less than the minimum deployment cost. Therefore, the demand and type of equipment not only have an impact on the fraction of the library to cache, but also determine whether or not caching content is even profitable.

Table 3.1 displays the values for n_i, s_i and t_i calculated with our tool for Scenario3. The total network cost for this equipment is 1,810k\$. Looking at the table, we notice significant discrepancies between the values of s_{i_1} and t_{i_1}, which signifies that resources are wasted



(a) Entire Network



(b) One location

Fig. 3.10 Scenario 2. (a) Total network cost with demand $M_i = M$ at each replica, where M varies from 2.5Gbps to 50Gbps. As M increases, the gain from applying the Integer Relaxation Heuristic increases. The demand must be at least $\approx 7 - 8Gbps$ to justify the installation of equipment. (b) We consider location i with the following specifications: $A_1 = 19k\$$, $B_1 = 2k\$$, $F_1 = 3Gbps$, $G_1 = 2TB$. As a function of the demand M_i we show the following costs: transport cost from this location when $h_i = 0$ (Cost of transport (Centralized)), cost of equipment (Cost of replica (IRH)) and transport (Cost of transport (IRH)) when we apply our Integer Relaxation Heuristic (IRH) and minimum deployment cost (installing one VoD server: $A_i + B_i$).

because of poorly chosen equipment. For example, at location 3, the required number of streaming devices is almost twice the number of storage devices, whereas the streaming capacity of the equipment is half of the storage capacity. We consider the effect of making equipment available at these locations with specifications that better match storage and streaming needs. For example, we change the value of F_3 from 1 to 3 in order to have a closer match for s_3 and t_3. We repeat for the other three locations where $n_i \neq 0$ (7, 10 and 11) and adjust the value of B accordingly; for example, B_3 increases from 9k$ to 15k$ to support an extra 2Gbps. All the modifications (B_{i_2}, F_{i_2} and G_{i_2}) and new results (n_{i_2}, s_{i_2}, t_{i_2}) are also shown in Table 3.1. We notice that the four locations where we modified the hardware now have $s_{i_2} = t_{i_2}$, which indicates a better usage of resources. Moreover, because of the savings at these locations, it is now beneficial to install more equipment at locations 1 and 14 to achieve a minimal network cost. We also note that even though the

Table 3.1 Scenario 3. Initial specifications of 14 locations (left). On the right, specifications of the locations after modifying equipment (modified values are highlighted with the surrounding box).

Location	M_i	A_i	B_{i_1}	F_{i_1}	G_{i_1}	n_{i_1}	s_{i_1}	t_{i_1}	B_{i_2}	F_{i_2}	G_{i_2}	n_{i_2}	s_{i_2}	t_{i_2}
1	10	19	12	2	2	0	0	0	12	2	2	$\boxed{3}$	$\boxed{4.3}$	$\boxed{4.3}$
2	13	8	29	1	9	0	0	0	29	1	9	0	0	0
3	18	12	9	1	2	18	17.8	11	$\boxed{15}$	$\boxed{3}$	2	$\boxed{6}$	$\boxed{5.5}$	$\boxed{5.5}$
4	5	13	16	4	1	0	0	0	16	4	1	0	0	0
5	9	15	19	4	2	2	0	0	19	4	2	2	$\boxed{1.7}$	$\boxed{1.7}$
6	2	15	22	4	3	0	0	0	22	4	3	0	0	0
7	19	10	30	4	6	4	4.5	3.6	$\boxed{22}$	4	$\boxed{3}$	4	$\boxed{4.2}$	$\boxed{4.2}$
8	1	15	12	1	3	0	0	0	12	1	3	0	0	0
9	9	18	24	2	6	0	0	0	24	2	6	0	0	0
10	19	16	27	3	6	5	6.3	3.6	$\boxed{16}$	3	$\boxed{2}$	$\boxed{6}$	$\boxed{5.8}$	$\boxed{5.8}$
11	19	19	24	3	5	5	6.1	4.4	$\boxed{16}$	3	$\boxed{2}$	$\boxed{6}$	$\boxed{5.6}$	$\boxed{5.6}$
12	10	6	29	2	8	0	0	0	29	2	8	0	0	0
13	8	13	35	2	10	0	0	0	35	2	10	0	0	0
14	14	14	22	4	3	1	0	0	22	4	3	$\boxed{4}$	$\boxed{3.1}$	$\boxed{3.1}$

value of n_5 has not changed, the streaming and storage requirements have increased from 0 to 1.7. This means that the initial solution is $n_5 = 2$ instead of $n_5 = 0$ and that the value of n_5 is already optimal after the first step of the heuristic, it does not change from 0 to 2 during the searching step. The new total network cost for this setup is 1,580k$, a 12.5% improvement. It is important to stress that the prices and capacity used in these scenarios are not intended to reflect the real values used in practice. However, this simple example shows the impact of modifying the type of equipment installed at each location on the total deployment cost.

3.5 Conclusion

Network cost is affected not only by where replicas are located, but also what equipment comprises a replica. We developed a design tool (that implements a cost function, hit ratio function and heuristic) to address the *VoD equipment allocation problem.* There are four principal contributions in solving that problem. We used extensive simulations to train a parametric function that generates accurate estimates of the hit ratio for given cache size, library size and file arrival rate. We constructed a cost function based on the hit ratio h_i, the demand M_i, and the number of VoD servers n_i at each location. We designed a two-step heuristic, called the Integer Relaxation Heuristic (IRH), that relaxes the integer constraint to produce an initial solution and then identifies a near-optimal integer solution in a reduced search space. The tool that implements our cost function, hit ratio function and IRH is truly interactive because it allows designers to create and change network models to generate optimal designs in an efficient and timely manner.

Our key conclusions are: (i) the nature of the available server equipment has a major impact on the design and cost of a VoD network; and (ii) it is not always beneficial to cache content. It is profitable to install VoD servers (regardless of the library size) if the demand at the given location is significant. On the other hand, even if the library has tens of thousands of assets, if the demand is too low, no amount of caching can reduce the network cost. Accounting for the available equipment during the VoD network design is critical as the choice of equipment has a direct impact on the minimum demand that makes caching profitable. Moreover, selecting equipment that jointly matches streaming and storage requirements at each location can result in substantial reductions in network cost; we provide an example in Section 3.4 which illustrates that only a few equipment changes can have a major impact.

In this chapter, we adressed the problem of determining the number of VoD servers to install at each location when the streaming and storage capacity at each site was fixed prior to the optimization. In the next chapter, we relax this assumption and define a new problem where we are given a set of available VoD servers that can be installed at any location. This extension of the VoD equipment allocation problem consists of determining both the number and model of VoD servers to install at each location, such that the total cost is minimized.

Chapter 4

VoD Servers Model Selection

In Chapter 3, we presented the equipment allocation problem and our approach to solving it based on the assumption that a fixed, single and predetermined type of VoD server was available at each location. As a result of making this simplifying assumption, if multiple models are available, a network planner has to iteratively change the available server model at various locations until the network deployment cost cannot be further decreased. We showed how it is possible to identify which locations have suboptimal VoD server model by inspecting the discrepancy between the number of required and installed streaming and storage devices. A large difference is an indication of wasted resources, and hence of a bad choice of server model. The exercise of identifying suboptimal VoD server models is not trivial and can become very tedious for a large network.

In this chapter, we relax the assumption that we must pre-determine the available type of VoD server for each location. Instead, we assume that any server model from a given set can be chosen for each location. However, we still restrict ourselves to the case where a single type of server is installed at each location (one cannot mix several types of server). This is motivated by practical considerations such as the purchase (one vendor), the physical installation (rack of same servers) or the software management (same OS) of the servers. We formulate a new problem statement and propose two heuristics (IRH and IGS) for finding approximate solutions.

4.1 Problem Statement

We address the problem of determining not only the number, but also the model of the VoD servers at each potential replica location. The assumptions for this problem are the identical to those described in Chapter 3 (Section 3.1). To solve this problem, we also 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 $\mathcal{V} = \{v_i \in \mathcal{W} : i = 1, \ldots, N\}$ where n_i is the number and v_i is the model of the servers installed at location i. The new optimization problem is expressed as follows:

$$\{\mathcal{N}^*, \mathcal{V}^*\} = \arg \min_{\mathcal{N},\mathcal{V}} C_{\mathrm{TOT}_\mathcal{V}}(\mathcal{N}) \tag{4.1}$$

where $C_{\mathrm{TOT}_\mathcal{V}}(\mathcal{N})$ is the total cost of the network C_{TOT} for a fixed set $\mathcal{V}$.

4.2 Cost function

We adopt an approach to solving this new equipment allocation problem different from that presented in Chapter 3. Instead of expressing cost as a function of the fraction of the library cached at each location X_i, we optimize the number of VoD servers n_i directly. Recall that the total cost C_{TOT} is the sum of the cost of infrastructures, C_T, and the cost of transport, C_S:

$$\begin{aligned} C_S &= \sum_{i=1}^{N} f_2(h_i, M_i) \\ C_T &= f_1(n_o) + \sum_{i=1}^{N} f_1(n_i) \\ C_{\mathrm{TOT}} &= f_1(n_o) + \sum_{i=1}^{N} f_1(n_i) + \sum_{i=1}^{N} f_2(h_i, M_i) \end{aligned} \tag{4.2}$$

To derive an expression for C_{TOT} solely in terms of n_i for $i = 1, \ldots, N$, we resolve h_i and n_o as functions of n_i (M_i it is assumed to be a fixed parameter). We develop expressions to calculate h_i and n_o for a fixed $\mathcal{N}$. The hit ratio at a location is limited by either the streaming or the storage capacity. The demand at the replica $h_i \cdot M_i$ cannot exceed the streaming capacity $n_i \cdot F_i$. We calculate the cache size ratio X_i from the storage capacity $n_i \cdot G_i$. Using the value of X_i just calculated and (3.3), the equation for $\widehat{H}$ presented in the previous chapter, we determine the maximum hit ratio h_i achievable for a given storage capacity. We express h_i as $f_3(n_i)$:

$$h_i = \min \left[\frac{n_i \cdot F_i}{M_i}, \widehat{H} \left(\frac{n_i \cdot G_i}{Y \cdot \text{file size}}, Y, Z \right) \right] = f_3(n_i) \tag{4.3}$$

The number of servers required at the origin, n_o, is also constrained by either streaming or storage. The storage capacity $n_o \cdot G_o$ must be at least equal to the amount of storage needed for a library of Y objects. The origin must also have enough streaming capacity $n_o \cdot F_o$ to handle the cache misses from all the replicas equal to the sum of $(1 - h_i) \cdot M_i$ for all locations i. In (4.4), we define n_o as $f_4(n_i)$ by substituting h_i with the expression in (4.3).

$$n_o = \max \left[\frac{\sum_{i=1}^{N} (1 - h_i) \cdot M_i}{F_o}, \frac{Y \cdot \text{file size}}{G_o} \right] = f_4(\mathcal{N}) \tag{4.4}$$

By replacing the equations for n_o and h_i in the initial definition of C_{TOT}, we derive a new expression solely in terms of n_i:

$$C_{\text{TOT}} = f_1(f_4(\mathcal{N})) + \sum_{i=1}^{N} f_1(n_i) + f_2(f_3(n_i)) \tag{4.5}$$

4.3 Description of Heuristics

The most obvious approach to find the solution that minimizes C_{TOT} is to perform a complete search in the solution space. However, this procedure, called Full Search (FS), is time consuming and not scalable. In this section, we quickly describe the full search and present four heuristics that can determine a near-optimal solution to the equipment allocation problem in a reasonable amount of time.

4.3.1 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 (4.6). We define $\mathbf{ub} = \{ub_i : i = 1, \ldots, N\}$ where ub_i is the upper-bound on the number of servers that represents the number of servers required to store the entire library and handle 100% of the requests ($h_i = 1.0$).

$$ub_i = \max\left(\frac{M_i}{F_i}, \frac{Y \cdot \text{file size}}{G_i}\right) \tag{4.6}$$

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 (complexity is discussed in more detail in Section 4.4).

4.3.2 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 4.1: Central 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 4.1; the Fully Distributed in Algorithm 4.2. This heuristic is straight-forward and highly suboptimal, but it provides an upper-bound that can used as a comparison base for other approaches.

```
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 $C_{\mathrm{TOT}_{\mathcal{V}^{'}}}(\mathcal{N}^{'})$;
8     if $C_{TOT_{\mathcal{V}^{'}}}(\mathcal{N}^{'}) < C_{FD}$ then $C_{FD} = C_{\mathrm{TOT}_{\mathcal{V}^{'}}}(\mathcal{N}^{'})$, $\mathcal{N}_{FD} = \mathcal{N}^{'}$, $\mathcal{V}_{FD} = \mathcal{V}^{'}$ ;
9 end
```

Algorithm 4.2: Fully Distributed Heuristic

4.3.3 Greedy Search (GS)

We define a 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 or changing the server model of the origin or any location. 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 [79]. The search continues until it reaches a local maximum (or minimum); no neighbours offer a better solution than the current one. We define $\mathcal{N} = \mathbf{0}$ as our initial solution, i.e., no servers installed at any of the locations. Then, at each iteration, the algorithm tries to place a server at each of the N locations and selects the placement that yields the lower cost. Because we also need to consider the server model, we adapted the greedy search to our new equipment allocation problem by making a few modifications, as shown in Algorithm 4.3.

For each node, not only do we try each of the N locations (lines 5-7), but also 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_1, 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_1 and v_i can change at every iteration.

Typically, if it is impossible to find a neighbour yielding a better solution than the current one, the search stops. 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 $C_j \geq C_{j-1}$ $\forall j \in k - I + 1 \ldots k$. Finally, another tactic to explore a larger

```
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
            $v_1' = w_j$ /* model at origin */;
            forall models $w_k \in \mathcal{W}$ do
                $v_i' = w_k$ /* model at location $i$ */;
                calculate cost $C_{\mathrm{TOT}_{\mathcal{V}'}}(\mathcal{N}')$;
                if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_{GS}$ then $C_{GS} = C_{\mathrm{TOT}_{\mathcal{V}'}}$, $\mathcal{N}_{GS} = \mathcal{N}'$, $\mathcal{V}_{GS} = \mathcal{V}'$ ;
                if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}') < C_k$ then $C_k = C_{\mathrm{TOT}_{\mathcal{V}'}}$, $\mathcal{N}_k = \mathcal{N}'$, $\mathcal{V}_k = \mathcal{V}'$ ;
            end
        end
    end
until $C_j \geq C_{j-1}$ $\forall j \in k - I + 1 \ldots k$;
```

Algorithm 4.3: Greedy Search (GS)

part of the solution space is to 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 4.3 becomes $n_i' = n_i - 1$. We then select the solution that produces the lowest cost.

4.3.4 Integer Relaxation Heuristic (IRH)

The Integer Relaxation Heuristic presented in Algorithm 4.4 is a modified version of the IRH presented in Section 3.2.3 of the previous chapter. As before, the first step is to find an initial non-integer solution and the second step is to search its neighborhood for a near-optimal integer solution. However, both steps have been adapted to this new problem. 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) is identical to the one described in Section 3.2.3: we iteratively try to remove or add up to two servers at each location until we find a local minimum.

4.3.5 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. Through simulations and results from Chapter 3, we noticed that the number of installed servers at a given location is either none or very close to the upper-bound. During the first step of the heuristic (lines 7-17 of Algorithm 4.5), 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

```
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 $C_{\mathrm{TOT}_{\mathcal{V}_j}}$;
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 $C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N})$;
    if $C_{TOT_{\mathcal{V}}} < C_{IRH}$ then $C_{IRH} = C_{\mathrm{TOT}_{\mathcal{V}}}$, $\mathcal{V}_{IRH} = \mathcal{V}$
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 $C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N}^{'})$;
        if $C_{TOT_{\mathcal{V}}}(\mathcal{N}^{'}) < C_{IRH}$ then $C_{IRH} = C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N}^{'})$, $\mathcal{N}_{IRH} = \mathcal{N}^{'}$ ;
    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 $C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N}^{'})$;
            if $C_{TOT_{\mathcal{V}}}(\mathcal{N}^{'}) < C_{IRH}$ then $C_{IRH} = C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N}^{'})$, $\mathcal{N}_{IRH} = \mathcal{N}^{'}$ ;
        end
    end
    $C_k = C_{IRH}$;
until $C_k \geq C_{k-1}$ ;
```

Algorithm 4.4: Integer Relaxation Heuristic (IRH)

```
Set $C_{IGS} = \infty$;
forall models $w_j \in \mathcal{W}$ do
    Set $\mathcal{V}'$: $v_i^{'} = w_j$ and calculate upper bounds $ub_i$ for $i = 1, \ldots, N$ ;
    forall models $w_k \in \mathcal{W}$ do
        Set $v_o^{'} = w_k$;
        Set $C_0 = \infty$ and $l = 0$;
        repeat
            $l++$;
            Set $\mathcal{N}^{'} = \mathcal{N}$;
            forall locations $i$ do
                Set $n_i^{'} = ub_i$ and calculate $C_{\mathrm{TOT}_{\mathcal{V}'}}(\mathcal{N}^{'})$;
                if $C_{TOT_{\mathcal{V}'}}(\mathcal{N}^{'}) < C_{IGS}$ then $C_{IGS} = C_{\mathrm{TOT}_{\mathcal{V}'}}, \mathcal{N}_{IGS} = \mathcal{N}^{'}, \mathcal{V}_{IGS} = \mathcal{V}^{'}$ ;
            end
            Set $C_l = C_{IGS}$;
        until $C_l \geq C_{l-1}$ ;
    end
end
Set $C_0 = C_{IGS}$, $\mathcal{N}_0 = \mathcal{N}_{IGS}$ and $\mathcal{V}_0 = \mathcal{V}_{IGS}$;
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
        for $m = -1$ and $m = 1$ do
            Set $\mathcal{N}^{'} = \mathcal{N}$ and $n_i^{'} = n_i + m$;
            calculate cost $C_{\mathrm{TOT}_{\mathcal{V}}}(\mathcal{N}^{'})$;
            if $C_{TOT_{\mathcal{V}}}(\mathcal{N}^{'}) < C_{IGS}$ then $C_{IGS} = C_{\mathrm{TOT}_{\mathcal{V}'}}, \mathcal{N}_{IGS} = \mathcal{N}^{'}, \mathcal{V}_{IGS} = \mathcal{V}^{'}$ ;
            if $C_{TOT_{\mathcal{V}}}(\mathcal{N}^{'}) < C_k$ then $C_k = C_{\mathrm{TOT}_{\mathcal{V}'}}, \mathcal{N}_k = \mathcal{N}^{'}, \mathcal{V}_k = \mathcal{V}^{'}$ ;
        end
    end
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 4.5: Improved Greedy Search (IGS)

(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).

4.4 Complexity Analysis

In this section, we analyze the worst-case complexity, WCC, of each of the heuristic 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.

4.4.1 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:

$$\begin{aligned} WCC_{FS} &= W^N \prod_i^N ub_i = W^N \cdot {U_{max}}^N \\ &= (W \cdot U_{max})^N \end{aligned} \tag{4.7}$$

In the case of the full search, 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. This justifies the development of the heuristics presented in this chapter.

4.4.2 CoFDH

CoFDH was written to generate an upper-bound and a comparison base for the solutions produced by 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).

$$WCC_{CoFDH} = 2W \tag{4.8}$$

4.4.3 GS

One iteration of the greedy search of Algorithm 4.3 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.

$$\begin{aligned} WCC_{GS} &= \sum_i^N ub_i \cdot (N \cdot W^2) = (N \cdot U_{max}) \cdot (N \cdot W^2) \\ &= N^2 W^2 U_{max} \end{aligned} \tag{4.9}$$

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

4.4.4 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) [81, 82] 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 4.4) of the heuristic 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 $\mathcal{N} = \mathbf{0}$; which requires up to N iterations. In the second part, each iteration requires five operations (trying each 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.

$$\begin{aligned} WCC_{IRH} &= (W \cdot N^2 + W) + (N^2) + 5N \sum_{i}^{N} ub_i \\ &= N^2(W + 1 + 5U_{max}) + W \end{aligned} \tag{4.10}$$

4.4.5 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 the one described in IRH: going from fully distributed to centralized or vice-versa.

$$\begin{aligned} WCC_{IGS} &= (W^2 N^2) + 2N \sum_{i}^{N} ub_i \\ &= (W^2 N^2) + 2N^2 U_{max} \end{aligned} \tag{4.11}$$

4.4.6 Worst-case heuristic comparison

We complete our analysis of the complexity by showing in Table 4.1 the WCC of all the heuristics described in this section. From this table, it is clear that a full search approach is unsuitable for our problem; even smaller problems such as $N = 5$, $W = 6$ and $U_{max} = 20$ take on the order of 10^9 operations. A large value of U_{max} is an indication of large worst-case demand M_i, files of large size or that the model is simply unfit for the specific location. The three other proposed approaches GS, IRH and IGS have reasonable worst-case complexity even for complex problems like $N = 100$, $W = 6$ and $U_{max} = 20$. We note that for most sample scenarios shown, the WCC of IRH and IGS together is still lower than running the GS. This leads us to think that it is possible to perform both searches and choose the best of the two solutions.

It is important to stress that the values and the expressions derived in this section are worst-case estimates and do not show the average complexity of these heuristics. The objectives were to provide an estimate of the maximum number of operations before con-

Table 4.1 Worst-case complexity for given N, W and U_{max}.

N	W	U_{max}	WCC_{FS}	WCC_{CoFDH}	WCC_{GS}	WCC_{IRH}	WCC_{IGS}
5	2	5	100,000	4	500	202	350
5	2	20	102,400,000	4	2,000	577	1,100
5	6	5	24,300,000	12	4,500	306	1,150
5	6	20	$2.4883 \cdot 10^{9}$	12	18,000	681	1,900
50	2	5	$1 \cdot 10^{50}$	4	50,000	20,002	35,000
50	2	20	$1.2677 \cdot 10^{80}$	4	200,000	57,502	110,000
50	6	5	$7.179 \cdot 10^{73}$	12	450,000	30,006	115,000
50	6	20	$9.1004 \cdot 10^{103}$	12	1,800,000	67,506	190,000
100	2	20	$1 \cdot 10^{100}$	4	200,000	80,002	140,000
100	2	20	$1.6069 \cdot 10^{160}$	4	800,000	230,002	440,000
100	6	5	$5.1538 \cdot 10^{147}$	12	1,800,000	120,006	460,000
100	6	20	$8.2818 \cdot 10^{207}$	12	7,200,000	270,006	760,000

vergence of our heuristics and confirm our intuition that the full search is unfit to solve this problem. The actual computational requirements are different than those estimates due to the different complexities of each iterations. In the next section, we compare the requirements of each heuristic by measuring the CPU time used during our simulations.

4.5 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 4.2 and choosing values for the other network parameters from uniform distributions with the ranges specified in Table 4.3. Simulations were executed on a AMD Athlon 3000+ with 1 GB of OCZ Premier Series 400 MHz Dual Channel memory.

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 compare the complexity and cost of our heuristics with the full search; other settings with larger inputs take too much time to solve (as shown in the previous section).

In Fig. 4.1, 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

Table 4.2 Values of constant variables used for the simulations.

Variable	Value
C_{IF}	10 k$
C_{DWDM}	25 k$
C_{LA}	10 k$
C_f	0.006 k$/km
dw_max	16
c	10 Gbps
max_{amp}	75 km
bit rate	3.75 Mbps
duration	5400 s
file size	2.53 GB

Table 4.3 Range of the variables used for the simulations.

Variable	Min	Max
d_{OR} (km)	0	50
d_{RC} (km)	0	5
Y (files)	1000	10000
Z (files/week)	0	100
priceGbps (k$/Gbps)	0	4
priceTB (k$/TB)	0	3
A (k$)	6	36
F (Gbps)	1	5
G (TB)	1	11
M (Gbps)	1	20

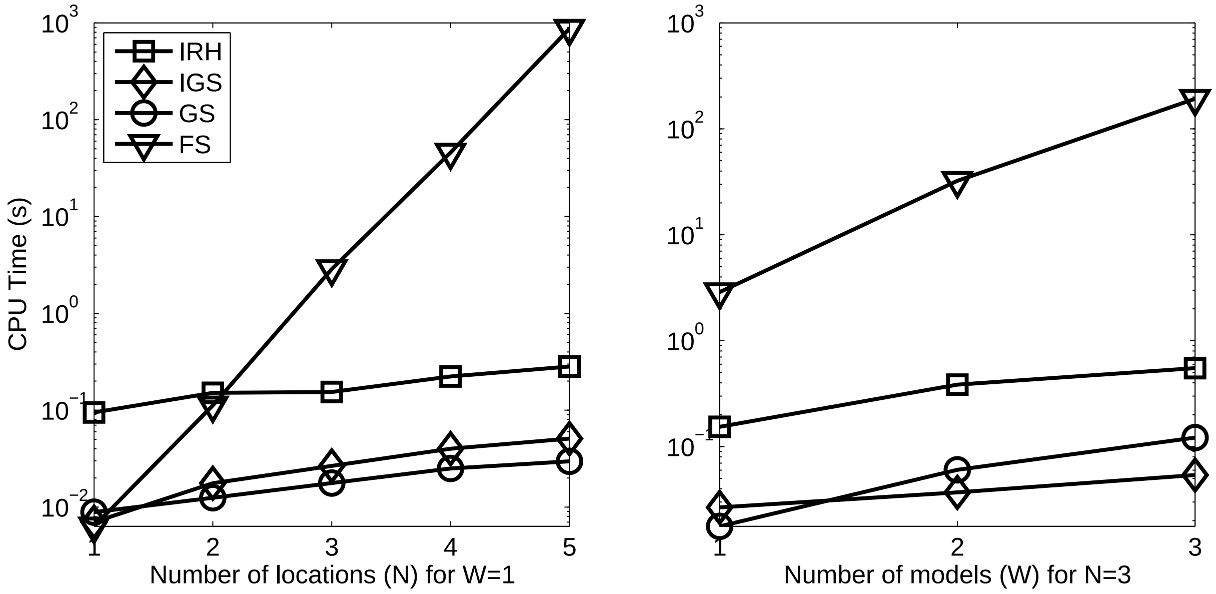


Fig. 4.1 Computational time in seconds required to find a solution by each of the heuristics averaged over 30 runs shown on a log-scale. Computational time of Full Search (FS) grows exponentially with the size of the network. Greedy Search (GS), Integer Relaxation Heuristic (IRH) and Improved Greedy Search (IGS) all provide solutions within 0.1 seconds.

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.

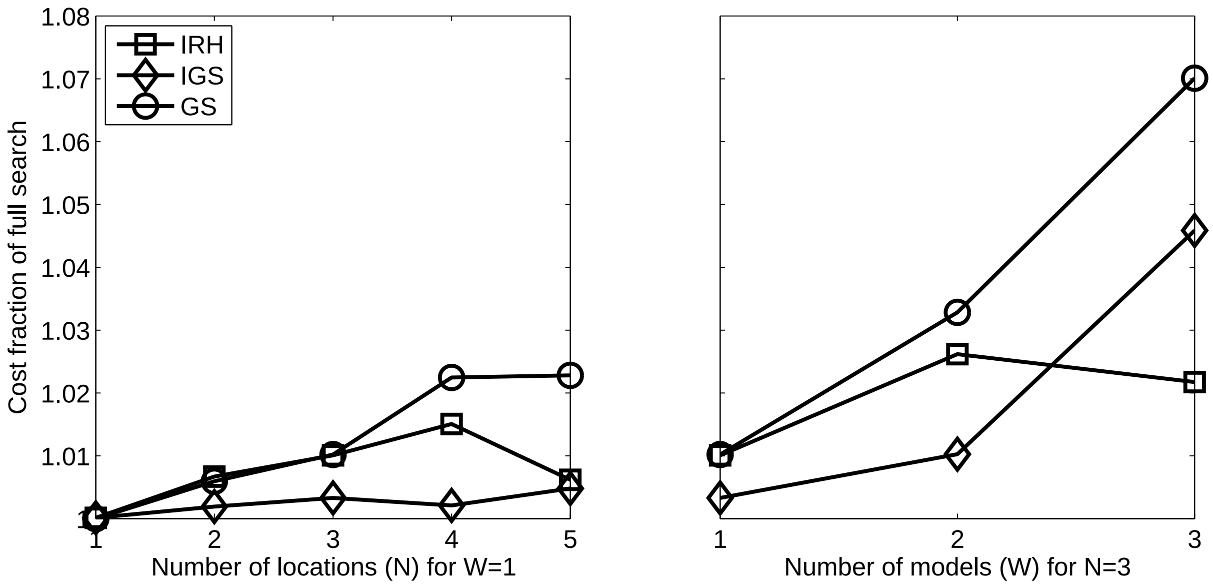


Fig. 4.2 Ratio between the cost of the heuristics solution and the full search (optimal) solution averaged over 30 runs.

In Fig. 4.2, 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.

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. CoFDH produces a very simple and quick solution by choosing the best our of a fully centralized (no replicas) and a fully distributed design.

In Fig. 4.3, we show values (averaged over 15 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

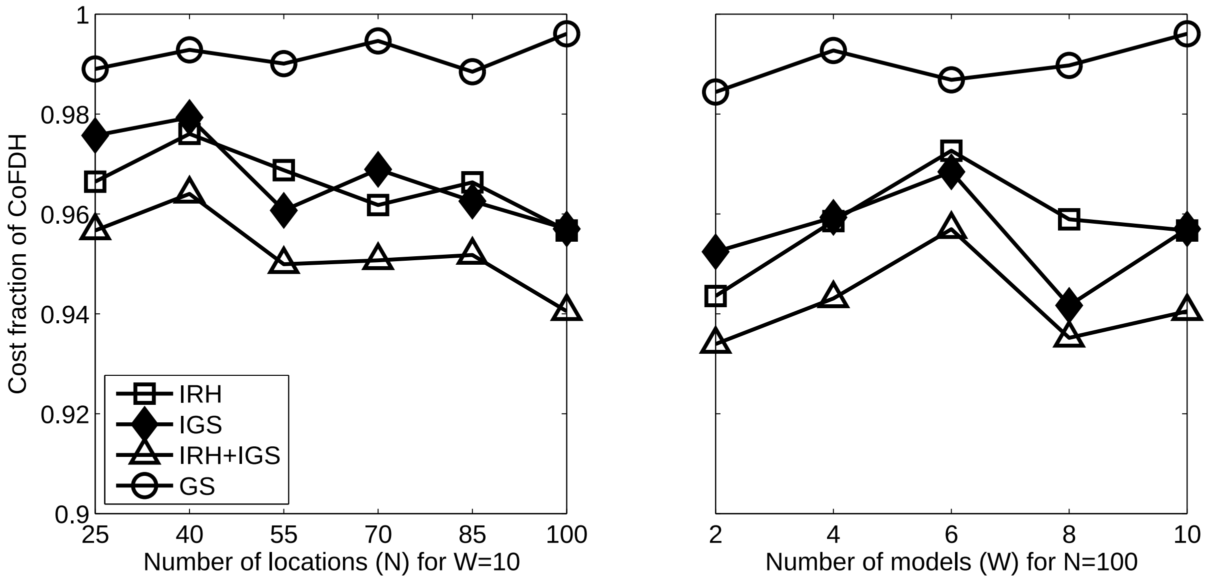


Fig. 4.3 Ratio between the cost of the heuristics solution and CoFDH averaged over 25 runs. IRH+IGS is the average of the minimum value between IRH and IGS for all runs.

combining both (choosing the best solution of the two), we obtain a slightly better heuristic (IRH+IGS): 4-6% of 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 improve cost.

For the same set of tests, we also show the complexity expressed as the computational time in seconds in Fig. 4.4 and the number of iterations (cost function evaluations) to obtain a solution in Fig. 4.5. As suggested by the Worst-Case Complexity analysis in section 4.4, 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, even if the number of iterations for Greedy Search is much larger than for IRH, their computational time is comparable in the left panel of Fig. 4.4. This is an indication that IRH's iterations take more time to execute than those in the greedy approaches (GS and IGS).

In Table 4.1, the WCC is lower for Integer Relaxation Heuristic than for the Improved Greedy Search, but for the most complex network we simulated, IGS produces a solution in less than half a minute and 50,000 iterations compared to the four minutes and 100,000 iterations taken by IRH. The Integer Relaxation Heuristic was the slowest of the tested heuristics, but it still converges in a reasonable amount of time. Since the computation

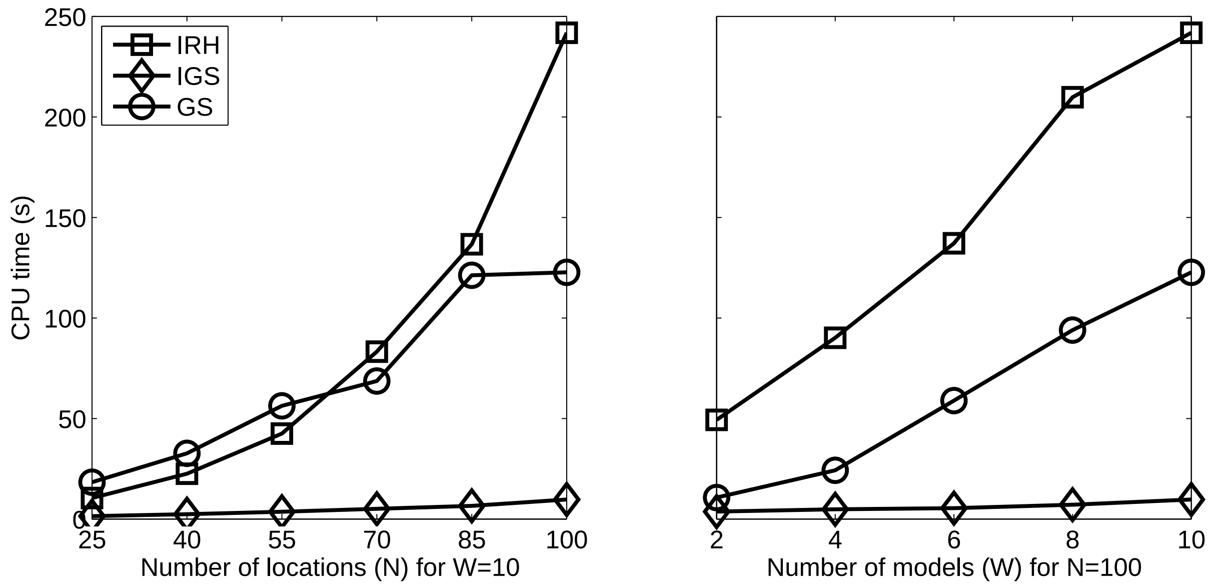


Fig. 4.4 CPU time in seconds for all heuristics averaged over 25 runs. IRH+IGS is the average of the sum of the time taken to perform both searches.

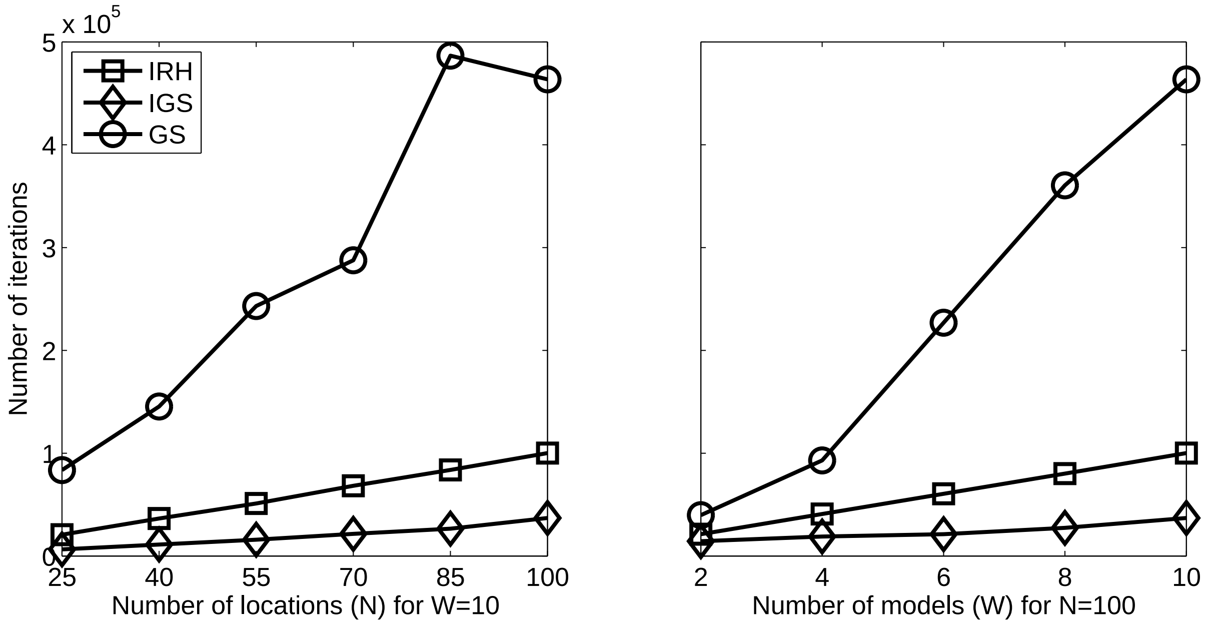


Fig. 4.5 Number of iterations (function evaluations) for all heuristics averaged over 25 runs. IRH+IGS is the average of the total number of iterations performed in both searches.

time of Improved Greedy Search is so low, we can combine IRH and IGS and obtain a solution in a timely fashion.

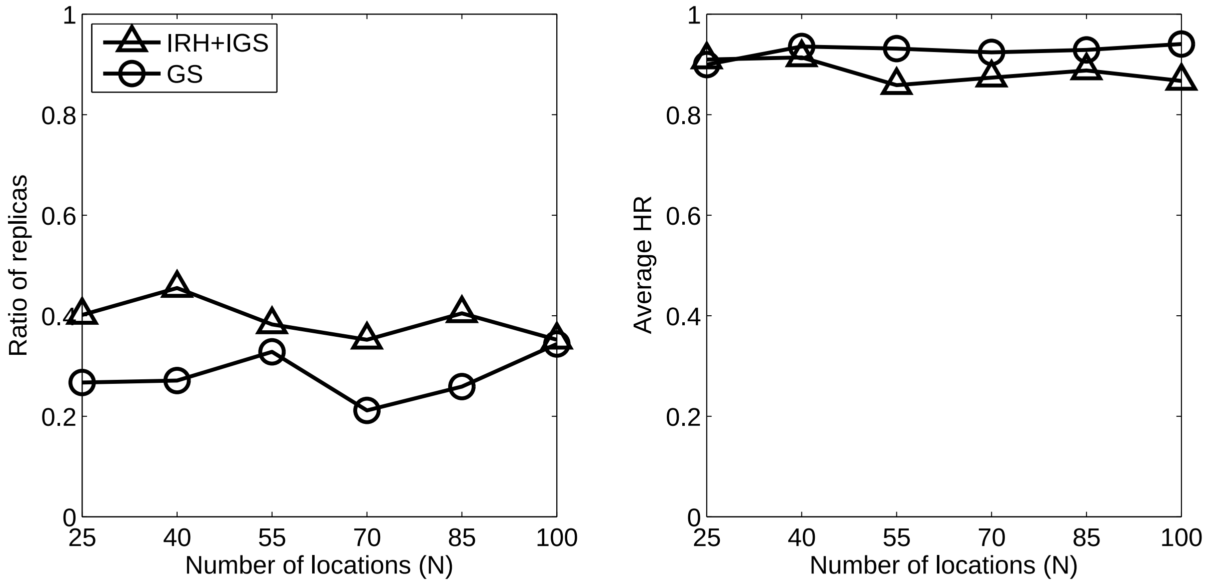


Fig. 4.6 LEFT: Ratio between the number of replicas and the number of locations (fraction of locations where content is cached). RIGHT: Average hit ratio at all the replicas. The values shown are averages of 25 runs with $W = 6$.

Finally, we focused 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. 4.6 and Fig. 4.7 show the results and provide interesting insights on the solution generated by the heuristics. The left panel of Fig. 4.6 shows that for networks of any size where demand is not uniformly distributed among all locations (i.e, 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 means that it is not always advantageous to cache content. Whether it is because the demand is too low or the site is too close to the origin, it might 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 4.4 where we show the number of servers installed at the origin. Because the fraction of locations where replicas are installed remains 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 4.4 Average number of VoD servers installed at the origin for different number of locations N. We show the average of the results obtained with the Greedy Search (GS) and by taking the best of the Integer Relaxation Heuristic and 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

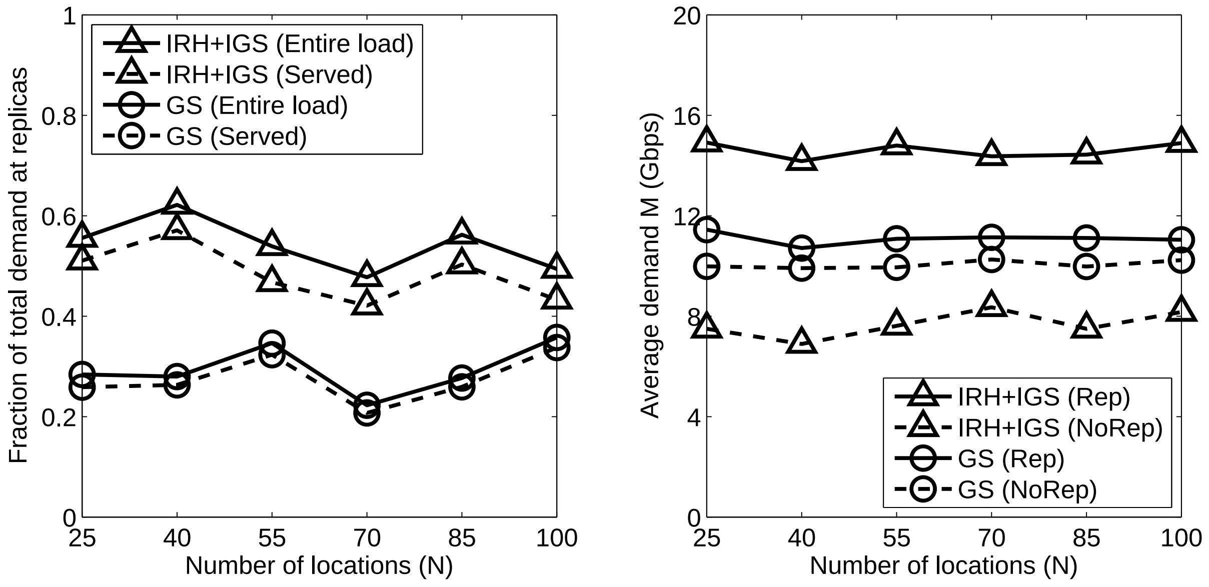


Fig. 4.7 LEFT: Fraction of the total network demand supported by replica locations. Total is the sum of the demand M_i at each location where a replica is installed and Real is the actual part of the demand that the replica handles $(M_i \cdot h_i)$. RIGHT: Average load on the locations where replicas 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. 4.6, 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. 4.7. 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. 4.6). 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. 4.7 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.

4.6 VoD in AAPN

Given that we proposed a solution to the *VoD equipment allocation problem*, we are now interested in validating our design choices by using the agile all-photonic network (AAPN) topology as an example. We also look at the advantages and disadvantages of using an AAPN as the core/backbone network to a video-on-demand deployment and describe the design process of such a network.

4.6.1 AAPN Architecture

An AAPN is a network in which the transmission and the switching through the core are done purely in the optical domain (all-photonic) [11, 84]. It is built using an overlaid star topology which connects all the edge nodes together using central core nodes (Fig. 4.8).

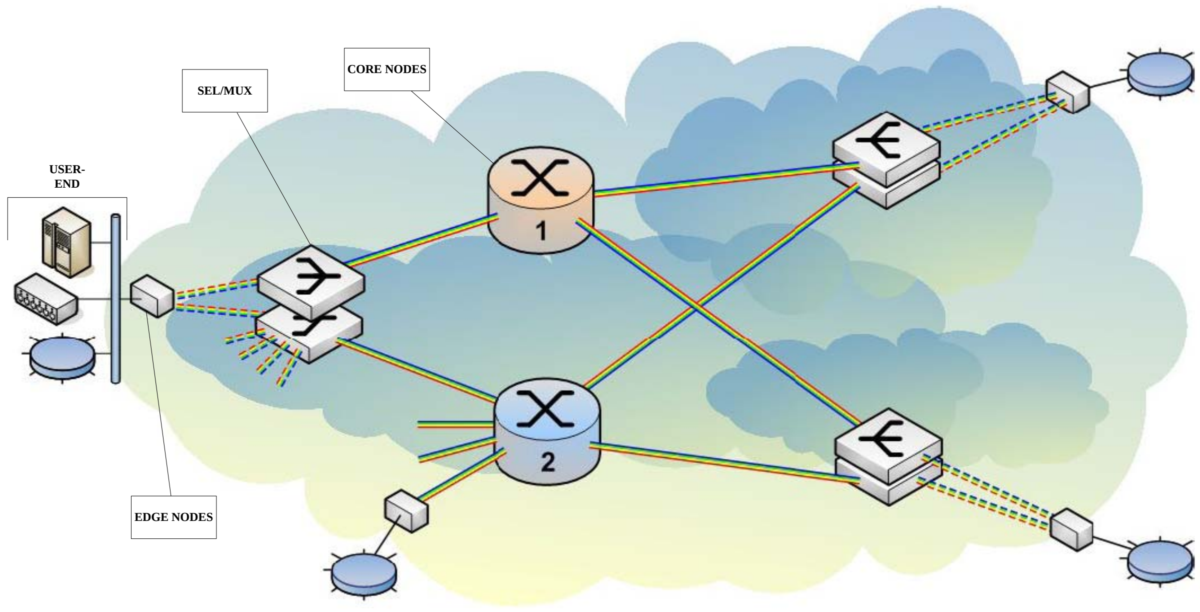


Fig. 4.8 The three-layer design of an agile all-photonic network (AAPN) includes edge nodes (switches that perform the O-E-O conversion), selector/multiplexor (Sel/Mux) devices, and all-photonic switches as the core nodes. The edge nodes are formed into sets and each set is connected to one or more Sel/Mux devices. Each Sel/Mux device is connected via DWDM equipment to one core node. (Reproduced from [11])

An edge node is the interface between the AAPN and the opto-electronic networks outside of the AAPN. These nodes can support a different number of wavelengths meaning that they do not all have necessarily the same traffic capacity. However, each node must be able to support a certain amount of traffic with every other edge node. All these edge nodes are connected to each other through more than one core nodes (for robustness). The core nodes are basically optical switches with an opto-electronic interface for control. The clients are connected to a single edge node (or second one for backup) directly or through a switch, which is the case in Fig. 4.8.

4.6.2 Analysis

The need for substantial bandwidth in the core of the network makes the AAPN topology a sensible candidate to support an application like video-on-demand. Based on its topology and our proposed architecture (see Section 3.1), we propose to collocate the replica servers with the edge nodes of the AAPN where the replica-client path is outside of the AAPN. The origin server is deployed near an edge-node collocated with the core node. Having the transmission path used to stream videos across larger distances (cache misses) and distribute (update) content at the replicas, traversing the AAPN is a clear advantage and should result in significant improvement in performance and reduction in cost. Because re-routing a cache miss to the origin or any other replica is equivalent in an AAPN, we suggest to implement a mechanism to share the load among all the replicas and the origin.

We consider two different scenarios for the design of a VoD network over an AAPN. First, we consider the case of an existing AAPN where the edge and core node locations have already been decided. In that case, VoD traffic is allocated a fraction of the overall AAPN traffic, thereby putting a constraint on the load from the origin to the replicas. We enforce this constraint by putting a lower-bound on the hit ratio of each replica based on the demand and calculate the minimum number of VoD servers to achieve such a hit ratio. This effectively reduces the solution space because the valid range for the number of servers at each location is smaller. The other case is the one where the AAPN and the VoD network are jointly designed. As it is anticipated that the video-on-demand network accounts for a substantial portion of the AAPN traffic, it influences the location of the AAPN edge nodes. The origin servers definitely generate a large amount of traffic for the distribution of objects to replica servers or for the delivery to users. Thus, it makes sense to collocate AAPN edge nodes with origin servers. Also, the users for a VoD system are mainly located in residential areas, which is typically not the main source for other types of network traffic, so the presence of a video-on-demand service changes the traffic pattern in the network.

4.7 Concluding remarks

In this chapter, we defined an extension of the *VoD equipment allocation problem* described in Chapter 3. 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 defined in Section 3.2.2 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 a modified version of the integer relaxation method (IRH) presented in the previous chapter. 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 (almost instantaneous) or combining IRH and IGS for a better solution that takes more time.

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 suggest 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. In the next chapter, we discuss these results and possible extensions to our design tool in more detail and describe the design process of a VoD network over an AAPN.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we focused on resource allocation during the network planning of a video-on-demand deployment. More specifically, we addressed the *VoD equipment allocation problem* of determining the number of storage and streaming devices needed at each potential replica location in a metropolitan-area network.

As a first step to solving that problem, in **Chapter 2** we reviewed previously proposed approaches for the delivery of multimedia objects. Depending on the network architecture, many aspects have to be considered to deploy a complete media delivery solution. A centralized architecture, in which a unique media server handles the entire demand, is the most simple solution, but it has serious weaknesses: a single point of failure and high load on one server and the backbone network. Many proxy-based solutions have been proposed to reduce both the latency at the user-end and the load on the origin by caching content at servers located closer to the clients. The trade-off is the complexity of the design; we presented solutions to the replica placement problem to determine the optimal location of proxy servers in the topology. To solve that problem, we must determine a cost function for the transport and storage of the media objects that depends on the content cached at each replica, the delivery protocol and the clients' requests handling mechanism. Due to the size of multimedia objects, it is problematic and costly to replicate the entire library at each site. The analysis of video rental statistics showed that a large fraction of the requests are for only a small portion of the library. It therefore makes sense to cache only the most popular content at the replicas. When performing program caching, it is important to have a proper mechanism to serve requests and to handle cache misses. When the requested content is not present at the replica, the request is routed either to the origin server or to another replica. Forming clusters or replica minimizes the number of requests that are routed to the origin, but a directory of the objects cached at each location must be maintain to direct requests properly. The streaming capacity at each replica depends on the chosen delivery protocol. Using unicast delivery is the simplest approach, but it consumes a significant amount of bandwidth. For that reason, authors have proposed to use multicast to reduce bandwidth requirements at the replicas and on the network.

In **Chapter 3**, we presented our solution to the *VoD equipment allocation problem*. We chose an architecture where the population is partitioned and each partition is assigned to a specific replica. We estimate the load at each location with the worst-case demand: the bandwidth required to serve all requests at peak hours using unicast delivery. If the replica does not have the requested content, the origin delivers the movie to the client. To avoid low-utilization of the resources, we use available bandwidth during off-peak hours to distribute and update content from the origin to the replicas. We developed a hit ratio function, cost function and heuristic integrated in an interactive design tool to solve the *VoD equipment allocation problem*. We *trained a parametric function* that generates accurate estimates of the hit ratio for given cache size, library size and file arrival rate and then *constructed a cost function* based on the hit ratio, the worst-case distributed demand and the number of VoD servers n_i at each location. To find a configuration that minimizes this cost, we *developed the Integer Relaxation Heuristic* that produces a non-integer initial solution and then searches its neighbourhood for a near-optimal integer solution.

Through simulations, we discovered that the model of installed equipment has a direct impact on the minimum demand that makes caching profitable. For that reason, in **Chapter 4**, we relaxed the assumption that the specifications (streaming and storage capacity) of the VoD server were fixed and pre-determined before the optimization. Instead, we require the pre-selection of a set of available VoD servers; the optimization determines which model should be installed at each location. In Section 4.1, we *generalized the VoD equipment allocation problem* as determining both the number and the model of the VoD servers to install at each potential replica location. Due to the higher complexity of this problem, new algorithms are required to generate a solution. We *described three heuristics* to find a near-optimal solution including a modified version of the Integer Relaxation Heuristic (IRH) presented in Chapter 3. The basic idea behind IRH remains the same, but we changed it so that the non-integer initial solution takes the set of available VoD server models into account. The two new heuristics are based on the greedy search approach. Our Greedy Search (GS) consists of adding one VoD server of any model at every iteration at the location which minimizes the total cost for that particular iteration. Whereas greedy search algorithms usually terminate when the placement of an additional server no longer reduces cost, we allow the search to continue for more iterations to explore a larger portion of the solution space. We developed an Improved Greedy Search (IGS) heuristic that uses greedy search tactics to generate an initial solution and to search its neighbourhood for a solution with a lower cost. We observed that it has lower complexity and is faster than IRH in practice, but does not always generate a better solution. By taking the best of the Integer Relaxation Heuristic and Improved Greedy Search designs, we produce a near-optimal solution in a timely manner.

5.2 Discussion

In Chapter 3, we described and proposed a solution to the simplified *VoD equipement allocation problem* of determining the number of VoD servers to deploy at each potential replica location in the given topology. Our results showed that the nature of the type of equipment installed at each location has a significant impact on the optimal design and the deployment cost. In Chapter 4, we extended the problem to a case where a set of available VoD server models for all locations is provided instead of having fixed and pre-determined streaming and storage capacity at each location. For networks where the demand is not evenly distributed among all locations, we noted that is was beneficial to cache content in only a small fraction of the locations for a given set of available VoD server models.

This leads to the following question: should the hardware manufacturer develop custom equipment or, if possible, should network engineers design topologies based on the available equipment at their disposal? From our perspective, the problem of jointly designing the VoD network and the logical topology is a very interesting and challenging one and represents the sensible extension to the resource allocation problem we addressed in this thesis. This problem consists of choosing a topology that allows an allocation of resources that minimizes the deployment cost of the network. Whereas throughout this thesis we assumed a given set of inter-nodal distances, potential replica location positions and distributed worst-case demands, in this problem, these variables are unknown and the number and position of the replica locations become part of the set of optimization variables with the number and model of the VoD servers. Not only is this a much more complex problem to solve, but it also introduces some new issues such as establishing a request routing mechanism and possibly forming and maintaining replica clusters.

In this thesis, we considered 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. For example, a provider who owns a backbone network such as AAPN is interested in offering video-on-demand. Even if there is no installation cost, there is 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.

We focused on large-scale deployments, but there is also the issue of scalability of such deployments. We assumed a growth in the library sizes and usage on video-on-demand services, but it is hard to predict the exact impact that this expansion will have on the designs. As the library reaches tens of thousands of assets, the access model we assumed changes as a larger portion of requests are located in the heavy tail of the popularity distribution. 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.

5.3 Future Work

We observed that it is difficult to select a model that matches the requirements of each location even when a set of many VoD server models is available. For that reason, an extension for the design tool is to determine how the topology should be designed, how the demand should be shared among the locations, for a given a set of available equipment. In Section 4.6.2, we presented two scenarios to consider for the design of a VoD network over an AAPN. We propose to adopt an iterative process for the joint design. First, we decide upon the location of the AAPN edge nodes based on a prior model for the traffic pattern in the network. We then solve the *VoD equipement allocation problem* for a specific demand. This placement changes the traffic pattern, so we repeat the AAPN topology design step (placement of edge nodes) for the new model of traffic demand. This process is repeated to adjust the locations according to the performance of the prior setup until a local minimum is reached.

In the previous section, we presented a scenario where the service provider owns infrastructures prior to the deployment. Our tool needs to be extended to support this scenario by including usage and maintenance costs for bandwidth and infrastructures. To do so, we need to either add components to the cost function to model these fees or modify it completely such that it is expressed as the cost of using (rather than installing) equipment for storing, streaming and transporting the data. By adding those features to the tool, we could address other problems such as the delivery and distribution in a peer-to-peer architecture similar to that presented in Section 2.1. In that case, no or very few replicas are required, but the installation of equipment for transport might be required and the cost of usage and maintenance definitely need to be included.

Because library size and usage of video-on-demand services will grow, providers are interested in the scalability of a deployment during the design. For larger libraries, the file access model and popularity distribution are different and affect the hit ratio function we designed. To asses that effect, we redefine a file access model and popularity distribution based on usage/rental statistics of video-on-demand services. Then, we train a new parametric hit ratio function by following the procedure described in Section 3.2.1. As usage increases, the load on the origin server becomes very high and more streaming capacity is required. The tool can be extended to impose an upper-bound on the number of servers at the origin to share the load among all locations. This is done by modifying the heuristics to avoid searching the regions that are no longer in the solution space (solutions that yield $n_o >$ upper-bound). The extended tool supports the constraints to simply flag a solution judged infeasible, but still provides sufficient information for the user to *gain better understanding of resource allocation for video-on-demand deployment*, which we feel was the main contribution of this work.

References

[1] G. Peng, "Cdn: Content distribution network," State University of New York (SUNY) at Stony Brook, Stony Brook, NY, Tech. Rep., Jan. 2003, research Proficiency Exam report.

[2] M. Yang and Z. Fei, "A model for replica placement in content distribution networks for multimedia applications," in *Proc. IEEE Int. Conf. Communications*, Anchorage, AK, May 2003.

[3] S. V. Rompaey, K. Spacy, and C. Blondia, "Bandwidth versus storage trade-off in a content distribution network and a single server system," in *Proc. Conf. Telecommunications*, Zagreb, Croatia, June 2003.

[4] A. Vakali and G. Pallis, "Content delivery networks: Status and trends," *IEEE Internet Computing 7*, vol. 6, pp. 68–74, Nov. 2003.

[5] N. Bartolini, F. Presti, and C. Petrioli, "Optimal dynamic replica placement in content delivery networks," in *Proc. IEEE Int. Conf. on Networking*, Sydney, Australia, Sept. 2003.

[6] K. Hosanagar, R. Krishnan, M. Smith, and J. Chuang, "Optimal pricing of content delivery network (CDN) services," in *Proc. Hawaii Int. Conf. System Sciences*, Big Island, Hawaii, Jan. 2004.

[7] S. Buchholz and T. Buchholz, "Replica placement in adaptive content distribution networks," in *Proc. Symp. Applied Computing*, Nicosia, Cyprus, Mar. 2004.

[8] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky, "A transport layer for live streaming in a content delivery network," *Proc. IEEE*, vol. 92, pp. 1408–1419, Sept. 2004.

[9] P.Lyman and H.R.Varian. (2003) How much information 2003? Study at School of Information Management and Systems at the University of California at Berkeley. [Online]. Available: http://www.sims.berkeley.edu/research/projects/how-much-info-2003

[10] 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.

[11] L. Mason, A. Vinokurov, N. Zhao, and D. Plant, “Topological design and dimensioning of agile all photonic networks,” *Computer Networks, Special issue on Optical Networking*, vol. 50, pp. 268–287, Feb. 2006.

[12] S. Tsao and Y. Huang, “An efficient storage server in near video-on-demand systems,” *IEEE Trans. Consumer Electronics*, vol. 44, pp. 27–32, Feb. 1998.

[13] T. Chiueh and C. Lu, “A periodic broadcasting approach to video-on-demand service,” in *Proc. SPIE*, vol. 2615, Oct. 1995, pp. 162–169.

[14] J. Lee, “On a unified architecture for video-on-demand services,” *IEEE Trans. Multimedia*, vol. 4, pp. 38–47, Mar. 2002.

[15] 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.

[16] S. A. Barnett and G. J. Anido, “A cost comparison of distributed and centralized approaches to video-on-demand,” *IEEE J. Selected Areas in Communications*, vol. 14, pp. 1173–1183, 1996.

[17] M. M. Hefeeda, B. K. Bhargava, and D. K. Y. Yau, “A hybrid architecture for cost-effective on-demand media streaming,” *Computer Networks*, vol. 44, pp. 353–382, 2004.

[18] D. A. Tran, K. A. Hua, and S. Sheu, “A new caching architecture for efficient video-on-demand services on the internet,” in *Proc. Symp. on Applications and the Internet*, Orlando, FL, Jan. 2003.

[19] P. Machanick, “Design of a scalable video on demand architecture,” in *Proc. South African Institute of Computer Scientists and Information Technologists (SAICSIT)*, Gordon’s Bay, South Africa, Nov. 1998.

[20] P. Mundur, R. Simon, and A. Sood, “End-to-end analysis of distributed video-on-demand systems,” *IEEE Trans. Multimedia*, vol. 6, pp. 129–141, Feb. 2004.

[21] M. Ditze, C. Loeser, P. Altenbernd, and K. Wan, “Improving content replication and QoS in distributed peer-to-peer VoD appliances,” in *Proc. Int. Conf. on Distributed Computing Systems (ICDCS)*, Tokyo, Japan, Mar. 2004.

[22] J. Nussbaumer, B. Patel, F. Schaffa, and J. Sterbenz, "Networking requirements for interactive video on demand," *IEEE J. Selected Areas in Communication*, vol. 13, pp. 779–787, 1995.

[23] C. Griwodz, M. Bar, and L. Wolf, "Long-term movie popularity models in video-on-demand systems: or the life of an on-demand movie," in *Proc. ACM Int. Conf. Multimedia*, Seattle, WA, Nov. 1997.

[24] Akamai. [Online]. Available: http://www.akamai.com

[25] C. Yuan, Y. Chen, and Z. Zhang, "Evaluation of edge caching/off loading for dynamic content delivery," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, pp. 1411–1423, Nov. 2004.

[26] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," in *Proc. IEEE Infocom*, Miami, FL, Mar. 2005.

[27] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed content delivery across adaptive overlay networks," *IEEE/ACM Trans. on Networking (TON)*, vol. 12, pp. 767–780, Oct. 2004.

[28] F. Schaffa and J.-P. Nussbaumer, "On bandwidth and storage tradeoffs in multimedia distribution networks," in *Proc. IEEE Infocom*, Boston, MA, Apr. 1995.

[29] J. Lu, "An architecture for delivering broadband video over the Internet," in *Proc. Int. Symp. Information Technology (ITCC)*, Las Vegas, NV, Apr. 2002.

[30] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. Katz, "Efficient and adaptive web replication using content clustering," *IEEE J. Selected Areas Communication*, vol. 21, pp. 979–994, Aug. 2003.

[31] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," in *Proc. IEEE Infocom*, New York, NY, June 2002.

[32] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE Infocom*, New York, NY, Mar. 1999.

[33] J. Almeida, D. Eager, M. Ferris, and M. Vernon, "Provisioning content distribution networks for streaming media," in *Proc. IEEE Infocom*, New York, NY, June 2002.

[34] N. Jian, D. Tsang, I. Yeung, and H. Xiaojun, "Hierarchical content routing in large-scale multimedia content delivery network," in *Proc. IEEE Int. Conf. Communications (ICC)*, Anchorage, AK, May 2003.

[35] S. Ramesh, I. Rhee, and K. Guo, “Multicast with cache (mcache): An adaptive zero delay video-on-demand service,” in *Proc. IEEE Infocom*, Anchorage, AK, Apr. 2001.

[36] G. K. Zipf, *Human Behavior and the Principal of Least-Effort.* Cambridge, MA: Addison-Wesley, 1949.

[37] (2000, March) Video store magazine. Avanstar Communications. [Online]. Available: http://www.videostoremag.com

[38] D. S. A. Dan and P. Shahabuddin, “Scheduling policies for an on-demand video server with batching,” in *Proc. ACM Int. Conf. Multimedia*, San Francisco, CA, Oct. 1994.

[39] K. A. Hua and S. Sheu, “Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems,” in *Proc. ACM SIGCOMM*, Cannes, France, Sept. 1997.

[40] J. Segarra and V. Cholvi, “Distribution of video-on-demand in residential networks,” in *Proc. Int. Workshop Interactive Disttributed Multimedia Systems (IDMS)*, Lancaster, UK, Sept. 2001.

[41] J. M. Almeida, J. Krueger, D. L. Eager, and M. K. Vernon, “Analysis of educational media server workloads,” in *Proc. ACM Int. Workshop Networks and Operating Systems Support for Digital Audio Video (NOSSDAV)*, Danfords on the Sound, NY, June 2001.

[42] M. Goncalves and K. Niles, *IP Multicasting: Concepts and Applications.* McGraw-Hill, 1998.

[43] J. Lichtenberg and J. Gomez, “Multicast based client for video-on-demand services - a case study,” in *Proc. IEEE Region 3 Technical, Professional, and Student Conference (SoutheastCon)*, Fort Lauderdale, FL, Apr. 2005.

[44] C. Aggarwal, J.Wolf, and P.Yu, “On optimal batching policies for video-on-demand storage servers,” in *Proc. Int. Conf. Microelectronics Computer Science (ICMCS)*, Hiroshima, Japan, June 1996.

[45] 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.

[46] S. Carter and D. Long, “Improving video-on-demand server efficiency through stream tapping,” in *Proc. IEEE Int. Conf. Computer Communications Networks (ICCCN)*, Las Vegas, NV, Sept. 1997.

[47] K. A. Hua, Y. Cai, and S. Sheu, “Patching: a multicast technique for true video-on-demand services,” in *Proc. ACM Multimedia*, Bristol, England, Sept. 1998.

[48] L. Gao and D. F. Towsley, “Supplying instantaneous video-on-demand services using controlled multicast,” in *Proc. Int. Conf. Microelectronics Computer Science (ICMCS)*, Florence, Italy, June 1999.

[49] D. Milic, M. Brogle, and T. Braun, “Video broadcasting using overlay multicast,” in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Irvine, CA, Dec. 2005.

[50] Y.-H. Chu, S. G. Rao, and H. Zhang, “A case for end system multicast,” in *Proc. ACM Sigmetrics*, Santa Clare, CA, June 2000.

[51] J. Ni and D. Tsang, “Large-scale cooperative caching and application-level multicast in multimedia content delivery networks,” *IEEE Communications Magazine*, vol. 43, pp. 98–105, May 2005.

[52] C. C. Hsu, T. Aoki, and H. Yasuda, “Distributing video content using a router-assisted multicast mechanism,” in *Proc. IEEE Pacific Rim Conf. Communications, Computers and Signal Processing (PACRIM)*, Victoria, Canada, Aug. 2003.

[53] U. Sarkar, S. Ramakrishnan, and D. Sarkar, “Modeling full-length video using markov-modulated gamma-based framework,” *IEEE/ACM Trans. on Networking*, vol. 11, pp. 638–649, Aug. 2003.

[54] K. Couch. (2005, January) Raising the bar for triple play with VoD. [Online]. Available: http://www.convergedigest.com/blueprints/ttp03/2005nortel1.asp?ID=189&ctgy=Headend

[55] D. E. Wrege, E. W. Knightly, H. Zhang, and J. Liebeherr, “Deterministic delay bounds for VBR video in packet-switching networks: fundamental limits and practical trade-offs,” *IEEE/ACM Trans. on Networking*, vol. 4, pp. 352–362, June 1996.

[56] R. Cruz, “A calculus for network delay, part I: Network elements in isolation,” *IEEE Trans. on Information Theory*, vol. 37, pp. 114–131, January 1991.

[57] E. Knightly and H. Zhang, “Traffic characterization and switch utilization using deterministic bounding interval dependent traffic models,” in *Proc. IEEE Infocom*, Boston, MA, April 1995.

[58] J. Liebeherr, D. E. Wrege, and D. Ferrari, “Exact admission control for networks with a bounded delay service,” *IEEE/ACM Trans. on Networking*, vol. 4, pp. 885–901, Dec. 1996.

[59] H. Zhang and D. Ferrari, “Improving utilization for deterministic service in multimedia communications,” in *Proc. Int. Conf. Microelectronics Computer Science (ICMCS)*, Boston, MA, May 1994.

[60] C. Lee, C. Lin, and P. Chang, "An improved traffic modeling scheme for MPEG video over content delivery networks," in *Proc. IEEE Int. Conf. Computational Science (ICCS)*, Singapore, Nov. 2002.

[61] W. Tan and A. Zakhor, "Packet classification schemes for streaming MPEG video over delay and loss differentiated networks," in *Proc. Int. Packet Video Workshop*, Kyongju, Korea, May 2001.

[62] J. Zhao, B. Li, and I. Ahmad, "Traffic modeling for layered video," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Baltimore, MD, July 2003.

[63] W. Zhou, S. Ramakrishnan, D. Sarkar, and U. Sarkar, "Bandwidth estimation for multiplexed videos using MMG-based single video traffic model," in *Proc. IEEE Globecom*, San Francisco, CA, Dec. 2003.

[64] Q. Zhang, C. Lin, H. Yin, and Q.-H. Dai, "An accurate scene-based traffic model for mpeg video stream," in *Proc. IEEE Int. Conf. Electronics, Circuits and Systems (ICECS)*, Sharjah, United Arab Emirates, Dec. 2003.

[65] R. Janakiraman, M. Waldvogel, and L. Xu, "Fuzzycast: Efficient video-on-demand over multicast," in *Proc. IEEE Infocom*, New York, NY, Jun. 2002.

[66] M. Masa and E. Parravicini, "Impact of request routing algorithms on the delivery performance of content delivery networks," in *Proc. Int. Performance Computing Communications Conf. (IPCCC)*, Phoenix, AZ, Apr. 2003.

[67] R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proc. IEEE Infocom*, Kobe, Japan, Apr. 1997.

[68] J. S. Chase, "Server switching: Yesterday and tomorrow," in *Proc. IEEE Workshop Internet Applications (WIAPP)*, San Jose, CA, July 2001.

[69] A. Rousskov and D. Wessels, "Cache digests," *Computer Networks and ISDN Systems*, vol. 30, pp. 2155–2168, 1998.

[70] S. Gadde, M. Rabinovich, and J. S. Chase, "Reduce, reuse, recycle: An approach to building large internet caches," in *Proc. Workshop Hot Topics Operating Systems (HotOS)*, Cape Cod, MA, May 1997.

[71] V. Valloppillil and K. W. Ross, "Cache array routing protocol v1.0," Internet draft, Feb. 1998. [Online]. Available: http://icp.ircache.net/carp.txt

[72] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," *Computer Networks*, vol. 31, pp. 1203–1213, 1999.

[73] M. Karlsson, C. Karamanolis, and M. Mahalingam, “A unified framework for evaluating replica placement algorithms,” Hewlett-Packard Laboratories,” Technical report, 2002.

[74] 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.

[75] S.-J. Kim and M. Choi, “A genetic algorithm for server location and storage allocation in multimedia-on-demand network,” in *Proc. Symp. Trends in Communications*, Bratislava, Slovakia, Oct. 2003.

[76] 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.

[77] T. Nguyen, C. Chou, and P. Boustead, “Resource optimization for content distribution networks in shared infrastructure environment,” in *Proc. Australian Telecommunications Networks and Applications Conf.*, Melbourne, Australia, Dec. 2003.

[78] M. Atallah, Ed., *Algorithms and Theory of Computation Handbook.* CRC Press LLC, 1999, page 19-26.

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

[80] P. Radoslavov, R. Govindan, and D. Estrin, “Topology-informed internet replica placement,” in *Proc. IEEE Workshop Web Content Caching and Distribution (WCW)*, Boston, MA, June 2001.

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

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

[83] A. Vinokurov, “Tools for optical networks design,” in *Proc. European Next Generation Internet Design and Engineering (EURO-NGI)*, Rome, Italy.

[84] G. Bochmann, M. Coates, T. Hall, L. Mason, R. Vickers, and O. Yang, “The agile all-photonic network: An architectural outline,” in *Proc. Queen’s Biennial Symp. on Communications*, Kingston, Canada, May 2004.