Joint Optimization for the Delivery of Multiple Video Channels in Telco-CDNs

Fen Zhou, Jiayi Liu, Gwendal Simon, and Raouf Boutaba

***Abstract*—The delivery of live video channels for services such as twitch.tv leverages the so-called Telco-CDN—Content Delivery Network (CDN) deployed within the Internet Service Provider (ISP) domain. A Telco-CDN can be regarded as an intra-domain overlay network with tight resources and critical deployment constraints. This paper addresses two problems in this context: (1) the construction of the overlays used to deliver the video channels from the entrypoints of the Telco-CDN to the appropriate edge servers; and (2) the allocation of the required resources to these overlays. Since bandwidth is critical for entrypoints and edge servers, our ultimate goal is to deliver as many video channels as possible while minimizing the total bandwidth consumption. To achieve this goal, we propose two approaches: a two-step optimization where the optimal overlays are firstly computed, then an optimal resource allocation based on these pre-computed overlays is performed; and a joint optimization where both optimization problems are simultaneously solved. We also devise fast heuristic algorithms for each of these approaches. The conducted evaluations of these two approaches and algorithms provide useful insights into the management of critical Telco-CDN infrastructures.**

***Index Terms*—Content Delivery Networks (CDNs), video delivery, joint optimization, Mixed Integer Linear Programming (MILP), heuristic algorithms.**

I. Introduction

IT has become clear over the past couple of years that Internet Service Providers (ISP) have to deploy a Content Delivery Network (CDN) *within* their network infrastructure to cope with the sharp increase of video traffic. The recent OpenConnect [2] proposal from Netflix illustrates this major shift: content providers now develop peering agreements with ISPs, and rely on co-controlled delivery platforms to serve end-users. As a matter of fact, the management of video streams in *Telco-CDNs* has become a critical topic for both network operators and content providers.

Telco-CDNs differ from traditional CDNs on several aspects. First, Telco-CDNs can be fully controlled, since ISPs own the network and the entire "last mile" from edge servers to end users. This enables to engineer Telco-CDNs through centralized optimization techniques so that the Quality of Service (QoS) can be guaranteed. Second, a Telco-CDN matches the topology of an underlying network. Consider the case of a French network operator (see Fig. 1). The "*entrypoints*" for the content providers are the Internet Exchange Points (IXP) where peering occurs (here the rectangular nodes). *Edge servers* in Telco-CDN are typically deployed near the routers that connect the ISP backbone core network to the metropolitan access networks of the different regions (the circular nodes). Thus each edge server is in charge of a regional area. However, the congestion of the core network is a serious concern for network operators, so content delivery within the Telco-CDN—between entrypoints and edge servers—is an essential operational and management concern.

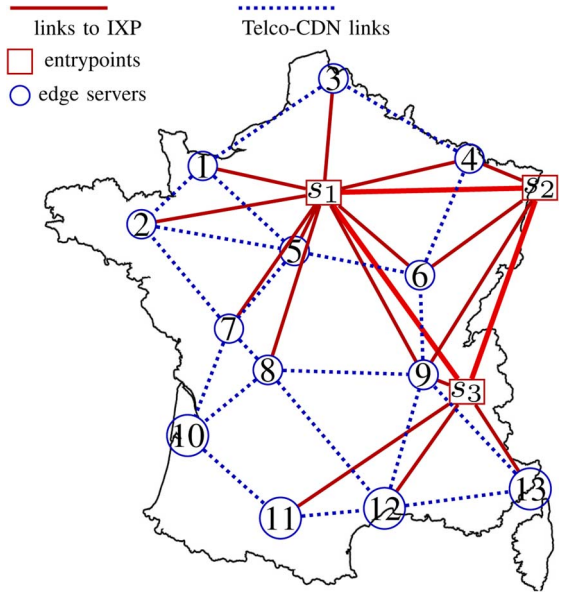


Fig. 1. Topology of a French Telco-CDN [4].

A key challenge for Telco-CDNs is the management of the traffic generated by *live video channels* from Over-The-Top (OTT) video services. OTT content providers include user-generated video platforms like twitch.tv or ustream, and also second-screen video services affiliated with traditional TV broadcasters. Traffic generated by these services is exploding. Moreover, the adoption of rate-adaptive streaming technologies (e.g., DASH) negatively affects the CDN infrastructure because the bit-rate of a video channel corresponds to the accumulated rate of all the video representations, which is frequently greater than 20 Mbps [3].

Manuscript received January 30, 2014; revised November 3, 2014 and January 22, 2015; accepted February 1, 2015. Date of publication February 9, 2015; date of current version March 17, 2015. The preliminary version of this paper [1] has been presented at the 9th International Conference on Network and Service Management (CNSM) as a short paper. The associate editor coordinating the review of this paper and approving it for publication was X. Fu.

F. Zhou is with CERI-LIA (Computer Science Laboratory) at the University of Avignon, France (e-mail: fen.zhou@univ-avignon.fr).

J. Liu is with the Xidian University, China (e-mail: jyliu@xidian.edu.cn).

G. Simon is with the Network Department of Telecom Bretagne, Institut Mines-Telecom, France (e-mail: gwendal.simon@telecom-bretagne.eu).

R. Boutaba is with D. Cheriton School of Computer Science at the University of Waterloo, Canada (e-mail: rboutaba@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSM.2015.2400915

As it has been shown in many previous works [5]–[7], the delivery of a live channel stream over a CDN can benefit from constructing an *overlay*. Edge servers contribute to the delivery of flows within the CDN using spare upload capacities. Several techniques have been developed for the construction of an overlay for one live channel over CDNs. In particular, *multi-tree* overlays that leverage *rateless codes* (e.g., Raptor codes [8]) enable video stream delivery with optimal bandwidth utilization [7], [9], [10]. This solution however performs well for *one* channel while the traffic that a CDN should carry for a platform like twitch.tv consists of a large number of simultaneous channels. Typically, agreements between twitch.tv and CDN stipulate the delivery of around fifty channels—the so-called "featured" channels.

In this paper, we study the problem of delivering multiple channels over a Telco-CDN. Due to the bandwidth constraint on edge servers in a Telco-CDN, not all video channels can be delivered at the same time. Given an importance value for each video channel, our goal is to maximize the sum of the importance of all delivered video channels while minimizing the total bandwidth consumption. This optimization problem involves two distinct subproblems: the construction of one overlay per channel, and the allocation of resources among the competing overlays. To tackle this problem, we explore two approaches:

- **Two-Step optimization**—First, we construct all overlays, then we allocate the bandwidth among the different overlays so that the most important channels are delivered, until resource exhaustion.
- **Joint optimization**—We jointly construct the overlays and reserve the required resources on the fly. This approach leads to optimal solutions.

From a theoretical stand point, the joint optimization approach is better than the two-step one. Here, we want to measure the actual performance gain that can be ultimately achieved. Therefore, we formulate the optimization problem in both approaches using Mixed Integer Linear Programs (MILP). As can be expected, the joint optimization MILP approach outperforms the two-step MILP. However, both approaches introduce so much complexity that they can only be performed on small networks.

From a practical stand point, we are interested in studying both approaches as well. We propose fast *heuristic* algorithms for delivering multiple channels in large-scale Telco-CDN and highlight their main advantages and drawbacks. In particular, the drawback of the joint-based heuristic algorithm is that it requires a complete re-computation of the overlays after any event (e.g., the termination of a video channel delivery). In our simulations, however the joint-based heuristic algorithm outperforms the two-step-based heuristic algorithms by a margin of overall profit ratio that justifies such drawback.

The contribution of this paper is two-fold. First, it provides a significant step into the study of joint optimization problems for the provisioning of multiple overlays. Joint optimization approaches achieve better performance at the cost of more intensive computations. Our study provides a comprehensive evaluation of this trade-off. Second, this paper highlights some of the critical problems underlying the management of Telco-CDN. In particular, scarce Telco-CDN resources need to be well administered. We show in this paper how optimization techniques can help.

The reminder of this paper is organized as follows. We review related work in Section II and present the system model in Section III. Then, we propose the two-step MILP formulation and the joint MILP formulation in Section IV and Section V respectively. Heuristic algorithms for two-step inspired optimization and joint optimization are introduced in Section VI. Simulation results are presented in Section VII. Finally, Section VIII concludes this paper.

II. Related Work

Overlay construction for live video delivery has been extensively studied in the context of peer-to-peer (P2P) streaming. Two classes of approaches are proposed: *mesh-based* overlay approaches and *tree-based* ones. The mesh-based approaches are tailored to solve the inherent peer churn problem in P2P systems, however, in relatively static system environment, such as CDN, they bring more operating overhead than the tree-based ones [11]. The tree based approaches multicast content by organizing peers into multiple delivery trees, thus they are also referred to as the Application Layer Multicast (ALP) scheme [12]. Numerous works have studied multi-tree packing for P2P ALP protocols (see [13], [14] for surveys). In [15], the authors proposed a mechanism to build multiple node-disjoint multicast trees to disseminate Multiple Description Coding (MDC) based content. The streaming capacity of a delivery forest (a bundle of trees) under node degree bound is studied in [16]. The authors devised tree construction algorithms to achieve the near maximum streaming rate for all receivers of a streaming session. But, they cannot guarantee that each peer receive all descriptions of a video. In [17], the authors rely on multiple trees structure to maximally utilize the clients uplink resource to achieve the desirable stream quality. However, none of the aforementioned previous work solved the inherent resource allocation problem among multiple overlays together with the end-to-end delay constraint for each overlay.

The use of rateless coding in content delivery overlays was introduced in [9]. Rateless codes are a class of erasure codes, using which limitless sequence of encoding symbols can be generated for a given set of source symbols. They have the property that the original source symbols can be recovered from any subset of the encoding symbols of size only slightly larger than the number of source symbols [7], [8]. The Raptor implementation of Rateless codes reaches speeds of several gigabits per second [7], [8].

Its key advantages include: (*i*) rateless codes have very low computational cost; (*ii*) they minimize delivery redundancy when a server receives data concurrently from multiple other servers; and (*iii*) they are adaptive to varying channel conditions since the encoder can generate on the fly as many encoded symbols as needed. Following this seminal work, several performance improvements were proposed in the literature. End-to-end delivery delay is reduced by the method described in [18]. Shorter start-up delay and more stable service are among

the objectives addressed in [19]. Previous works consider the delivery in only one overlay. The objective of this paper is different since it aims at constructing *several* rateless coding based overlays while addressing the critical problem of bandwidth reservation for these overlays.

Optimization problems related to the construction of tree overlays have been widely studied in the context of IP multicast [20]–[25]. Researchers have identified two main approaches for building a multicast tree. The first approach consists in building a Steiner tree using the heuristic algorithms given in [20], [21]. The second approach tries to construct a height-bounded cost-optimal tree. For example, the authors of [25] studied the problem of finding a cost-optimal tree for multimedia communications under the constraint of bounded end-to-end delay. A genetic algorithm is proposed to find a tree with near-optimal cost. Optimization of multiple multicast sessions has been more rarely addressed in the literature. In the optimization problem formulated in [22], the objective is to minimize the total cost of all multicast sessions and the maximum delay. Each multicast group is served by only one delivery tree and the bandwidth is constrained on the links between routers rather than on the router itself. A decomposition technique and a branch-and-bound algorithm are also presented. Multicast tree construction problems are also studied in [23], where the objective is to minimize the maximum traffic congestion over all links while affording all multicast sessions. Heuristic algorithms have been proposed for this NP-hard problem [23], [24].

In our Telco-CDN system, we use a more contemporary video delivery technique based on rateless codes [7], [8], where each video channel is encoded into a number of sub-streams greater than a pre-defined parameter (say K). Each sub-stream is delivered on a different tree. Any K sub-streams are enough to decode the video. Our multi-tree based video delivery system is different from previous works in several aspects. To ensure guarantee of QoS, the end-to-end delay should be bounded for each subscribing edge server. Second, as bandwidth is critical for both entrypoints and edge servers, the sum of nodal degree in all delivery trees should be limited due to the limitation of available bandwidth. Our goal is to deliver as many video channels as possible while minimizing the total bandwidth consumption. Third, rateless coding is used to ensure a robust and reliable video transmission. However, the use of rateless codes makes the tree construction more complicated, since each subscribing edge server should be spanned in at least K different trees (i.e., receive K different sub-streams) to reconstruct a video channel. Finally, bandwidth is limited on edge servers rather than in the communication links between edge servers. The combination of these features makes our problem quite different from those solved in the literature. Our model is hence more in line with the latest works related to CDNs (see [10] for example).

III. System Model

The Telco-CDN network is modeled as a connected symmetric digraph $G(V, E)$. See an example in Fig. 1. The set $V(|V| = n)$ contains the *edge servers* (here the access points to the regional areas) and a set of *entrypoints*$S \subseteq V$, which are located near the peering points (the three main French peering IXPs in Fig. 1). A content provider usually delivers its most popular channels to a given ISP through only one peering point. But a Telco-CDN serves *several* content providers, so the set of channels $\mathcal{I}$ comes from several entrypoints. Each entrypoint s is associated with the set of channels $\mathcal{I}(s) \subseteq \mathcal{I}$ that it actually receives. The sets $\mathcal{I}(s)$ for all $s \in S$ form a partition of $\mathcal{I}$. Each edge server v uses its *upload capacity*c_v to assist the entrypoints for video delivery. Each channel $i \in \mathcal{I}$ is associated with two key parameters. First, the *targets* are edge servers that must receive the channel i. This subset of edge servers is denoted $V_i \subseteq V \setminus S$. Second, the *importance* of channel i, denoted by π_i, sets the priority for the delivery of channels. The channel importance is generally related to the popularity of the content in the area. We assume all channels have the same bitrate, and do not consider rate-adaptive video streaming systems like [26] where video bitrates are heterogeneous. But the stream of each channel can be extended to "a bundle of streams", which contains all the "representations" of the same video channel. All representations are sent to the edge servers, and the adaptation mechanism happens between the edge servers and the end-users. The monitoring of the CDN and the algorithms used to set the above parameters are out of the scope of this paper. For major services, such as Netflix, the content provider provides these parameters to the CDN provider.

We leverage previous works on multi-tree (forest) overlay networks based on rateless codes [7], [10]. Given a video chunk with a size of K streams of symbols, infinitive sequences of rateless code symbols can be generated by applying rateless coding. The reception of slightly more than K streams (say $\lceil(1+\epsilon)K\rceil$, where ϵ is very small) of rateless code symbols enables to recover the original video chunk with high probability [7], [8], which provides the required resilience for video transmissions in Telco-CDNs. The Raptor implementation of rateless code can be used in our system, which achieves speeds of gigabits per second and linear-time decoding [8]. In transmissions, a video clip in a playback time may be segmented into many streams (e.g., a stream may contain several UDP packets). To simplify, we express the upload capacity as the number of streams that the server can transmit over a period of time. In each delivery tree, an entrypoint encodes video clips into rateless code symbols and sends them along the tree. Subscribing edge servers perform rateless decoding to obtain the original video clips. The system introduced in [10] suggests to build, for each channel i, a forest overlay F_i where each overlay tree supports one encoded video stream. An edge server can re-build the stream if it is spanned in at least $\lceil(1+\epsilon)K\rceil$ trees. To both maximize the amount of *distinct* video streams and minimize the entrypoint throughput, the entrypoint s has one and only one direct child in each tree in F_i when $i \in \mathcal{I}(s)$. That is, the delivery of channel i requires the entrypoint to build a number of trees ranging from $\lceil(1+\epsilon)K\rceil$ (if each tree can relay a different video stream to all edge servers) to $\lceil(1+\epsilon)K\rceil \times |V_i|$ (if one tree is needed for delivering a video stream to each edge server).

Multi-Channel Video Delivery Problem: The objective is twofold: to deliver the maximum number of channels (with regard to their importance) and to reduce the traffic load on

the CDN infrastructure, i.e., to reduce the number of edge servers involved in the delivery of a channel if they are not a target for this channel. The former objective prevails over the latter one. Therefore we formulate the optimization problem as first to deliver the maximum number of channels, then to find the delivery scheme that imposes the least traffic load on the network. Let R_i be a binary variable, which equals one if the channel i is delivered to all edge servers in V_i, and zero otherwise. Our first objective is to maximize the overall importance of the delivered video channels, i.e., $\sum_{i\in\mathcal{I}} R_i \times \pi_i$. Our second objective is to minimize the traffic load on the CDN, which corresponds to the number of overlay links used to deliver the channels to the edge servers. In our rateless code based multiple-channel video streaming system, three main constraints are considered:

- **Edge server capacity constraint**. The contribution of edge servers to stream delivery should not surpass their capacities. Let $deg_T^+(v)$ be the out degree of edge server v in a tree T. This out degree corresponds to the reserved upload capacity of edge server v (i.e., the number of streams that v can forward at a time) for the delivery tree T. We denote the multi-tree overlay constructed for channel i by F_i. Clearly, the sum of v's out-degree in the trees of all the overlays it is involved in cannot be greater than its upload capacity:

$$\sum_{i\in\mathcal{I}} \sum_{T\in F_i} deg_T^+(v) \leq c_v,\ \forall v \in V \tag{1}$$

- **Video content constraint related to rateless codes**. Each edge server subscribing to channel i should be spanned in at least $\lceil(1+\epsilon)K\rceil$ trees in the forest F_i dedicated to channel i, so that the edge server is able to receive $\lceil(1+\epsilon)K\rceil$ different streams and decode the video. Let $F_i(v)$ be the set of trees in which the edge server v is included in F_i, and $\hat{K} = \lceil(1+\epsilon)K\rceil$ We have

$$\forall i \in \mathcal{I},\ \forall v \in V_i, |F_i(v)| \geq \hat{K} \tag{2}$$

- **Delay constraint**. This constraint is used to bound the delay from the entrypoint to each subscribing edge server for the guarantee of Quality of Service (QoS). For each tree $T \in F_i$, total transmission delay from the entrypoint s_i to any node of T should be smaller than a given threshold H. Let $SP_T(v)$ be the path (list of arcs) from s_i to v in tree T, and d_e be the delay of an arc e in this path, we have

$$\forall i \in \mathcal{I},\ \forall T \in F_i,\ \forall v \in T,\ \sum_{e\in SP_T(v)} d_e \leq H \tag{3}$$

To illustrate a forest overlay, we give in Fig. 2 an example of video delivery in the French Telco-CDN (Fig. 1). We suppose that all edge servers have an upload capacity of 2 while an entrypoint has a capacity of 5. All links have an identical delay of one, and the end-to-end delay should be bounded by $H = 2$ to guarantee the QoS. For instance, a video channel generated by the entrypoint s_1 should be delivered to edge servers 10 and 12. Let us suppose that $\hat{K} = 2$ streams of symbols are required for decoding the video clip, then three trees with a depth of two can be used, which are illustrated in Fig. 2. We can see that the entrypoint has only one direct child in each tree, and both the delay constraint and the capacity constraint are satisfied.

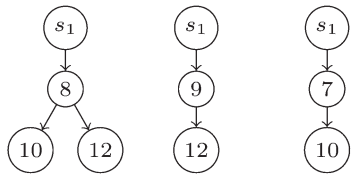


Fig. 2. Forest overlay for video delivery: source s_1 and subscribing edge servers 10 and 12, delay constraint $H = 2$, and rateless coding parameter $\hat{K} = 2$.

TABLE I
NOTATIONS

$G(V, E)$	The graph representing the CDN network
n	$= \|V\|$, the number of edge servers in graph $G(V, E)$
$i \in \mathcal{I}$	A video channel and the set of video channels
S	The set of entrypoints
s_i	The entrypoint delivering the channel i
$\mathcal{I}(s)$	The set of channels delivered by entrypoint s
K	The number of symbol streams a video clip has in a playback time
ϵ	a very small number
$\hat{K}$	$\lceil(1+\epsilon)K\rceil$, Minimum number of symbol streams an edge server should receive to decode a video clip
W_i	$\hat{K} \times \|V_i\|$, the maximum number of trees required for channel i
G^{ik}	A copie of graph G on which we construct a video delivery tree for channel i
T^{ik}	The video delivery tree constructed on G^{ik}
V_i	The subset of edge servers subscribing to channel i
π_i	The importance of channel i
H	The bound on the end-to-end delay to guarantee the quality of service
$N^+(v), N^-(v)$	Set of nodes with an arc from/to v in G
c_v	Upload capacity of an edge server v
$deg_T^+(v)$	The out degree of edge server v in the tree T
M	A big constant number, $\geq H + \max_{e\in E} d_e$
V_s	For any $s \in S$, $V_s = V \setminus \{s\}$
F_i	The resulting forest for the video channel i, *i.e.*, $F_i = \{T^{i1}, \cdots, T^{iW_i}\}$
$F_i(v)$	The set of trees in F_i where edge server v is involved
c_v^i	The capacity of edge server v reserved for F_i
$SP_T(v)$	The path (list of arcs) from the source to v in tree T
Cap	The load on the CDN infrastructure
$Deli$	The quality of the video delivery proportional to the overall importance

To solve this optimization problem, we propose and compare in the following two different MILP models: a two-step optimization MILP and a joint optimization MILP.

IV. Two-Step Optimization MILP

We first describe a two-step approach for solving the multi-channel video delivery problem. In this case, the original problem is divided into two subproblems: first we construct a forest overlay for each channel under both the delay and the capacity constraints, then assign the resources of CDN nodes to the different overlays. System notations are given in Table I and MILP variables are defined in Table II. We name this two-step optimization MILP as *SOP-ILP*.

TABLE II
MILP VARIABLES

$L_{uv}^{ik} \in \{0,1\}$	Binary variable. Is 1 if arc (u,v) is used in T^{ik}, 0 otherwise.
$D_v^{ik} \in [0,\infty)$	Real variable. The delay from entrypoint of video channel i to v in tree T^{ik} (if $v \in T^{ik}$)
$R_i \in \{0,1\}$	Binary variable. Equals 1 if video channel i can be successfully delivered, 0 otherwise.

A. Delay and Capacity Bounded Forest Overlay Construction

For any channel i, its video entrypoint s_i has to build up to $W_i = \hat{K} \times |V_i|$ delivery trees. The main idea of this MILP is to decompose the network graph G into a set of instances denoted by G^{ik} with $i \in \mathcal{I}$ and $k \in \{1,\ldots,W_i\}$. Each instance is the support for a tree T^{ik} rooted at the entrypoint of channel i. A tree T^{ik} may be null with zero arcs while the non-null trees form the forest F_i dedicated to the delivery of channel i, i.e., $F_i = \{T^{i1},\cdots,T^{iW_i}\}$. For any arc $(u,v) \in E$, let $L_{uv}^{ik} \in \{0,1\}$ equal 1 if (u,v) is used in T^{ik}, and 0 otherwise.

For the sake of readability, we omit in the MILP formulation below the use of set membership indication $\in$ when it stands for the standard whole set. In other words, we write $\forall i$, $\forall s$, $\forall u$, $\forall v \notin V_i$, and $\forall k$ to imply $\forall i \in \mathcal{I}$, $\forall s \in S$, $\forall u \in V$, $\forall v \in V \setminus V_i$, and $\forall k \in \{1,\ldots,W_i\}$, respectively. We also use $N^+(v)/N^-(v)$ to denote the set of nodes with an arc from/to v in G. It is worth noting that only L_{uv}^{ik}, R_i and D_v^{ik} (introduced later) are MILP variables while the rest are network input parameters.

Our goal here is to build the forest overlay F_i under both the delay constraint and the node capacity constraint for the delivery of *each* given channel $i \in \mathcal{I}$. We suppose that the CDN has enough capacity to serve any single channel $i \in \mathcal{I}$. We aim at constructing the leanest overlay, i.e., the overlay that minimizes the total used capacity while satisfies both the delay constraint and the edge server capacity constraint. Thus, we have to solve $|\mathcal{I}|$ forest overlay optimization problems in this part. For each video channel $i \in \mathcal{I}$, the objective function is thus defined as:

$$\text{minimize} \sum_{v\in V} \sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^+(v)} L_{vu}^{ik},\ \forall i \quad \text{(SOP-OC)} \tag{4}$$

subject to constraints (5)–(14).

$$\sum_{v\in N^-(s)} L_{vs}^{ik} = 0 \qquad \forall s,\ \forall i \in \mathcal{I}(s),\ \forall k \tag{5}$$

$$\sum_{v\in N^+(s)} L_{sv}^{ik} \le 1, \qquad \forall s,\ \forall i \in \mathcal{I}(s),\ \forall k \tag{6}$$

$$\sum_{u\in N^-(v)} L_{uv}^{ik} \le 1, \qquad \forall i,\ \forall k,\ \forall v \notin \{s_i\} \tag{7}$$

$$\sum_{u\in N^+(v)} L_{vu}^{ik} \le \sum_{u\in N^-(v)} L_{uv}^{ik} \times c_v, \qquad \forall i,\ \forall k,\ \forall v \notin \{s_i\} \tag{8}$$

$$\sum_{u\in N^+(v)} L_{vu}^{ik} \ge \sum_{u\in N^-(v)} L_{uv}^{ik}, \qquad \forall i,\ \forall k,\ \forall v \notin V_i \tag{9}$$

$$\sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^-(v)} L_{uv}^{ik} \ge \hat{K}, \qquad \forall i,\ \forall v \in V_i \tag{10}$$

$$\sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^+(v)} L_{vu}^{ik} \le c_v, \qquad \forall i,\ \forall v \tag{11}$$

Constraint (5) ensures that the entrypoint of channel i is not involved in channel delivery with another role than being the root node for each delivery tree of channel i. Constraint (6) makes sure that an entrypoint has *at most* one direct child in a delivery tree when a channel is delivered. This constraint limits an entrypoint source to use only one unit upload capacity in each delivery tree. Constraint (7) guarantees that an edge server has at most one input arc in each tree of a channel. Constraint (8) makes sure that no node in V can be the root of a tree in F_i except the entrypoint s_i. This constraint permits a relay edge server to connect as many as c_v other edge servers in a delivery tree. Thus, we will obtain a delivery tree instead of a path for one stream, although the entrypoint source has only one direct child. Constraint (9) guarantees that the leaf nodes in a channel i are in V_i (edge servers belonging to unneeded channels are only data relays). Constraint (10) ensures that each node $v \in V_i$ should be spanned at least $\hat{K}$ times in F_i. Constraint (11) imposes a capacity limitation on both entrypoints and nodes. It indicates that an edge server $v \in V$ is able to use as much as c_v capacity for the overlay of channel i. It is important to understand that the construction of this overlay is independent of that of the other overlays.

We also have to guarantee the end-to-end delay and ensure that there is no cycles in trees. We define a new variable $D_v^{ik} \in [0,\infty]$, which stands for the delay from the entrypoint s_i to an edge server v in T^{ik}. The additional constraints are:

$$D_s^{ik} = 0, \qquad \forall s,\ \forall i \in \mathcal{I}(s),\ \forall k \tag{12}$$

$$D_v^{ik} \le H, \qquad \forall i,\ \forall k,\ \forall v \notin \{s_i\} \tag{13}$$

$$D_v^{ik} - D_u^{ik} \ge d_{uv} - M\left(1 - L_{uv}^{ik}\right),\ \forall i,\ \forall k,\ \forall v,\ \forall u \in N^-(v) \tag{14}$$

Constraints (12) set the delay of an entrypoint s as zero in the delivery tree originated from s and constraints (13) bound the delay of each edge server. Constraints (14) make sure that no cycle exists in F_i. Together with constraints (7) and (8), they guarantee the tree structure in each instance G^{ik} if it exists.

By solving the above $|\mathcal{I}|$ optimization problems, we will obtain a set of forest overlays $\{F_1, F_2, \cdots, F_{|\mathcal{I}|}\}$ one for each video channel. The formulated SOP-OC problem is a minimum spanning tree packing problem, which tries to minimize the total cost of the spanning forest while satisfying all the constraints. Suppose that each link has a cost of 1 and that only one stream is enough ($\hat{K} = 1$). Then the SOP-OC problem becomes a Steiner tree problem, which is NP-hard [20].

B. Bandwidth Allocation

The first step results in $|\mathcal{I}|$ forest overlays. Now we allocate resources with respect to the requirements of each overlay. A forest F_i can be either null (zero arc), or a forest that spans all peers of V_i in at least $\hat{K}$ trees. When a forest F_i is null, no capacity should be allocated. In other words, the channel with null F_i cannot be transmitted. On the contrary, a non-null forest F_i may be used for the delivery of channel i, but it requires that

the engaged resources are available. Let c_v^i be the capacity that should be reserved by an edge server $v \in V$ for the forest F_i:

$$c_v^i = \sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^+(v)} L_{vu}^{ik} \qquad \forall i \in \mathcal{I},\ \forall v \tag{15}$$

The sum of all capacities c_v^i over all channels $i \in \mathcal{I}$ can be greater than c_v. It means that not all the computed forests can be delivered. For any channel $i \in \mathcal{I}$, we define a decision variable $R_i \in \{0, 1\}$. Now we need to decide whether a channel i should be *accepted* ($R_i = 1$ and all the engaged capacities in F_i should be secured to deliver the video channel i) or *rejected*($R_i = 0$). The only constraint in this step is that, for each edge server, the sum of its used capacity in all accepted overlays should not be above its capacity. Our objective is to find the set of accepted channels so that the overall profit $\pi_i \times R_i$ is maximized:

$$\text{maximize} \sum_{i\in\mathcal{I}} R_i \times \pi_i \quad \text{(SOP-BA)} \tag{16}$$

subject to the following constraint:

$$\sum_{i\in\mathcal{I}} c_v^i \times R_i \le c_v, \qquad \forall v \tag{17}$$

In this MILP model, c_v^i is regarded as the input while R_i is the variable. There are $|\mathcal{I}|$ variables and $|V|$ constraints. Only the accepted video channels ($R_i = 1$) can be delivered by using the forest overlay F_i constructed in SOP-OC. SOA-BA is a multi-Dimensional Knapsack Problem (DKP), which is strongly NP-hard for any $|\mathcal{I}| \ge 2$ [27]. In addition, the DKP problem does not admit any fully-polynomial time approximate scheme [28]. Thus greedy-like fast heuristic algorithms are needed.

In the two-step approach, we treat the overlay construction and bandwidth separately. Although we find the optimal solution for each subproblem, we may not guarantee that the final solution is globally optimal, since these two subproblems interact with each other. This is why we will introduce a joint optimization MILP in the next section.

V. Joint Optimization MILP

Different from the previous two-step approach, here we formulate the multi-channel video delivery problem as a joint optimization MILP, which enables to solve the overlay construction and bandwidth allocation simultaneously. We name joint optimization MILP as *JOP-ILP*. Then, we compare these two different MILP formulations.

A. Joint Optimization Formulation

The joint MILP formulation allows the computation of a globally optimal solution in the Telco-CDN by solving the overlay computation and bandwidth allocation at the same time. Our goal is to maximize the sum of the importance of all delivered video channels while respecting both the nodal capacity and the end-to-end delay constraints. Once guaranteeing this main objective, we secondly want to reduce the traffic load on the CDN infrastructure. The load on the CDN infrastructure, namely Cap is expressed as follows:

$$Cap = \sum_{i\in\mathcal{I}} \sum_{v\in V} \sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^+(v)} L_{vu}^{ik} \tag{18}$$

The quality of the delivery, which we call $Deli$, is defined as:

$$Deli = n\hat{K} \times \sum_{i\in\mathcal{I}} |V_i| \times \sum_{i\in\mathcal{I}} (R_i \times \pi_i) \tag{19}$$

The only component that matters is $\sum_{i\in\mathcal{I}} R_i \times \pi_i$. All other components are constants introduced to make the $Deli$ objective prevail over Cap in the general objective function, which is thus expressed as:

$$\text{maximize } Deli - Cap \quad \text{(JOP)} \tag{20}$$

Since $n\hat{K} \times \sum_{i\in\mathcal{I}} |V_i|$ is bigger than Cap and π_i is an integer, we can see that the $Deli$ dominates the objective function. Thus, our main objective is to maximize the quality of the delivery instead of finding a tradeoff between delivery quality and the traffic load. Nevertheless, we believe that finding such tradeoff can be very valuable and is left for future research.

Most part of the *JOP-ILP* formulation is exactly the same as that of the SOP-OC ILP formulation (i.e., constraints (5)–(14)). However, we construct the overlay, decide the delivery of a video channel, and allocate the ressource at the same time in the *JOP-ILP* model. Thus, we should have three kinds of decision variables L_{uv}^{ik}, R_i and D_v^{ik}. Constraints (6), (7), (10) and (11) should also be modified accordingly as follows:

$$\sum_{v\in N^+(s)} L_{sv}^{ik} \le R_i, \qquad \forall s,\ \forall i \in \mathcal{I}(s),\ \forall k \tag{21}$$

$$\sum_{u\in N^-(v)} L_{uv}^{ik} \le R_i, \qquad \forall i,\ \forall k,\ \forall v \notin \{s_i\} \tag{22}$$

$$\sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^-(v)} L_{uv}^{ik} \ge R_i \times \hat{K}, \quad \forall i,\ \forall v \in V_i \tag{23}$$

$$\sum_{i\in\mathcal{I}} \sum_{k\in\{1,\ldots,W_i\}} \sum_{u\in N^+(v)} L_{vu}^{ik} \le c_v, \qquad \forall v \tag{24}$$

Constraint (21) makes sure that an entrypoint has *at most* one child in a tree when a channel i is delivered and no child at all when channel i is not delivered. For each channel i, constraint (22) guarantees that an edge server has at most one input arc in each tree if channel i is accepted, otherwise it has zero incoming arc. Constraint (23) ensures that each node $v \in V_i$ is spanned at least $\hat{K}$ times in F_i if channel i is delivered. Constraint (24) imposes that the bandwidth of node v (edge servers and entry-points) used by all channels should not be bigger than its capacity.

Thus, the *JOP-ILP* formulation is subject to constraints (5), (21), (22), (8), (9), (23), (24) and (12)–(14). This method will produce a set of forest overlays $\{F_1, F_2, \cdots, F_{|\mathcal{I}|}\}$ for all video channels at the same time.

B. Comparison With the Two-Step Optimization MILP

We compare the complexity of the proposed joint optimization MILP (*JOP-ILP*) with that of the two-step MILP ($SOP - ILP =$ SOP-OC + SOP-BA) in Table III. We can see that both MILP formulations have exactly the same number of variables, but the latter one requires $|\mathcal{I}| \times |V|$ more constraints. Instead

TABLE III
COMPARISON OF VARIABLES AND CONSTRAINTS IN THE PROPOSED TWO MILP FORMULATIONS

Variables	Number of Variables	Constraints	Number of Constraints
Two-step Optimization (SOP-ILP= SOP-OC + SOP-BA)			
L_{vu}^{ik} D_v^{ik} R_i	$\hat{K} \times \lvert E\rvert \times \sum_{i\in\mathcal{I}} \lvert V_i\rvert$ $\hat{K} \times \lvert V\rvert \times \sum_{i\in\mathcal{I}} \lvert V_i\rvert$ $\lvert\mathcal{I}\rvert$	constraints (5)-(14)	$(1+\hat{K}\times(\lvert E\rvert+4\lvert V\rvert))\times\sum_{i\in\mathcal{I}}\lvert V_i\rvert+(1+\lvert\mathcal{I}\rvert)\times\lvert V\rvert-\hat{K}\times\sum_{i\in\mathcal{I}}\lvert V_i\rvert^2$
Joint Optimization (JOP-ILP)			
L_{vu}^{ik} D_v^{ik} R_i	$\hat{K} \times \lvert E\rvert \times \sum_{i\in\mathcal{I}} \lvert V_i\rvert$ $\hat{K} \times \lvert V\rvert \times \sum_{i\in\mathcal{I}} \lvert V_i\rvert$ $\lvert\mathcal{I}\rvert$	constraints (5), (21), (22) (8), (9), (23), (24) (12)-(14)	$(1+\hat{K}\times(\lvert E\rvert+4\lvert V\rvert))\times\sum_{i\in\mathcal{I}}\lvert V_i\rvert+\lvert V\rvert-\hat{K}\times\sum_{i\in\mathcal{I}}\lvert V_i\rvert^2$

of solving the multi-channel video delivery problem entirely at once, the two-step MILP approach divides the problem into $|\mathcal{I}|$ separate overlay optimization subproblems plus one bandwidth allocation subproblem, which are also NP-hard. Thus, the gain in time complexity is difficult to estimate, while overall system performance will obviously degrade. Further performance comparisons are given in Section VII.

VI. HEURISTIC ALGORITHMS

Since both JOP and SOP-OC+SOP-BA MILP formulations are computationally expensive, they do not scale with the number of channels and network size in practical Telco-CDNs. Thus, we propose three fast heuristic algorithms for solving the delivery of multiple video channels. Both heuristic algorithms are greedy algorithms, with one main loop. We assume that the delay is the same for all links in the network, which we call one unit delay, e.g., 5 ms. Before explaining these two algorithms, we next introduce a min-cost based forest overlay construction algorithm, which can be used for both JOP and SOP heuristic algorithms.

A. Capacity-and-Delay-Bounded Min-Cost Forest Overlay Construction Algorithm

The difficulty of our forest overlay construction problem is that both the delay and the node capacity are bounded in each tree of the overlay, which makes our problem much more complicated than the degree-bounded spanning tree problem or the delay-bounded Steiner tree problem [25]. Thus existing algorithms in the literature cannot be applied for computing the required forest overlay. To this end, we propose a Capacity-and-Delay-Bounded Min-Cost Forest Overlay construction algorithm, whose pseudo code is given in Algorithm 1. Given a video channel $i \in \mathcal{I}$, its subscribing edge servers V_i and rateless coding parameter $\hat{K}$, and the bandwidth of each edge server, Algorithm 1 computes the video delivery tree one by one iteratively until all subscribing edge servers are spanned in at least $\hat{K}$ trees so that they are able to recover the video trunk. Since one objective is to minimize the traffic load on the CDN infrastructure, the basic idea of our algorithm is to add each edge server to the video delivery tree with the minimum cost. For the sake of explaining the pseudo code of Algorithm 1, some definitions are provided as follows. We define V_k and E_k as the node set and the edge set of tree T_k respectively. We note $h_{T_k}(v)$ as the depth of an edge server v in the delivery tree T_k, which is the distance from the tree root s_i to v. We use V_k^+ to denote connection-useful edge servers in T_k, whose capacity is not null and depth is smaller than H. These connection-useful edge servers can be used to connect a new edge server to tree T_k. An auxiliary graph G_c is introduced, in which $SP_{G_c}(a,b)$ is the shortest path between edge server a and b, and $dist_{G_c}(a,b)$ is its length. We note V_K as the edge servers not yet spanned $\hat{K}$ times, V_c as the edge servers still with capacity, and V_{Kc} as the subscribing edge servers still with capacity or not yet spanned in T_k, and V_H as the edge servers with a depth of H in tree T_k. The edge servers in $\overline{V_c \cup V_{Kc}}$ and V_H are not able to connect a new edge server to the delivery tree due to the out usage of capacity or the depth constraint. Thus, they are removed from the orignal graph to generate an auxiliary graph G_c.

Algorithm 1: Capacity-and-Delay-Bounded Min-Cost Forest Overlay

Input : $G(V,E)$, $i \in \mathcal{I}$, V_i, $\{c_v : v \in V\}$, $\hat{K}$, and H
Output: R_i, F_i, and residual capacity $\{c'_v : v \in V\}$

$k \leftarrow 1$; $V_K \leftarrow V_i$;
$V_c \leftarrow \{v : v \in V_i,\ c'_v = c_v > 0\}$;
while $c'_{s_i} > 0$ **and** $V_c \neq \emptyset$ **and** $V_K \neq \emptyset$ **do**
  $V_k \leftarrow \{s_i\}$; $V_k^+ \leftarrow \{s_i\}$;
  $T_k \leftarrow \{V_k, \emptyset\}$;
  $G_c \leftarrow G - \overline{V_c \cup V_K}$;
  while $\overline{V_k} \cap V_K \neq \emptyset$ **and** $V_k^+ \neq \emptyset$ **do**
    find the nearest edge server $a \in \overline{V_k} \cap V_K$ and the connector edge server $b \in V_k^+$ **s.t.** $dist_{G_c}(a,b) = \min\{dist_{G_c}(v,u) : v \in \overline{V_k} \cap V_K, u \in V_k^+, dist_{G_c}(v,u) + h_{T_k}(u) \leq H\}$;
    if $a \neq NULL$ **then**
      $V_k \leftarrow V_k \cup \{v : v \in SP_{G_c}(a,b)\}$;
      $E_k \leftarrow E_k \cup \{e : e \in SP_{G_c}(a,b)\}$;
      $T_k \leftarrow \{V_k, E_k\}$;
      for $v \in \{v : v \in SP_{G_c}(a,b), v \neq a\}$ **do**
        $c'_v \leftarrow c'_v - 1$;
      $V_c \leftarrow \{v : v \in V_i,\ c'_v > 0\}$
      $V_k^+ \leftarrow \{v : v \in V_k \cap V_c, v \neq s_i, h_{T_k}(v) < H\}$;
      if *a is spanned $\hat{K}$ times* **then**
        $V_K \leftarrow V_K - \{a\}$;
      $V_{Kc} \leftarrow V_K - \{v : v \in V_K \cap V_k, c'_v = 0\}$;
      $V_H \leftarrow \{v : v \in V_k \cap V_c, h_{T_k}(v) = H\}$;
      $G_c \leftarrow G - \overline{V_c \cup V_{Kc}} - \{s_i\} - V_H$;
    else no edge server found
      if $b = s_i$ **then**
        Not all subscribing edge servers can receive the video and goto step (30);
      else
        break;
  $k \leftarrow k + 1$;
if $V_K = \emptyset$ **then**
  $R_i \leftarrow 1$; $F_i \leftarrow \cup_{i\in k} T_k$;
else
  $R_i \leftarrow 0$; $F_i \leftarrow \emptyset$;

For each spanning tree T_k, we begin with the entrypoint s_i of a video channel i. Then, we should find the first nearest edge server in V_K to connect to s_i. The entrypoint should have only one outgoing branch as it delivers only one stream of symbols in each tree T_k. For computing the nearest edge server to s_i without violating the capacity of edge servers, we construct an auxiliary graph G_c by removing from the original graph the incapacitated edge servers except the subscribing edge servers, i.e. $\overline{V_c \cup V_K}$. In the auxiliary graph G_c, the nearest edge server r to s_i is added to T_k using the shortest path in $SP_{G_c}(r, s_i)$ under the condition that the distance of this shortest path is no bigger than H. If this distance is bigger than H, it means that no edge server can be added to the kth delivery tree. Consequently, not all subscribing edge servers can receive $\hat{K}$ streams, no forest overlay can be found for this video channel and Algorithm 1 terminates. If the first edge sever satisfying the depth constraint can be found, then we will continue the span of the delivery tree T_k. We update the connection-useful edge server set V_k^+ by adding capacitated edge servers in T_k with a depth smaller than H. Any edge server in V_k^+ can be used to connect an edge server in V_K to the tree once the tree height constraint is satisfied. Now, what we want to find is the nearest edge server to the actual delivery tree T_k without violating the tree height constraint. To this end, the auxiliary graph should be generated a little bit differently from the first step. Here, we remove connection-useless edge servers from the orignal graph to obtain G_c, which are incapacitated edge servers except the subscribing edge servers not yet spanned in T_k, the edge servers with depth H and the entrypoint s_i, i.e. $\overline{V_c \cup V_{Kc}} + \{s_i\} + V_H$. Then, Dijkstra's algorithm is used to compute the shortest path from each edge server $v \in \overline{V_k} \cap V_K$ to each edge server $u \in V_k^+$. Among them, the shortest one satisfying the height constraint will be added to the delivery tree, i.e, $dist_{G_c}(v, u) + h_{T_k}(u) \leq H$. The same procedure will be repeated to span tree T_k until no satisfying edge server can be found or all edge servers are spanned. If there are still some edge servers not yet spanned $\hat{K}$ times (V_K is not empty), we start a new delivery tree T_{k+1} using the same technique.

Algorithm 1 returns either a forest overlay F_i for delivering video channel i or null if the video channel cannot be delivered.

B. Two-Step-Inspired Heuristic Algorithms

The *SOP-heu* algorithm is inspired by the two-step optimization approach where all overlays are computed before the resources are allocated. By applying Algorithm 1 for each video channel, $|\mathcal{I}|$ forest overlays are constructed beforehand. What we should do next is to find a bandwidth allocation strategy to maximize the total importance of transmitted videos. The bandwidth allocation problem is in essence a 0-1 multidimensional knapsack problem. Thus, the Greedy-Like heuristic algorithms in [29] can be applied directly to solve this problem.

For ease of expression, we introduce channel sets $\mathcal{I}_P$, $\mathcal{I}_A$ and bandwidth vectors B, C. Set $\mathcal{I}_P$ is the set of video channels to be processed, while $\mathcal{I}_A$ is the set of video channels accepted for delivery. The bandwidth of edge server v consumed for delivering accepted videos in $\mathcal{I}_A$ is denoted by B_v, i.e., $B_v = \sum_{i \in \mathcal{I}_A} c_v^i$, and the bandwidth consumption vector consists of the bandwidth consumption of all edge servers $B = \{B_{v_1}, \ldots, B_{v_n}\}$. If $B_v \leq c_v$, there is no need for considering the bandwidth constraint of edge server v. C is the vector of initially available bandwidth of all edge servers, i.e., $C = \{c_{v_1}, \ldots, c_{v_n}\}$. The bandwidth required by a video channel i is represented by vector $B^i = \{c^i_{v_1}, c^i_{v_2}, \ldots, c^i_{v_n}\}$. Here we emphasize: (*i*) all the overlays are pre-computed using Algorithm 1 presented in the previous subsection; and (*ii*) the video channels are selected iteratively according to a *utility score*$\mathcal{U}$, which is defined as

$$\mathcal{U} = \max_{i \in \mathcal{I}} \mathcal{U}_i = \max_{i \in \mathcal{I}} \frac{\pi_i}{\mathcal{P}_i} \tag{25}$$

where $\mathcal{P}_i$ represents the *penalty factor* of the video channel i. $\mathcal{P}_i$ may be defined in several different ways which will then define different selection criteria, such as the channel importance, or the required bandwidth for a channel. Once the selected video channel satisfies the bandwidth constraint of all edge servers, it will be accepted for delivery, otherwise it will be dropped. We repeat this procedure until all video channels have been processed. This algorithm will return the forest overlays for all accepted video channels. Pseudocode is given in Algorithm 2.

Algorithm 2: Two-Step-Inspired Heuristic Main Loop

```
Input  : I, {(π_i, V_i) : ∀i ∈ I}, G, {c_v : ∀v ∈ V}
Output: Accepted overlay forest F_i for all i ∈ I
precompute F_i for all i ∈ I using Algorithm 1 and get
  the bandwidth vector B^i required by channel i;
I_P ← I;
I_A ← ∅;
B ← {0, ..., 0}^n;
while I_P ≠ ∅ do
    find ī ∈ I, s.t. U_ī = max_{i∈I_P} π_i/P_i;
    if B^ī + B ≤ C then
        I_A ← I_A ∪ {ī};
        B ← B + B^ī;
        add forest overlay F_ī to the solution;
    I_P ← I_P − {ī};
```

The key point of Algorithm 2 is the definition of the utility score $\mathcal{U}$. Here, we define them in the following two ways by varying the penalty factor $\mathcal{P}_i$.

- ***SOP1-heu***. The simplest implementation of the utility score is to set $\mathcal{U}(i) = \pi_i, \ \forall i \in \mathcal{I}$, i.e., the penalty factor of each video is equal to $\mathcal{P}_i = 1$ and the channels are processed according to their importance.
- ***SOP2-heu***. It is however well-known from knapsack and bin packing literature [28] that better performance can be obtained using utility scores that take into account both the gain (here π_i) and the *penalty cost*$\mathcal{P}_i$ that this overlay produces on the infrastructure. In our implementation, we utilize the following parameters to set the penalty cost for each video channel $i \in \mathcal{I}$: (*i*) *required bandwidth*: If two candidate video channels of $\mathcal{I}_P$ have equal importance, it is more advantageous to select the one consuming less bandwidth. Video channel i consumes bandwidth vector B^i, and hence the penalty factor $\mathcal{P}_i$ should be large if the norm of B^i is large. However, this remark should be refined by considering each bandwidth ressource separately. Thus, we use $B_v + c_v^i$ to represent this parameter,

which is the bandwidth consumption of edge server v if video channel i is accepted for delivery. (*ii*) *remaining bandwidth*: We try to prevent edge servers from capacity exhaustion, with the risk of network disconnection. Indeed, the bandwidth consumption vector B is used out by the currently accepted video channels, thus bandwidth vector $C - B$ remains available for the video channels to be processed in $\mathcal{I}_P$. The minimum element of $C - B$ designates the currently scarcest bandwidth of all edge servers, which should then be spared as much as possible. This factor can be represented by $c_v - (B_v + c_v^i)$. (*iii*) *saving on critical edge servers*: We suppose channel i is accepted for delivery. Then, we analyse the channels that have not been processed yet, and estimate the edge servers that are the most demanded. If the future bandwidth demand is large, we should also tend to avoid selecting a video channel requiring too much bandwidth of v. This parameter can be represented by $\sum_{\bar{i}\in\mathcal{I}_P} c_v^{\bar{i}} - c_v^i$. From these remarks, $\mathcal{P}_i$ should be governed by the worst effect that the video channel i would have on the following quantities of bandwidth of edge server v:

$$\mathcal{P}_i = \max_{v\in V} \frac{\left(B_v + c_v^i\right)\left(\sum_{\bar{i}\in\mathcal{I}_P} c_v^{\bar{i}} - c_v^i\right)}{\left(c_v - B_v - c_v^i\right)} \tag{26}$$

The two-step inspired heuristic algorithms have the following advantages and drawbacks.

1) **Advantages**: The overlay construction and bandwidth allocation are two completely independent procedures, one will not influence another. In case of churn of video channels, only the bandwidth allocation part should be recomputed, while the overly forest computed beforehand can still be used. Thus, computation time can be saved. Moreover, the most costly channels are not processed first if they are not very important. This strategy is expected to allow the rapid distribution of video channels.
2) **Drawbacks**: For each video channel, Algorithm 2 uses the original bandwidth vector C to pre-compute a forest overlay without the knowledge of bandwidth consumption of other channels. If all of these forests use too much bandwidth of critical edge servers, only few of them can be accepted for delivery. Therefore it may be unable to utilize the last remaining resources at the end of the loop. However, if we know the remaining bandwidth of critical edge servers before computing the forest overlay, we can find alternative paths by avoiding passing through those critical edge servers.

C. Joint-Inspired Heuristic Algorithm

The *JOP-heu* algorithm is inspired by the joint optimization approach. Pseudocode is given in Algorithm 3. In this method, overlay computation and bandwidth allocation are conducted at the same time. Higher priority is given for allocating bandwidth to more important video channels, and the overlay computation takes into account the remaining capacities of edge servers. This algorithm works as follows. At each step, the video channel $\bar{i}$ with the biggest importance is selected for processing (as emphasized in line 5 of Algorithm 3). Before computing a forest overlay for video channel $\bar{i}$, the remaining bandwidth of each edge server is already known and they are used as input parameters for the forest computation (refer to line 6). Once a valid forest $F_{\bar{i}}$ is found, the bandwidth of each edge server is updated by removing the used bandwidth (refer to line 9).

Algorithm 3: Joint-Inspired Heuristic Main Loop

Input : $\mathcal{I}, \{(\pi_i, V_i) : \forall i \in \mathcal{I}\}, G, \{c_v : \forall v \in V\}$
Output: Accepted forest overlay F_i for all $i \in \mathcal{I}$

1 $\mathcal{I}_P \leftarrow \mathcal{I}$;
2 $\mathcal{I}_A \leftarrow \emptyset$;
3 $C \leftarrow \{c_{v_1}, ..., c_{v_n}\}^n$;
4 **while** $\mathcal{I}_P \neq \emptyset$ **do**
5 find $\bar{i} \in \mathcal{I}$, **s.t.** $\mathcal{U}_{\bar{i}} = \max_{i\in\mathcal{I}_P} \pi_i$;
6 Use currently available bandwidth vector C as input parameters and apply Algorithm 1 to compute the forest overlay for video channel $\bar{i}$ and get the bandwidth vector $B^{\bar{i}}$ required by channel $\bar{i}$;
7 **if** $R_{\bar{i}} == 1$ **then**
8 $\mathcal{I}_A \leftarrow \mathcal{I}_A \cup \{\bar{i}\}$;
9 $C \leftarrow C - B^{\bar{i}}$;
10 add forest overlay $F_{\bar{i}}$ to the solution;
11 $\mathcal{I}_P \leftarrow \mathcal{I}_P - \{\bar{i}\}$;

The advantages and drawbacks of the joint-inspired heuristic algorithm are as follows:

1) **Advantages**: Since the forest overlays are computed with the remaining capacity, this algorithm is able to construct overlays even when most capacities have been exhausted. This is because alternative solution may be found to avoid overusing the bandwidth of critical edge servers. It should result in no waste of resources.
2) **Drawbacks**: Algorithm 3 is oblivious to the amount of capacities that are utilized by the channels. That is, an important but very costly channel can be processed before some channels that are just slightly less important, but far cheaper in terms of resources. Moreover, if a new channel has to be delivered, or if a channel should not be delivered anymore, the algorithm should be executed from scratch.

VII. Evaluations

Extensive simulations are conducted to evaluate the performance of the proposed solutions in this paper: *JOP-ILP*, *SOP-ILP*, *JOP-heu*, *SOP1-heu* and *SOP2-heu*. In our evaluation, we use both the French CDN network (16 nodes and 36 links) as shown in Fig. 1 and the USA AtHome ISP backbone network (46 nodes and 55 links) [30] as the testbeds.

The following five metrics are used in our comparisons:

1) **Profit ratio**. It indicates the satisfaction of the service provider by measuring if channels are well delivered, according to their importance, i.e.,

$$\frac{\sum_{i\in\mathcal{I}} R_i \times \pi_i}{\sum_{i\in\mathcal{I}} \pi_i} \tag{27}$$

2) **Number of delivered video channels**. It is expressed as

$$\sum_{i\in\mathcal{I}} R_i \tag{28}$$

TABLE IV
HEURISTIC SOLUTIONS VS MILP SOLUTIONS: 6 CHANNELS

video bit-rate	*JOP-ILP*	*SOP-ILP*	*JOP-heu*	*SOP1-heu*	*SOP2-heu*
Profit ratio					
512 kbps	100%	100%	100%	100%	100%
1024 kbps	100%	90.7%	100%	100%	100%
1536 kbps	100%	76.7%	100%	100%	100%
2048 kbps	100%	67.4%	100%	93%	76.7%
Number of delivered channels					
512 kbps	6	6	6	6	6
1024 kbps	6	4.5	6	6	6
1536 kbps	6	3.5	6	6	6
2048 kbps	6	3.5	6	5	5
Ratio of used capacity					
512 kbps	12.7%	12.8%	14%	13.8%	13.8%
1024 kbps	21.7%	16.5%	23.7%	23.3%	23.3%
1536 kbps	30.8%	18.1%	33.2%	32.7%	32.7%
2048 kbps	40.4%	21.2%	42.6%	36%	34.3%
Average computing time (Seconds)					
2048 kbps	749.5	228	0.3	< 0.1	0.3

3) **Ratio of used capacity**. It is the ratio between the used bandwidth of all edge servers and their original available bandwidth, i.e.,

$$\frac{Cap}{\sum_{v \in V} c_v} \tag{29}$$

4) **Computing time**. It is the computing time for an algorithm to find a solution for delivering all video channels.

We set the channel importance such that it is proportional to the popularity of the channel. We assume an identical delay over all links in the network, which equals one unit delay. We use a Zipf distribution to model the popularity of channels (consequently their importance). The bandwidth of edge servers follows a lognormal distribution. The video was compressed with the H.264 coder at various bitrates ranging from 512 kbps to 2,560 kbps. The entrypoint applies rateless coding on each chunk and sends the encoded symbols in successive UDP packets. After receiving the packets, the subscribing edge servers apply rateless decoding to recover the original video. For rateless coding, the Raptor code model proposed in [8] was used. With this model, a redundancy of 5%, gives a very high probability of successful decoding [7], [8], [31]. Thus, for an original video bitrate of 512/1, 024/1, 536/2, 048/2, 560 kbps respectively, an entrypoint will generate a video channel of about 537/1, 075/1, 613/2, 150/2, 688 kbps respectively, and an edge server needs to receive the same rate of encoded symbols to recover the original chunk. In our simulations, *JOP-ILP* and *SOP-ILP* models are solved by ILOG CPLEX, while all heuristic algorithms are implemented directly in C++.

A. Comparison of MILP Solutions and Heuristic Solutions

We used the IBM ILOG CPLEX optimizer to solve the MILP problems. The solver is run on a PC equipped with 5 cores Intel(R) Xeon(R) 3.00 GHz CPUs, 8 GB RAM memory, and Windows 7 system. As MILP solutions are time consuming, simulations are done in the small French CDN network in Fig. 1. We present in Table IV numerical comparisons between the different solutions. Unfortunately, the proposed MILP models (*JOP-ILP* and *SOP-ILP*) were unable to obtain optimal solutions when we used larger instances (when the number of edge servers requesting channel i is large, the number of channels $|\mathcal{I}|$ is large and the video has large video bite rate, i.e., $\hat{K}$). This is why we use a small CDN topology and restrict the number of video channels to six ($|\mathcal{I}| = 6$) for performance comparisons. The bandwidth of each edge server follows a lognormal distribution with mean bandwidth 12 Mbps and heterogeneity of 0.1. We consider four different video bit rates: 512 kbps, 1,024 kbps, 1,536 kbps and 2,048 kbps. The tree height is bounded by four ($H = 4$). As reported in [32], Youtube video popularity follows Zipf-like distribution, similarly we suppose video channel importance has a distribution of Zipf(1,6), where the first parameter is the value of the exponent characterizing the distribution and the second parameter is the number of video channels. The number of edge servers that request channel i ranges from three to six according to the importance of the channel. The results presented in Table IV represent the average of five instances.

We make five main observations based on the comparison in Table IV. First, the *JOP-ILP* finds solutions that the *SOP-ILP* cannot find, especially when the resources are constrained. When the resources get tighter, the *SOP-ILP* may have disastrous performance due to its incapacity to prevent some critical edge servers to be exhausted and to disconnect the network. Second, the heuristic algorithms perform well. The *JOP-heu* is able to provide the optimal solution for the given configurations as *JOP-ILP*. Both the *SOP1-heu* and *SOP2-heu* algorithms require a small over-utilization of resources for video bit-rates 512 kbps and 1,024 kbps, while they miss one video channel delivery for 2,048 kbps. The third observation concerns the bandwidth consumption of edge servers. We can see that when the video bit-rate is 512 kbps, the French CDN is over-provisioned since all methods are able to transmit 6 video channels. In this configuration, the *JOP-ILP* solution consumes the least bandwidth, which is 12.7% of the total bandwidth ressource, while the other four methods result in almost the same bandwidth usage. The fourth observation is that the *SOP-ILP* may result in the worst solution. It is only able to achieve 67.4% profit ratio for the video bit-rate of 2 Mbps, which is even worse than the *SOP2-heu* with a profit ratio of 76.7%. It is because *SOP-ILP* searches minimum cost forest overlay without knowing the bandwidth usage by other video channels. As it is cost-optimal, it may use too much bandwidth on critical edge servers for each video channel. Consequently, few of them can be delivered successfully in the bandwidth allocation part of the solution. However, two-step inspired heuristic algorithms may find alternative approximated minimum cost solutions which does not require as much bandwidth on critical edge servers as that of *SOP-ILP*. Thus, more video channels may be delivered at the same time. Finally, the two MILP methods take a lot of time (up to 12.5 minutes for video bit rate of 2,048 kbps) to obtain a solution, while the computing time is less than one second for heuristic algorithms as presented in Table IV.

B. Evaluations on Real-Scale Systems

Here, we focus on the performance of the proposed heuristic algorithms: *JOP-heu*, *SOP1-heu* and *SOP2-heu*. We increase not only the network size but also the number of video channels and the video quality. Node bandwidth follows a lognormal

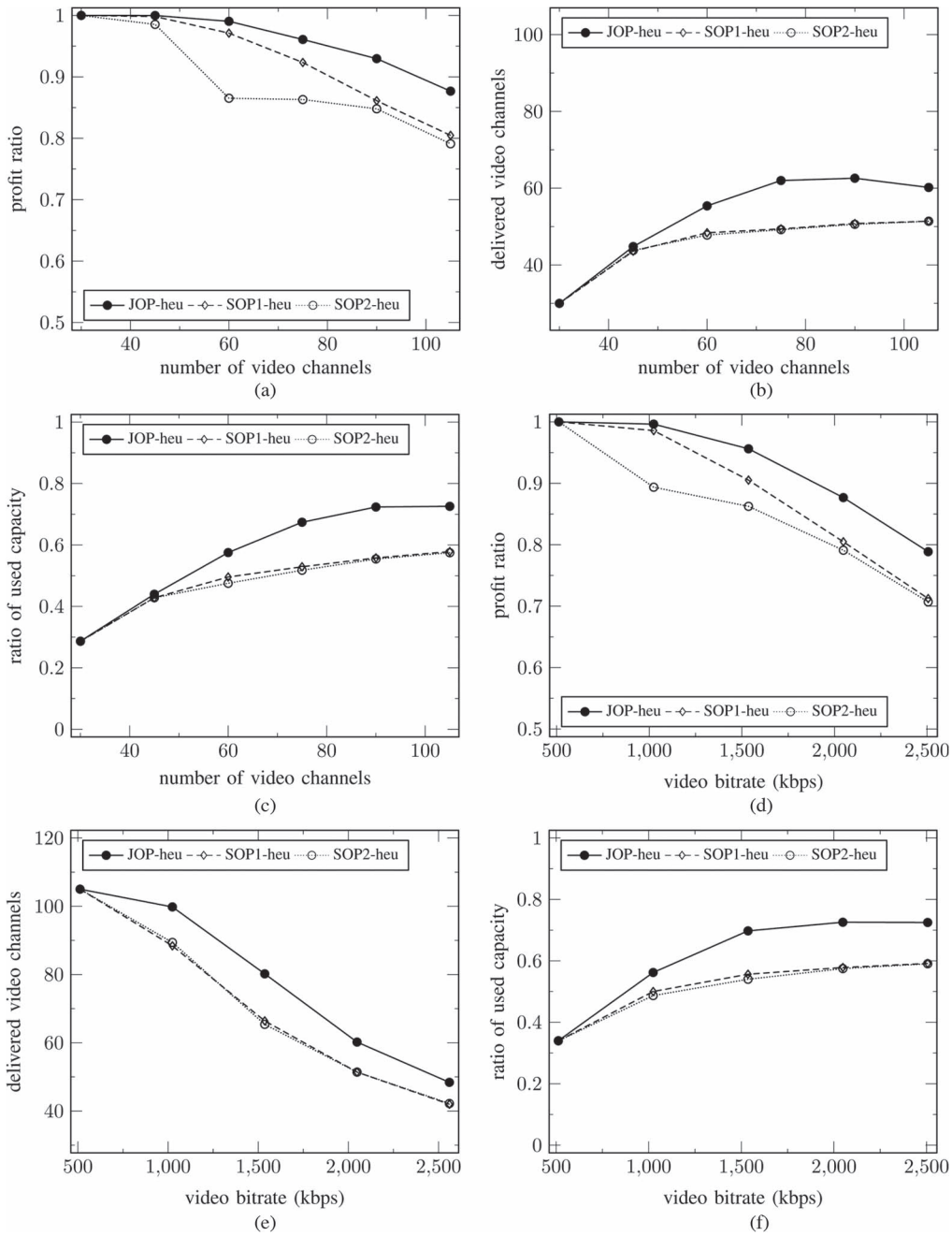


Fig. 3. Simulation Results in a French Telco-CDN. (a) profit ratio vs. channels; (b) delivered channels vs. channels; (c) capacity ratio vs. channels; (d) profit ratio vs. bit-rate; (e) delivered channels vs. bit-rate; (f) capacity ratio vs. bit-rate.

distribution with an average of 96 Mbps. The channel importance follows the Zipf(0.5,105) distribution. The simulations are run on a PC equipped with Intel Core 3.3 GHz CPU, 4 GB RAM memory, and Windows 7 system.

We first study the performance in the French CDN in Fig. 3(a)–(f). The height is bounded by six. The number of subscribers $|V_i|$ for a channel i ranges from 3 to 9. Thus, there are some configurations where the network is over-provisioned (e.g., when $|\mathcal{I}| \in \{30, 45\}$), and some others where not all channels can be delivered.

In Fig. 3(a)–(c), the video bit-rate is 2,048 kbps, and we evaluate the performance of our algorithms with respect to the number of channels from 30 to 105. We found that the proposed three heuristic algorithms serve well the most important channels, so the overall profit is almost the same for all algorithms in the over-provisioned network configuration. In the under-provisioned scenario, i.e., when the number of channels is bigger than 45, all algorithms obtain an achievable profit ratio no less than 80%. The *JOP-heu* algorithm is always able to obtain up to 10% higher profit ratio and deliver 15 more video channels than the two-step heuristic algorithms. Thus, it clearly results in high network bandwidth usage, i.e., 17% higher. Moreover, the simple importance-first heuristic *SOP1-heu* performs better than *SOP2-heu* in term of achievable profit ratio, while the latter one tends to accept more video channels with smaller bandwidth requirement. This may be explained by

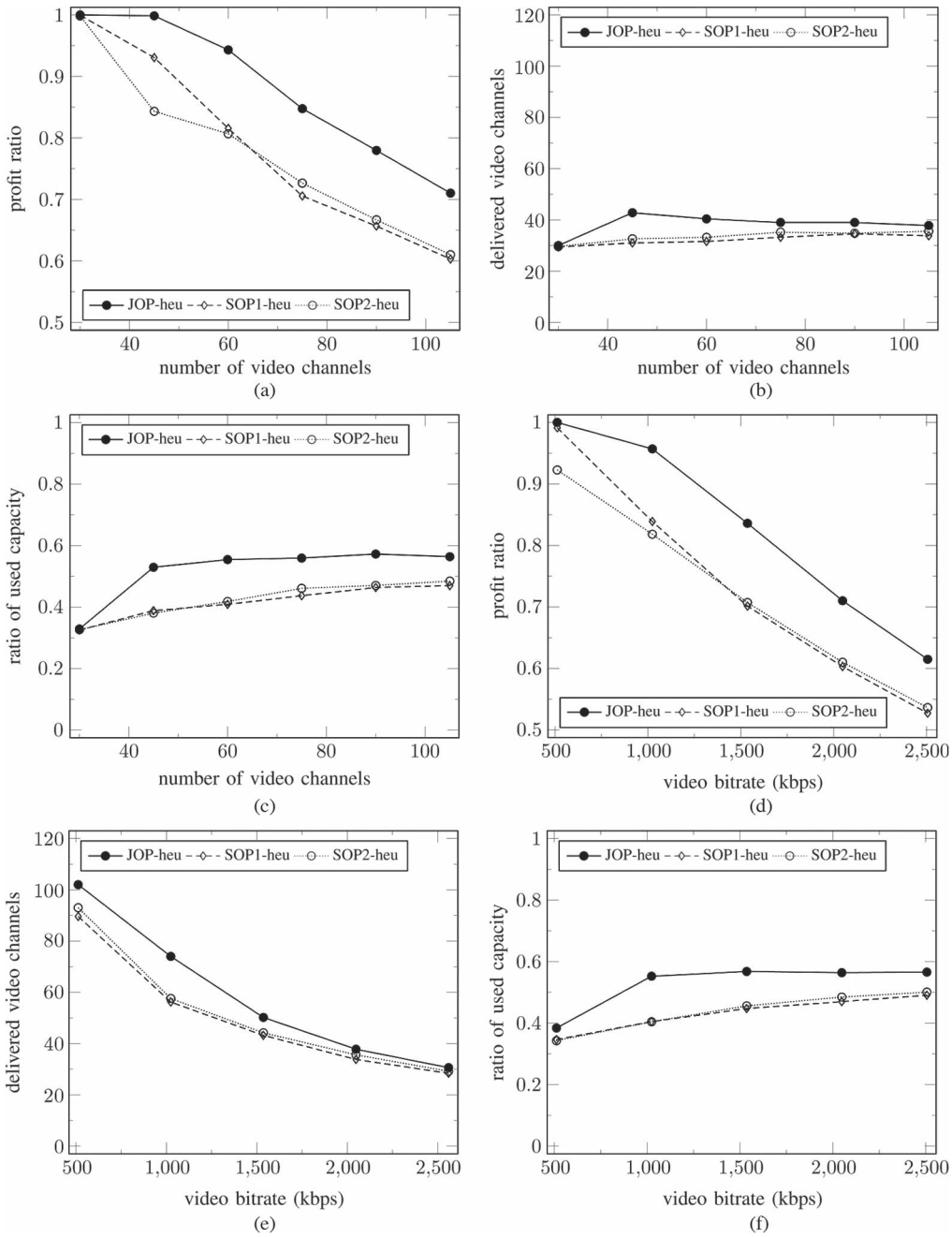


Fig. 4. Simulation Results in USA AtHome ISP Backbone Network [31]. (a) profit ratio vs. channels; (b) delivered channels vs. channels; (c) capacity ratio vs. channels; (d) profit ratio vs. bit-rate; (e) delivered channels vs. bit-rate; (f) capacity ratio vs. bit-rate.

the fact that a video channel with smaller importance requires less bandwidth due to the Zipf distribution of importance value. The *SOP1-heu* packs the most important videos better than *SOP2-heu*, meaning that the more sophisticated utility score of *SOP2-heu* is counter-productive.

In Fig. 3(d)–(f), the number of channels is fixed to $|\mathcal{I}| = 105$ and we study the performance of our algorithms when the video bitrate varies from 512 kbps to 2,560 kbps. We can find that *JOP-heu* algorithm is always superior to the others. As the video bitrate increases, the achievable profit ratio degrades slightly while the number of delivered videos diminishes sharply. This is due to the lack of bandwidth in the video entrypoints or the edge servers around the videos entrypoints for accommodating high quality videos. It also should be noted that the network bandwidth usage is up to 72%, which is not high. This is because, we employ a mesh topology (c.f. Fig. 1), where there are only three entrypoints. Since an entrypoint node does not have a link to all the edge servers, the bandwidth of an edge server far away from the entrypoint node may be wasted when the neighbors of the entrypoint exhaust their bandwidth.

We have also performed simulations on the USA AtHome ISP backbone network (46 nodes and 55 links) [30], which is nearly three times bigger than the French Telco-CDN. The objective is to validate our results and show the scalability of our heuristic algorithms. We adopted the same simulation configuration as that in the French CDN, except that the number

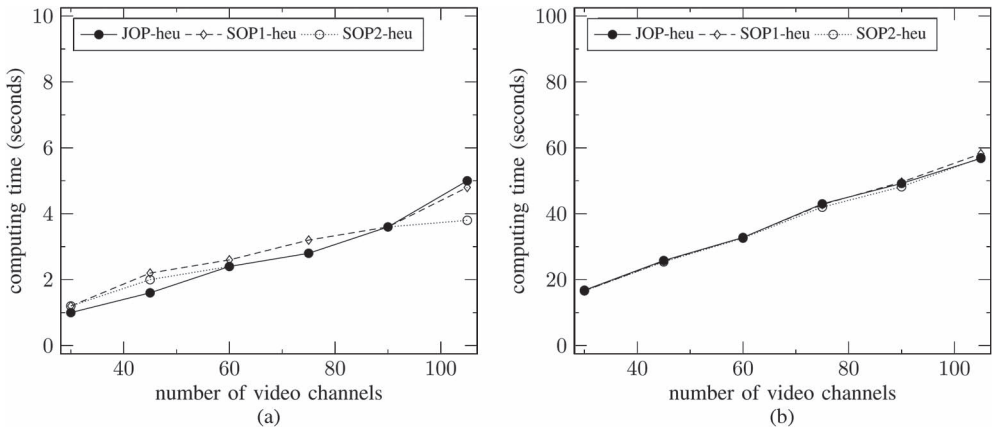


Fig. 5. Average Computing Time (Seconds) on Real-Scale Systems: 30 ∼ 105 channels, video bitrate 2048 kbps. (a) Computing time in French-CDN (16 nodes); (b) computing time in AtHome ISP (46 nodes).

of subscribing edge servers ranges from 3 to 23 and six entry-points are used. The height of delivery tree is bounded by 20. According to the obtained results in Fig. 4(a)–(f), *JOP-heu* is also able to achieve at least 10% higher profit ratio and deliver more video channels than the others in the AtHome network. We also note that this improvement is valid for all high video bitrates. The *SOP1-heu* algorithm performs slight better than *SOP2-heu* algorithm. These results confirm the superiority and the scalability of our *JOP-heu* algorithm.

Furthermore, we have measured the computation complexity of the proposed heuristic algorithms, which are shown in Fig. 5(a) and (b). Given 105 high quality video channels of 2,048 kbps, all the three heuristic algorithms need about 5 seconds to find a solution for delivering all channels in the French CDN, while they require about 58 seconds for the big USA AtHome ISP backbone network. From the curves, we can see that only 3 seconds and 29 seconds are needed respectively in the two topologies to find a solution for delivering the fifty featured channels of twitch.tv. Recent research [33] on dynamic live streaming systems like twitch.tv shows that there is a strong correlation between the popularity of the video channels and their "stability", i.e., how long are the channels online. It is worth noting that CDNs are used for the most popular video channels, which are stable. Thus, it is reasonable to conclude that the proposed algorithms are time efficient for video delivery in Telco-CDNs.

In summary, the proposed heuristic algorithms perform well in both over-provisioned and under-provisioned network configurations, and they are simple to implement. Among them, *JOP-heu* performs best since it is able to compute fast a near-optimal solution. When the resources are more constrained, *JOP-heu* clearly outperforms the *SOP*-heuristics, serving up to one third of the channels more. In the considered Telco-CDNs, such gain in performance is enough to justify the implementation of *JOP-heu* for the many cases where resources are constrained.

VIII. Conclusion

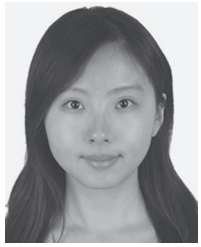
This paper addressed the problem of overlay construction and bandwidth allocation for delivering video channels from the entrypoints of the Telco-CDN to edge servers. The pursued goal is to maximize the total profit of delivered channels while preserving network resources. To this end, two optimization methods have been compared: joint optimization and two-step optimization. Our work analyzed the relevance of implementing joint optimization approaches, which are theoretically more efficient. We believe such analysis will continue flourishing in the future since resource allocation in capacity-constrained environments is often one of the multiple optimization problems to solve. In this respect, this paper explored some of the trade-offs at stake and provided insights to network operators for making informed resource provisioning decisions in the support of a Telco-CDN. As a future work, we will extend our optimization methods to solve network planning problems for adaptive video streaming systems.

References

[1] F. Zhou, J. Liu, G. Simon, and R. Boutaba, "Joint optimization for the delivery of multiple video channels in Telco-CDNs," in *Proc. IEEE/ACM CNSM*, Oct. 2013, pp. 161–165.
[2] Netflix Open Connect Content Delivery Network. [Online]. Available: https://signup.netflix.com/openconnect
[3] V. K. Adhikari *et al.*, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," in *Proc. IEEE INFOCOM*, 2012, pp. 1620–1628.
[4] Z. Li and G. Simon, "In a Telco-CDN, pushing content makes sense," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 3, pp. 300–311, Sep. 2013.
[5] M. Adler, R. K. Sitaraman, and H. Venkataramani, "Algorithms for optimizing the bandwidth cost of content delivery," *Comput. Netw.*, vol. 55, no. 18, pp. 4007–4020, Dec. 2011.
[6] J. M. Almeida, D. L. Eager, M. K. Vernon, and S. J. Wright, "Minimizing delivery cost in scalable streaming content distribution systems," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 356–365, Apr. 2004.
[7] N. Thomos and P. Frossard, "Network coding of rateless video in streaming overlays," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 12, pp. 1834–1847, Dec. 2010.
[8] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
[9] C. Wu and B. Li, "rstream: Resilient and optimal peer-to-peer streaming with rateless codes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 1, pp. 77–92, Jan. 2008.
[10] F. Zhou, S. Ahmad, E. Buyukkaya, R. Hamzaoui, and G. Simon, "Minimizing server throughput for low-delay live streaming in content delivery networks," in *Proc. ACM NOSSDAV*, 2012, pp. 65–70.
[11] N. Magharei, R. Rejai, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *Proc. IEEE INFOCOM*, May 2007, pp. 1424–1432.
[12] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM*, 2002, pp. 205–217.

[13] M. Hosseini, D. Ahmed, S. Shirmohammadi, and N. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 58–74, 3rd Quart. 2007.
[14] K. Jinu and S. Kamil, "A survey on the design, applications, enhancements of application-layer overlay networks," *ACM Comput. Surv.*, vol. 43, no. 1, pp. 1–34, Dec. 2010.
[15] M. Castro *et al.*, "Splitstream: High-bandwidth multicast in cooperative environments," in *Proc. ACM SOSP*, 2003, pp. 298–313.
[16] S. Liu *et al.*, "P2p streaming capacity under node degree bound," in *Proc. IEEE ICDCS*, Jun. 2010, pp. 587–598.
[17] R. Sweha, V. Ishakian, and A. Bestavros, "Angelcast: Cloud-based peer-assisted live streaming using optimized multi-tree construction," in *Proc. ACM MMSys*, 2012, pp. 191–202.
[18] M. Grangetto, R. Gaeta, and M. Sereno, "Rateless codes network coding for simple and efficient P2P video streaming," in *Proc. IEEE ICME*, Jun. 2009, pp. 1500–1503.
[19] H. R. Oh, D. O. Wu, and H. Song, "An effective mesh-pull-based p2p video streaming system using fountain codes with variable symbol sizes," *Comput. Netw.*, vol. 55, no. 12, pp. 2746–2759, Aug. 2011.
[20] H. Takahashi and A. Matsuyama, "An approximate solution for the steiner problem in graphs," *Math. Japonica*, vol. 24, no. 6, pp. 573–577, 1980.
[21] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for steiner trees," *Acta Informatica*, vol. 15, no. 2, pp. 141–145, 1981.
[22] C. A. Noronha and F. Tobagi, "Optimum routing of multicast streams," in *Proc. IEEE INFOCOM*, Jun. 1994, pp. 865–873.
[23] C.-F. Wang, C.-T. Liang, and R.-H. Jan, "Heuristic algorithms for packing of multiple-group multicasting," *Comput. Oper. Res.*, vol. 29, no. 7, pp. 905–924, Jun. 2002.
[24] C. A. Oliveira, P. M. Pardalos, and M. G. Resende, "Optimization problems in multicast tree construction," in *Handbook of Optimization in Telecommunications*. New York, NY, USA: Springer-Verlag, 2006, pp. 701–731.
[25] Q. Zhang and Y.-W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 53–62, Apr. 1999.
[26] J. Liu, C. Rosenberg, G. Simon, and G. Texier, "Optimal delivery of rate-adaptive streams in underprovisioned networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 706–718, Apr. 2014.
[27] J. Puchinger, G. R. Raidl, and U. Pferschy, "The multidimensional knapsack problem: Structure and algorithms," *INFORMS J. Comput.*, vol. 22, no. 2, pp. 250–265, Apr. 2010.
[28] A. Fréville, "The multidimensional 0-1 knapsack problem: An overview," *Eur. J. Oper. Res.*, vol. 155, no. 1, pp. 1–21, May 2004.
[29] R. Loulou and E. Michaelides, "New greedy-like heuristics for the multidimensional 0-1 knapsack problem," *Oper. Res.*, vol. 27, no. 6, pp. 1101–1114, 1979.
[30] K. Noriaki, M. Tatsuya, K. Ryoichi, H. Shigeaki, and H. Haruhisa, "Analyzing influence of network topology on designing ISP-operated CDN," *Telecommun. Syst.*, vol. 52, no. 2, pp. 969–977, 2013.
[31] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, "Reliable multimedia download delivery in cellular broadcast networks," *IEEE Trans. Broadcast.*, vol. 53, no. 1, pp. 235–246, Mar. 2007.
[32] A. Abdolreza and S. Mojgan, "Workload generation for youtube," *Multim. Tools Appl.*, vol. 46, no. 1, pp. 91–118, Jan. 2010.
[33] P. Karine and S. Gwendal, "Dash in twitch: Adaptive bitrate streaming in live game streaming platforms," in *Proc. ACM VideoNext*, 2014, pp. 13–18.

Fen Zhou received the Ph.D. degree in computer science from INSA Rennes (France) in 2010. He is currently an Associate Professor at the Computer Science lab (LIA) of the University of Avignon, France. His research interests include routing, resource allocation and survivability in optical networks, content delivery networks (CDNs), and vehicular networks.



Jiayi Liu is currently working as a Lecture in Xidian University, Xi'an, China. She received her Master Degree in 2009 and her PhD Degree in 2013, both in Computer Science from Telecom Bretagne, France. Her reserch interests include video streaming systems, bandwidth efficient streaming solutions, and content distribution in mobile networks.

Gwendal Simon received his Master Degree in Computer Science in 2000 and his PhD degree in Computer Science in December 2004 from University of Rennes 1 (France). From 2001 to 2006 he was a researcher at Orange Labs, where he worked on peer-to-peer networks and social media innovations. Since 2006, he is Associate Professor at Telecom Bretagne, a graduate engineering school within the Institut Mines-Telecom. He has been a visiting researcher at University of Waterloo from September 2011 to September 2012. His research interests include large-scale networks, optimization problems and video delivery systems.

Raouf Boutaba received the M.Sc. and Ph.D. degrees in computer science from the University Pierre and Marie Curie (France) in 1990 and 1994, respectively. He is currently a Professor of computer science with the University of Waterloo (Canada). His research interests include control and management of networks and distributed systems. He is a fellow of the IEEE and the Engineering Institute of Canada.