國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

具深度資訊之 3D 視訊於快取之應用

Efficient Caching for Multi-view 3D Videos with Depth-Image-Based Rendering

李繼唐

Ji-Tang Lee

指導教授：廖婉君 博士

Advisor: Wanjiun Liao, Ph.D.

中華民國 105 年 7 月

July 2016

doi:10.6342/NTU201601753

國立臺灣大學碩士學位論文
口試委員會審定書

具深度資訊之 3D 視訊於快取之應用
Efficient Caching for Multi-view 3D Videos with
Depth-Image-Based Rendering

本論文係李繼唐君（R03942049）在國立臺灣大學電信工程學研究所完成之碩士學位論文，於民國 105 年 7 月 20 日承下列考試委員審查通過及口試及格，特此證明

口試委員：[illegible]（簽名）
（指導教授）

楊得年

周承復　林宗男

所　　長　吳宗霖（簽名）

誌謝

可以完成這篇論文，要感謝的人太多了，首先感謝論文的指導教授廖婉君老師，以及中研院的大學長楊得年老師。廖老師雖然很忙，但還是謝謝您這兩年來的指導，從第一次 meeting 跌跌撞撞，到中間被電得唏哩花啦，到最後可以得到您的肯定，這中間也越來越清楚要怎麼樣做研究才是正確的方向與態度。謝謝楊得年老師，謝謝您提供的許多幫助，從論文開始之前的訓練、創新的問題、數學部分嚴謹的檢討、還有這一年半來許多額外的訓練與幫忙、生涯規劃等等，真的讓我非常非常感激！

謝謝實驗室的學長姐們，在我低潮的時候給予鼓勵。謝謝江江，從大四下的專題開始，就很照顧我，而且看你的 presentation 也讓我學了不少，對於你是滿滿的欽佩。謝謝萱萱，也是從大四下開始就認識了，總能夠讓我面對研究（或老師）時不要太緊張。謝謝弘諺，雖然我總是嗆你而你也總是裝弱，但還是給了我許多幫助。謝謝昱均，論文的背景知識如果沒有你，我肯定要摸索很久，我就像是站在你的基礎上建設一樣，謝謝你。謝謝學弟們，短短相處一年，有很多有趣的事情發生，祝你們都能順利完成研究。謝謝實驗室助理筑涵，幫了我們許多忙，默默擋了很多箭哈哈。

謝謝我的兩位戰友彥增與柏璇，雖然我跟你們做的東西完全不相干，也比較少參與你們的討論，但兩年的同甘共苦還是很充實！原來我們直到口試完才一起第一次單獨出去玩，之後還會有機會一起吃吃喝喝的。也謝謝另外兩位 Social 組的夥伴，昶亦跟伯翰，很開心大家可以一起奮鬥一起哀嚎，祝你們四位朋友未來都順順利利的！

謝謝中研院的朋友們。謝謝睿翊，幫我跑了一堆有的沒的模擬，還要被楊老師電得東倒西歪，繼續加油啊！謝謝瑄瑜也是一起幫忙了好多。謝謝良豪學長的提點，謝謝讀書會朋友的分享，都讓我學習了不少。謝謝帥，從大學就受你照顧，一直到研究所還是一直受到你庇蔭哈哈。

doi:10.6342/NTU201601753

謝謝媽媽跟姐姐，雖然不知道兒子跟弟弟在幹嘛，怎麼都很忙的感覺，還是可以當我背後的後盾，讓我好好完成學業。謝謝孟涵，謝謝你總聽我吐苦水，謝謝你忍受我寫論文或者打 code 的冷漠孤僻，謝謝你的陪伴，謝謝你的積極，讓我可以有現在的成果、以及未來的工作機會，辛苦你了！

最後要謝謝自己，算是不負自己和老師的期許，做出了還算完整的作品。目前階段性任務已經完成，未來還要延伸完成的部分也要繼續加油。也期許自己未來三年可以有不錯的生活態度、工作態度，不要忘了這一切都是多麼得來不易，是有如此多人的幫忙才能有目前的成績，好還要更好！

加油！繼唐！

2016.7.31

doi:10.6342/NTU201601753

中文摘要

近幾年隨著支援 3D 影片與虛擬實境的裝置問世，多視角 3D 影片將扮演著重要的角色。與傳統單一視角的影片相比，多視角影片不可避免地需要更大的空間來儲存。然而尚未有文獻提及，如何將多視角影片在代理伺服器的快取中有效地儲存、置換。3D 影片的深度資訊合成技術使得使用者可以透過鄰近的視角合成出使用者想要觀看的視角，因此，我們可以大幅降低一部 3D 影片所佔的儲存空間。我們提出了一個新的快取置換問題，叫做視角選擇與快取操作問題，並且利用馬可夫決策過程找到此問題的最佳解。另外，為了解決馬可夫決策過程在大案例會有複雜度過高的問題，我們也提出了一個啟發式演算法以解決此問題。模擬結果顯示，與過去的快取置換策略相比，我們所提出的演算法與馬可夫決策過程的最佳解可以大幅提升快取命中的比率，並且降低快取在置換、傳送影片時的遲滯以及所消耗的頻寬。

關鍵字：多視角 3D 影片 、深度資訊合成技術、代理伺服器、快取

doi:10.6342/NTU201601753

ABSTRACT

Due to the emergence of mobile 3D and VR devices, multi-view 3D videos are expected to play increasingly important roles shortly. Compared with traditional single-view videos, it is envisaged that a multi-view 3D video requires a larger storage space. Nevertheless, efficient caching of multi-view 3D videos in a proxy has not been explored in the literature. In this thesis, therefore, we first observe that the storage space can be effectively reduced by leveraging Depth Image Based Rendering (DIBR) in multi-view 3D. We then formulate a new cache replacement problem, named View Selection and Cache Operation (VSCO), and find the optimal policy based on Markov Decision Process. In addition, we devise an efficient and effective algorithm, named Efficient View Exploration Algorithm (EVEA), to solve the problem in large cases. Simulation results manifest that the proposed algorithm can significantly improve the cache hit rate and reduce the total cost compared with the previous renowned cache replacement algorithms.

Key Words: Multi-view 3D video, DIBR, proxy caching.

doi:10.6342/NTU201601753

CONTENTS

doi:10.6342/NTU201601753

doi:10.6342/NTU201601753

LIST OF FIGURES

doi:10.6342/NTU201601753

LIST OF TABLES

doi:10.6342/NTU201601753

LIST OF ALGORITHMS

doi:10.6342/NTU201601753

Chapter 1 Introduction

1.1 Background

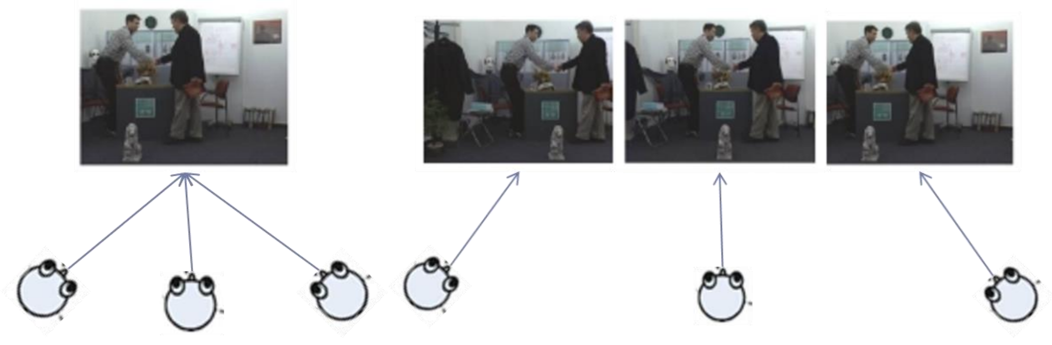
The need of data traffic has grown tremendously in mobile broadband networks, whereas video data are expected to occupy more than 70% of total traffic in 2020 [1]. Particularly, high-resolution and multi-view 3D videos have received much attention recently from both research and industry communities. Video service providers, such as YouTube and Netflix, now provide 3D videos and 3D live streaming services, whereas multi-view 3D videos further allow a user to select the preferred view and thereby stimulate innovative applications in television, movies, education, advertising, and virtual reality (VR). With the emergence of various VR devices, such as Google cardboard, Oculus Rift, Samsung Gear VR, and HTC VIVE, multi-view 3D videos are envisaged to play increasingly important roles in the near future.

1.1.1 Multi-view 3D Video

In contrast to traditional single-view 3D videos with only one view, a multi-view 3D video taken by cameras array from different positions and angles typically offers 5, 16, and 32 different view angles [2]. Each video sequence in the multi-view video represents a unique viewpoint of the scene, creating several times larger traffic than traditional

doi:10.6342/NTU201601753

multimedia. Fig. 1.1(a) is an example for single-view video scenario; users at different positions perceive the same viewing scene. However, a multi-view scenario as Fig. 1.1(b) allows users to have different viewing scenes at different positions; therefore, the users can have a better experience. There are many applications for such multi-view 3D video, such as 3DTV, free viewpoint video, remote surgery and wireless multimedia networks.



(a) Single-view (b) Multi-view

Fig. 1.1 An Example for 3D Video

Several 3D video data formats and 3D video coding strategies currently co-exist, among which multi-view video plus depth (MVD) format has emerged as an efficient data representation for 3D video scene. Compared to multi-view 2D video format which synthesizes scenes by using image interpolation, the main advantage of MVD format is that virtual views at arbitrary viewpoint positions can be conveniently generated via *Depth-Image-Based Rendering* (DIBR) [3] technique for interactive application.

doi:10.6342/NTU201601753

1.2 Related Works

It is not surprising that a multi-view 3D video requires a larger storage space in a video proxy. Nevertheless, efficient caching of multi-view 3D video in a proxy has not been studied in the literature. Recent researches have shown that media objects can benefit from proxy caching [4], whereas various cache replacement algorithms have been proposed [5]. It has been reported that combining different replacement scheme, such as Least Recently Used (LRU) [6] and the 80/20 law, can effectively lower the miss rate [7]. Also, caching in wireless networks to create more Coordinated Multipoint Transmission (CoMP) opportunities between the relay and macro base stations has been studied [8], whereas User Preference Profiles (UPPs) with adaptive bit rate control are exploited to further maximize the cache hit rates [9]-[12]. Moreover, leveraging the preference of friends in social networks to predict the future need has also been incorporated in the cache design [13]. Nevertheless, the above researches are not designed for multi-view 3D videos.

1.3 Motivation and Challenges

In traditional cache problems, a data item can only serve the users that request the item. In contrast, for multi-view 3D videos, DIBR allows a user to synthesize the desired

doi:10.6342/NTU201601753

view from the nearby left and right views because adjacent views usually share many similar parts. Researchers in computer vision and video processing have introduced many sophisticated algorithms for DIBR to ensure the synthesized quality is very close to the original view, by optimizing the bit allocation between the texture and depth map [14]-[16]. With DIBR, a view cached in a proxy is able to serve not only the users requesting the view but also other users subscribing to nearby views. Nevertheless, the nearby left view and right view cannot be arbitrarily far away to ensure a guaranteed video quality [17]. Specifically, the quality constraint in DIBR specifies that the left and right view are allowed to be at most D views away (i.e., $D-1$ views between them). Therefore, it is desired to design a new cache replacement algorithm to support DIBR in multi-view 3D videos.

The difference of caching with DIBR and traditional caching are three-fold. 1) In traditional caching, if a requested view is not cached in a proxy, the proxy needs to access the remote video server to acquire the subscribed view. For caching with DIBR, if the nearby left and right views are cached in a proxy, the proxy can directly send the two views to the user. Nevertheless, the distance between the two views is constrained, and properly selecting the views to be cached is crucial. 2) When a cache miss occurs in traditional caching, the proxy can only access the missed view from the remote video

doi:10.6342/NTU201601753

server. By contrast, for caching with DIBR, the proxy can access the requested view or a nearby left (or right) view when another nearby right (or left) view has been stored. 3) In addition to the cached views and accessed views, the replacement of views is also different from the traditional cache. A view whose request frequency or recency is small or old, is usually evicted by the proxy. In contrast, for caching with DIBR, the view can a play an important role when it can be exploited to synthesize many popular nearby views. Therefore, the selection of the cached views, accessed views and replaced views becomes more flexible and thus more challenging in caching with DIBR in multi-view 3D because more candidate views are necessary to be considered.

In this thesis, therefore, we formulate a new cache replacement problem, named *View Selection and Cache Operation Problem* (VSCO). Given the initial cache state, user request, and DIBR constraint, we aim to find the best views to be fetched, evicted (i.e., replaced), and returned to the users. The objective is to minimize a cost function of miss count, remote access cost, local transmission cost and the synthesis penalty. To solve VSCO, we model the cache replacement as a Markov Decision Process (MDP) [18] to find the optimal solutions in small cases and then extract the intrinsic ideas behind to devise an efficient and effective heuristic algorithm, named *Effective View Exploration Algorithm* (EVEA), for large cases. Simulation results manifest that, our proposed

doi:10.6342/NTU201601753

approaches can gain 30% hit rate and reduce 30% costs against traditional cache replacement policies.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 describes the system model and the problem formulation, and algorithm EVEA is presented in Chapter 3. The simulation results are summarized in Chapter 4, and finally, we conclude this thesis in Chapter 5.

doi:10.6342/NTU201601753

Chapter 2 View Selection and Cache Operation Problem

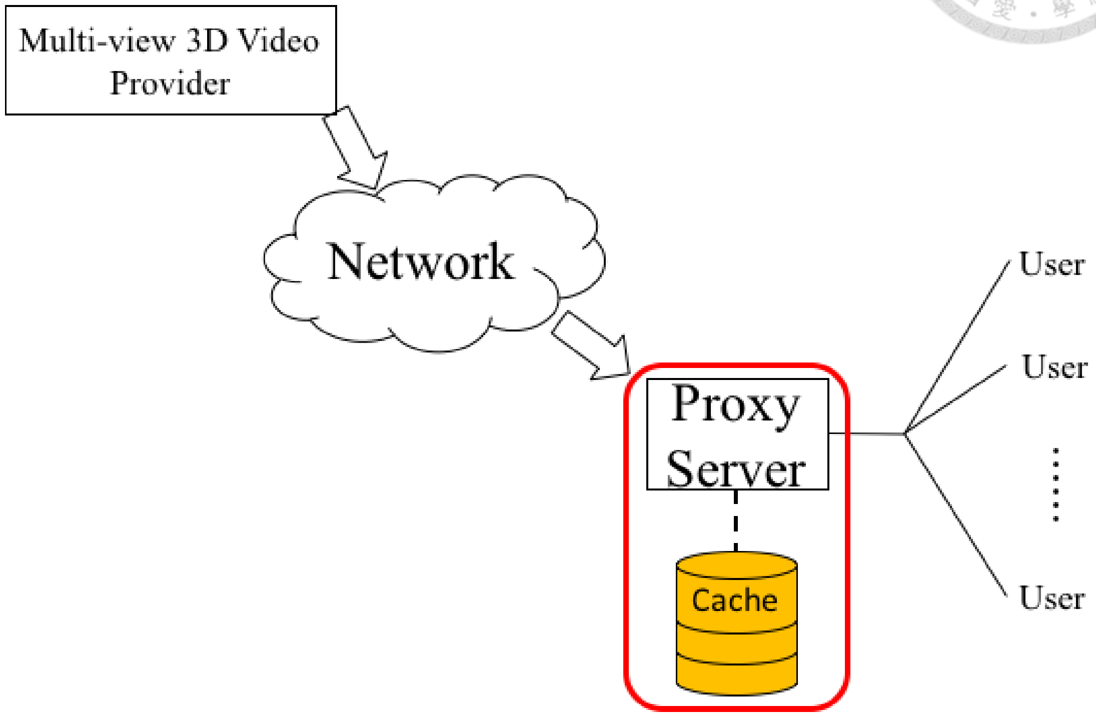


Fig. 2.1 System Model

2.1 System Model

We consider a proxy server with a cache, which caches many multi-view 3D video as the red part in Fig. 2.1. The proxy is connected to many users and the multi-view 3D videos are requested from remote video providers. The architecture to store a multi-view 3D video is similar to the one for traditional videos. Each view of a video is segmented into many small chunks [19], whereas VCR functionality according to the dynamic user requests with a tunable-victimization procedure can also be supported [20]. When a user request (for accessing a new view or switching to another view) arrives, the proxy can

doi:10.6342/NTU201601753

directly serve the user (for a cache hit) or wait until it acquires the video chunks from a remote video server (for a cache miss). Nevertheless, the differences of a multi-view 3D video proxy and the traditional proxy are the definitions of a cache hit, a cache miss, and the corresponding cache replacement policy, because DIBR in multi-view 3D enables the proxy to serve a user request even when the desired view is not cached. In the following, to explore various possibilities to serve a view with different nearby views, we first focus on the selection of cached views in a proxy for a video and then extend it to multiple videos in Chapter 3.3.

Specifically, let $V \subseteq \mathbb{N}$ denote the universal set of views in a multi-view 3D video. A proxy can cache at most N views for a multi-view 3D video, where $1 \leq N \leq |V|$. Let $\mathcal{V}_t$ denote the state of the cache. A cache hit for a view request v occurs in the following two cases. 1) The request view v is stored in the cache, $v \in \mathcal{V}_t$. Then the proxy can directly return the view v to the user. 2) The view v is not cached but its nearby left view and right view, say v_l and v_r, are in the cache, where $|v_l - v_r| \leq D$. The relationship between the setting of D and the video quality has been quantified in [15],[17], and the quality degradation is also incorporated in our cost model (detailed later). In this case, the cache can serve the rsequest by returning this view pair $(v_l,\ v_r)$ to the user. Note that the bandwidth consumption from a proxy to the user client grows when

doi:10.6342/NTU201601753

the proxy plans to serve a view request with two nearby views, and it is thus important to incorporate the above bandwidth cost into the objective function (described later).

On the other hand, for a cache miss, note that the traditional proxy can only access the requested view from the remote video server. In contrast, a multi-view 3D video proxy is allowed to access the nearby one or two views (e.g., when they are more popular than the desired view). More specifically, the selection of views in the cache replacement policy for multi-view 3D includes the following three parts: 1) The desired view v directly, 2) a nearby left or right view, v_l or v_r, and 3) both nearby views. Therefore, it is envisaged that a cached view in a multi-view 3D video proxy is able to serve more users (i.e., not only the users requesting the view but also the users requesting nearby views). Nevertheless, finding the optimal cache policy becomes more challenging because more factors, such as the additional bandwidth and synthesis costs in DIBR, are necessary to be considered carefully.

For practical situations, the computation overhead and extra energy consumption incurred by DIBR is small enough to be supported by current mobile devices [21],[22]. For HTTP video streaming (ex., YouTube and MPEG-DASH) with TCP [23],[24] (instead of UDP in Skype), DIBR can be performed when the views are waiting in the streaming buffer before playback, and thus no extra delay will be incurred.

doi:10.6342/NTU201601753

Whenever the proxy needs to access the remote video server and returns the view(s) to the user, two types of cost (i.e., the remote access cost and local transmission cost) are involved. Let c_f and c_s denote the bandwidth costs to fetch a view from the remote server and the cost to return a view to the user, respectively. Let c_m denote a penalty cost for a cache miss. Finally, to encourage directly transmitting the desired view, a view synthesis cost (representing the DIBR computation cost and the quality degradation) $c_p = f(v_l, v_r)$ is introduced according to v_l and v_r, and a larger cost is induced when the two views are more distant. Note that c_p can be set according to [15],[17],[21],[22].

2.2 Problem and Markov Decision Process

In this section, we first formulate a new optimization problem, named *View Selection and Cache Operation Problem* (VSCO), for cache replacement in a multi-view 3D video proxy. Given the universal set of views V, size of the cache N, DIBR synthesize range D, the initial cache state $\mathcal{V}_0$, and the request $\mathcal{R}$ (which can be described as a stochastic process, as explained later in this section), the problem is to select 1) the appropriate view(s) to be accessed from the remote server, 2) the view(s) to be replaced (i.e., evicted) from cache, and 3) the view(s) to be returned to the user. The objective is to minimize the average cost over an infinite time horizon by carefully selecting the views to be fetched,

doi:10.6342/NTU201601753

evicted, and delivered whenever a request arrives.

Notation	Description
V	The universal set of views in a multi-view 3D video
N	The size of the cache
D	The synthesize range of DIBR
$\mathcal{S}_t$ $\mathcal{V}_t$ $\mathcal{R}_t$	The global state of MDP model at time t, $\mathcal{S}_t = \{\mathcal{V}_t, \mathcal{R}_t\}$ The state of the cache at time t The request at time t
$\mathcal{A}_t$ V_f V_e V_s	The action takes at time t, $\mathcal{A}_t = \{V_f, V_e, V_s\}$ The view set to be fetched The view set to be evicted The view set to be sent to the user
q_{ij}	The request transition probability When the current request is view i, the next request will be view j with probability q_{ij}

doi:10.6342/NTU201601753

$C(s,a)$	The induced cost when action a is taken at state s
c_m	The cost when a cache miss happens
c_f,c_s	The cost to fetch and send a view, respectively
c_p	The quality degradation penalty when a user receives a view pair for synthesis

Table 2.1 Notation Table

In the following, we first present a Markov Decision Process (MDP) to find the optimal solution of VSCO. Markov Decision Process aims to minimize the long-term average cost in a proxy for an infinite horizon (i.e., infinite time). Table 2.1 summarizes the notations throughout this paper. The MDP model $\{\mathcal{S}, \mathcal{A}, C(s,a)\}$ consists of the state space $\mathcal{S}$, the action space $\mathcal{A}$, and the cost function C, respectively. Decision epochs (indexed by $t = 0,1,2$ ...) are defined as the instants with view requests arriving at the proxy. The status of time t is described by $\mathcal{S}_t = \{\mathcal{V}_t, \mathcal{R}_t\}$, where $\mathcal{V}_t$ denotes the state of the cache, and $\mathcal{R}_t$ denotes the view requested at time t. Note that $\mathcal{V}_t$ is a subset of V of cardinality $|\mathcal{V}_t| \leq N$, and let $\mathcal{V}_t^s$ denote the views that can be *synthesized* by $\mathcal{V}_t$ according to the DIBR synthesize range D. For example, if $|V| = 16$, $\mathcal{V}_t = \{2,5,9,11\}$ and $D = 3$, then $\mathcal{V}_t^s = \{3,4,10\}$. Notice that $\mathcal{V}_t \cap \mathcal{V}_t^s = \emptyset$.

When a view request $\mathcal{R}_t$ arrives at the proxy with the state of cache as $\mathcal{V}_t$, let $\mathcal{A}_t$

doi:10.6342/NTU201601753

denote the decision of the proxy for the request $\mathcal{R}_t$. There are three components in a decision action $\mathcal{A}_t = \{V_f, V_e, V_s\}$, where $V_f \subseteq V - V_t$ denotes the view set required to be fetched from remote video server, and $V_e \subseteq V_t \cup V_f$ represents the view set to be evicted (i.e., replaced) from a proxy when the proxy is full. Finally, let $V_s \subseteq V_t \cup V_f$ be the view set returned to the user, which is either $\mathcal{R}_t$ or a view pair that can synthesize the desired view by DIBR with the smallest view distance, i.e., $V_{s,l} = \sup_{v \in V_t \cap V_f} \{v < \mathcal{R}_t\}$, $V_{s,r} = \inf_{v \in V_t \cap V_f} \{v > \mathcal{R}_t\}$, and $|V_{s,l} - V_{s,r}| \leq D$. Note that the cardinalities of the three sets are no more than two, i.e., $|V_f|, |V_e|, |V_s| \leq 2$, implying that the proxy can fetch, evict, and returns at most two views at each instant.

For each state $\mathcal{S}_t = \{\mathcal{V}_t, \mathcal{R}_t\}$, the next state of the proxy $\mathcal{V}_{t+1}$ depends on $\mathcal{V}_t$ and $\mathcal{A}_t$, since $\mathcal{V}_{t+1} = \mathcal{V}_t \cup V_f - V_e$. Note that the distribution of $\mathcal{R}_{t+1}$ is probabilistically dependent on $\mathcal{R}_t$. In other words, each state is allowed to have a different arrival probability for each view in the model. If the current request is for view $\mathcal{R}_t = i$, the next one will be view $\mathcal{R}_{t+1} = j$ with probability q_{ij}[1]. Therefore, the state transition from $\mathcal{S}_t$ to $\mathcal{S}_{t+1}$ is jointly determined by the action $\mathcal{A}_t$ and the request transition probability q_{ij}.

[1] Note that the request transition probability can also be independent on the current request, i.e. $q_{1,j} = q_{2,j} = \cdots = q_{|V|,j} = \boldsymbol{q}_j$.

doi:10.6342/NTU201601753

After an action $\mathcal{A}_t = a$ is taken by the proxy at state $\mathcal{S}_t = s$, the cost $\mathcal{C}(\mathrm{s},\mathrm{a})$ incurred is described as follows.[2]

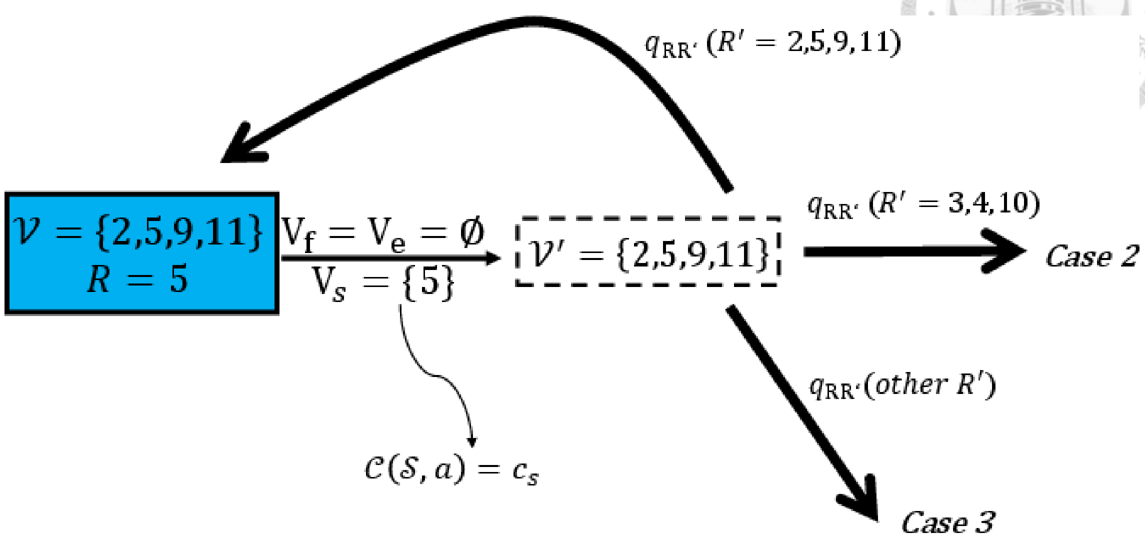
$$\mathcal{C}(\mathrm{s},\mathrm{a}) = c_m \mathbf{1}_{\{\mathcal{R}_t \notin \mathcal{V}_t \cup \mathcal{V}_t^s\}} + |V_f| c_f + |V_s| c_s + c_p, \tag{2.1}$$

where $\mathbf{1}_{\{x\}}$ is the indicator function (with the value as 1 or 0) to describe if x is true or false. The cost function is the same as the one introduced in the end of Chapter 2.1, where c_f and c_s denote the cost to fetch a view from the remote server and the cost to send a view to the user, respectively, c_m denote an penalty cost for a cache miss, and the view synthesis cost is $c_p = \alpha |V_{s,l} - V_{s,r}|$ with the synthesis unit cost α.

[2] Note that each parameter in the cost model needs to be normalized according to their weight in real situations.

doi:10.6342/NTU201601753

2.2.1 State Transition



$q_{RR'}\ (R' = 2,5,9,11)$
$\mathcal{V} = \{2,5,9,11\}$
$R = 5$
$V_f = V_e = \emptyset$
$V_s = \{5\}$
$\mathcal{V}' = \{2,5,9,11\}$
$q_{RR'}\ (R' = 3,4,10)$
Case 2
$q_{RR'}(other\ R')$
$\mathcal{C}(\mathcal{S}, a) = c_s$
Case 3

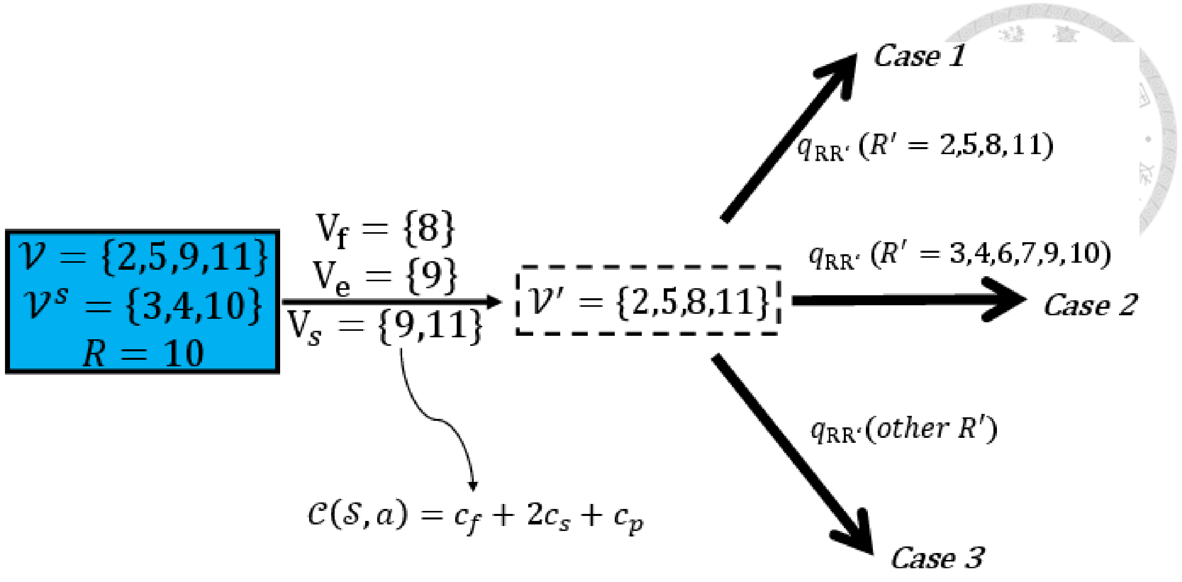
Fig. 2.2 An Example for State Transition (Case 1)

Next, we explain the state transition of MDP in detail with the following cases.

Case 1

If the requested view $\mathcal{R}_t$ is cached in the proxy, i.e., $\mathcal{R}_t \in \mathcal{V}_t$, the desired view will be directly returned to the user, that is, $V_f = V_e = \emptyset$ and $V_s = \{\mathcal{R}_t\}$. Thus $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_s$. No view replacement will be provoked. Fig. 2.2 presents an example for case 1. If $|V| = 16$, $\mathcal{V}_t = \{2,5,9,11\}$ and $\mathcal{R}_t = 5$, view 5 will be directly returned to the user. The dashed block in the middle of each figure denotes the cache state after an action, and the arrow at the right hand side represents the next request $\mathcal{R}'$ with probability $q_{\mathcal{R}\mathcal{R}'}$.

doi:10.6342/NTU201601753



$\mathcal{V} = \{2,5,9,11\}$
$\mathcal{V}^s = \{3,4,10\}$
$R = 10$
$V_f = \{8\}$
$V_e = \{9\}$
$V_s = \{9,11\}$
$\mathcal{V}' = \{2,5,8,11\}$
$\mathcal{C}(\mathcal{S}, a) = c_f + 2c_s + c_p$
$q_{RR'}\ (R' = 2,5,8,11)$
Case 1
$q_{RR'}\ (R' = 3,4,6,7,9,10)$
Case 2
$q_{RR'}(other\ R')$
Case 3

Fig. 2.3 An Example for State Transition (Case 2)

Case 2

If the requested view is not cached in the proxy but can be synthesized by two nearby cached views in $\mathcal{V}_t$, i.e., $\mathcal{R}_t \in \mathcal{V}_t^s$, the situation is further divided into many small sub-cases because the proxy can access, replace, and return one or two views, in order to minimize the total cost. For example, in Fig. 2.3, $\mathcal{V}_t = \{2,5,9,11\}$, $\mathcal{V}_t^s = \{3,4,10\}$ and $\mathcal{R}_t = 10$, the proxy can access view 8, replace view 9, and return view 9, 11 to the user. Note that the user need to receive two views, but the proxy can satisfy more requests in the future (i.e., views $2,5,8,11$ are directly cached, and views $3,4,6,7,9,10$ can be synthesized as well). Therefore, the cache state becomes $\mathcal{V}_{t+1} = \{2,5,8,11\}$ after the action. Since the proxy accesses one view and returns two vies to the user, the cost is $c_f + 2c_s + c_p$.

doi:10.6342/NTU201601753

Case 2-1: $\left|\boldsymbol{V_f}\right| = |\boldsymbol{V_e}| = \mathbf{0}, |\boldsymbol{V_s}| = \mathbf{2}$

No replacement is performed in this case. The elements in V_s are a view pair with smallest view distance to synthesize the desired view with DIBR, i.e., $V_{s,l} = \sup_{v \in V_t \cap V_f} \{v < \mathcal{R}_t\}$, $V_{s,r} = \inf_{v \in V_t \cap V_f} \{v > \mathcal{R}_t\}$, and $\left|V_{s,l} - V_{s,r}\right| \leq D$. $\mathcal{C}(\mathrm{s}, \mathrm{a}) = 2c_s + c_p$.

Case 2-2-1: $\left|\boldsymbol{V_f}\right| = |\boldsymbol{V_e}| = |\boldsymbol{V_s}| = \mathbf{1}$

In this case, the proxy acquires the desired view from the remote video server and returns the view to the user. Thus $V_f = V_s = \{\mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$ and $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_f + c_s$.

Case 2-2-2: $\left|\boldsymbol{V_f}\right| = |\boldsymbol{V_e}| = \mathbf{2}, |\boldsymbol{V_s}| = \mathbf{1}$

In this case, the proxy replaces two views and sends one view to the user. $V_f = \{\mathcal{R}_t, v | v \in V - \mathcal{V}_t - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$, $V_s = \{\mathcal{R}_t\}$, and $\mathcal{C}(\mathrm{s}, \mathrm{a}) = 2c_f + c_s$.

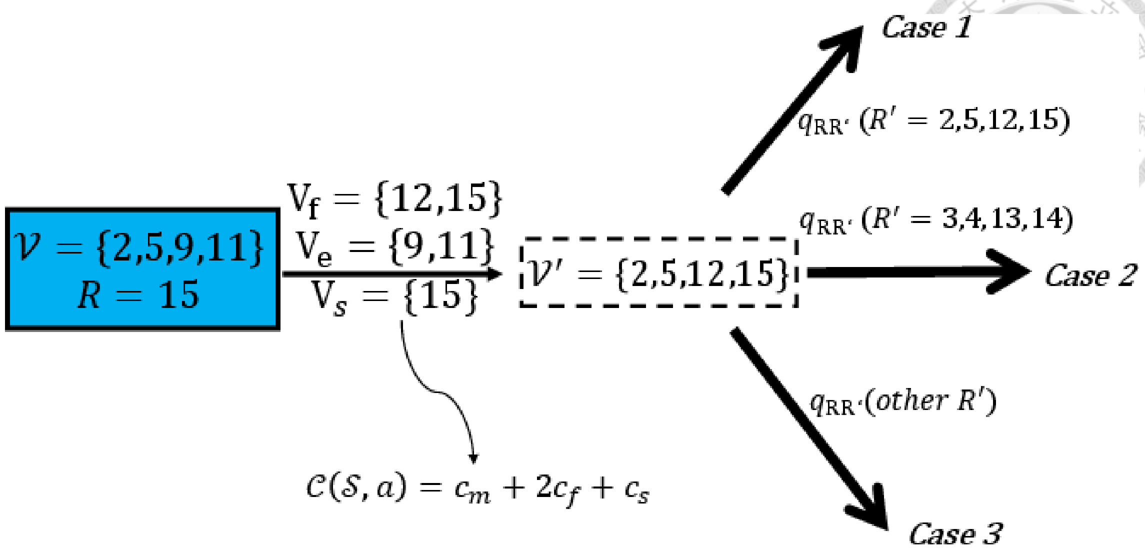
Case 2-3-1: $\left|\boldsymbol{V_f}\right| = |\boldsymbol{V_e}| = \mathbf{1}, |\boldsymbol{V_s}| = \mathbf{2}$

The proxy replaces a view for a higher hit probability in the future and sends a view pair to the user. $V_f = \{v | v \in V - \mathcal{V}_t - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$, and the elements in V_s are the same as in Case 2-1. $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_f + 2c_s + c_p$.

Case 2-3-2: $\left|\boldsymbol{V_f}\right| = |\boldsymbol{V_e}| = |\boldsymbol{V_s}| = \mathbf{2}$

The proxy acquires two views for a higher hit probability in the future and sends a view pair to the user. $V_f = \{v_1, v_2 | v_1, v_2 \in V - \mathcal{V}_t - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$, and the elements in V_s are the same as in Case 2-1. $\mathcal{C}(\mathrm{s}, \mathrm{a}) = 2c_f + 2c_s + c_p$.

doi:10.6342/NTU201601753



$\mathcal{V} = \{2,5,9,11\}$
$R = 15$
$V_f = \{12,15\}$
$V_e = \{9,11\}$
$V_s = \{15\}$
$\mathcal{V}' = \{2,5,12,15\}$
$\mathcal{C}(\mathcal{S}, a) = c_m + 2c_f + c_s$
$q_{RR'}\ (R' = 2,5,12,15)$
Case 1
$q_{RR'}\ (R' = 3,4,13,14)$
Case 2
$q_{RR'}(other\ R')$
Case 3

Fig. 2.4 An Example for State Transition (Case 3)

Case 3

If the requested view $\mathcal{R}_t$ cannot be satisfied by the proxy, a cache miss happens, i.e., $\mathcal{R}_t \notin \mathcal{V}_t \cup \mathcal{V}_t^s$. In this case, the proxy will access either the view $\mathcal{R}_t$, a neighbor view of $\mathcal{R}_t$, or a pair of views which can synthesize $\mathcal{R}_t$ from the remote video server. In addition, it can replace and return one or two views to the users to minimize the total cost. Fig. 2.4 presents an example for case 3. When $\mathcal{R}_t = 15$, the proxy may access two views 12, 15 to replace view 9 and 11, and deliver view 15 to the user with the corresponding cost $c_m + 2c_f + c_s$.

Case 3-1-1: $|V_f| = |V_e| = |V_s| = 1$

In this case, the proxy requests the missed view to serve the user. Thus $V_f = V_s = \{\mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$ and $\mathcal{C}(\mathrm{s,a}) = c_m + c_f + c_s$.

doi:10.6342/NTU201601753

Case 3-1-2: $|\boldsymbol{V_f}| = |\boldsymbol{V_e}| = \mathbf{2}, |\boldsymbol{V_s}| = \mathbf{1}$

The proxy requests two views from the content provider, including the missed view, and send the desired view to the user. $V_f = \{\mathcal{R}_t, v | v \in V - \mathcal{V}_t - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$, $V_s = \{\mathcal{R}_t\}$, and $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_m + 2c_f + c_s$.

Case 3-2-1: $|\boldsymbol{V_f}| = |\boldsymbol{V_e}| = \mathbf{1}, |\boldsymbol{V_s}| = \mathbf{2}$

The proxy requests the neighbor view of the missed view to synthesize the missed view. Therefore the server sends two views to the user. $V_f = \{v | v \in [\mathcal{R}_t - (D-1), \mathcal{R}_t + (D-1)] - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$ and the elements in V_s is the same as in Case 2-1. Thus $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_m + c_f + 2c_s + c_p$.

Case 3-2-2: $|\boldsymbol{V_f}| = |\boldsymbol{V_e}| = |\boldsymbol{V_s}| = \mathbf{2}$

The proxy may request a view pair to synthesize the missed view, or to achieve a higher hit probability in the future. $V_f = \{v_1, v_2 | v_1, v_2 \in V - \mathcal{V}_t - \mathcal{R}_t\}$, $V_e \subseteq V_t \cup V_f$, and the elements in V_s is the same as in Case 2-1. Thus $\mathcal{C}(\mathrm{s}, \mathrm{a}) = c_m + 2c_f + 2c_s + c_p$.

doi:10.6342/NTU201601753

2.2.2 Optimal Policy

The action $\mathcal{A}_t = a$ for request $\mathcal{R}_t$ is described by $\mathcal{A}_t = \pi(\mathcal{S}_t = s)$ for a policy $\pi: \mathcal{S} \to \mathcal{A}$. For MDP, the expected average cost over an infinite horizon is

$$\lim_{T\to\infty} \mathbb{E}^{\pi}\{\tfrac{1}{T+1}\textstyle\sum_{t=0}^{T} \mathcal{C}\big(\mathcal{S}_t, \pi(\mathcal{S}_t)\big)\},$$

(2.2)

Our goal is to find an optimal policy π^* such that the above equation is minimized. That is,

$$\pi^* = arg \min_{\pi} \lim_{T\to\infty} \mathbb{E}^{\pi}\{\tfrac{1}{T+1}\textstyle\sum_{t=0}^{T} \mathcal{C}\big(\mathcal{S}_t, \pi(\mathcal{S}_t)\big)\}.$$

(2.3)

However, directly solving equation (2.3) is computationally intractable. Suppose a value function $V(s)$ is defined for each state $s = (v, r)$ to describe the long-term average cost when the beginning state is s. Therefore, the optimal solution can be achieved by solving the equivalent Bellman's equation [18].

Lemma 1 *(Equivalent Bellman's Equation). If a scalar* β *and a value function* $V(s)$ *satisfy the Bellman's equation for problem VSCO, written as* $\forall s \in \mathcal{S}$,

$$\beta + V(s) = \min_{a\in A}\left\{\mathcal{C}(s,a) + \sum_{s'} p_{ss'}\, V(s'|s,a)\right\}$$

$$= \min_{a\in A}\left\{\mathcal{C}(s,a) + \sum_{r'\in V} q_{rr'}\, V(v', r'|v, r, a)\right\}$$

(2.4)

where $C(s,a)$ *is the instantaneous cost under the state* s *and action* a*, then* β *is the optimal average cost:*

$$\beta = \min_{\pi} \bar{C}(\pi)$$

Moreover, $\pi^* = arg\min_{a\in A}\{C(s,a) + \sum_{r'} q_{rr'} V(v',r'|v,r,a)\}$ *is the optimal policy.*

After this transformation, we are able to choose the best action a leading to the smallest one-period cost $C(s,a)$, plus the expected cost value of landing at state s'.

The Bellman's equation can be solved with value iteration [18] or policy iteration [18], which is a general solution to calculate the optimal utility function iteratively. We demonstrate the procedure of value iteration to compute the value function $V(s)$ in Algorithm 1. Line 1 is for initialization. Line 2 to 11 compute the value function iteratively. Line 5 updates the value function for all states in each iteration. Note that the value function may converge to a large value; therefore, we choose an arbitrary state s_r as the reference state. In each iteration, the value function $V^{\hat{t}}(s)$ is replaced by a relative value to that of the reference state (Line 8).

doi:10.6342/NTU201601753

```
Algorithm Value Iteration
Require: Cost function $\mathcal{C}(s,a)$ and request distribution $q_{i,j}$.
Ensure: The optimal value function $V(s)$, $\forall s \in \mathcal{S}$
 1: $\hat{t} = 0$, $V^{\hat{t}}(s) = 0$, $\forall s \in \mathcal{S}$;
 2: while Not Converge do
 3:     $\hat{t} = \hat{t} + 1$;
 4:     for all $s \in \mathcal{S}$ do
 5:         $V^{\hat{t}}(s) = \min_{a \in \mathcal{A}} \{\mathcal{C}(s,a) + \sum_{r' \in V} q_{rr'} V^{\hat{t}-1}(v', r' | v, r, a)\}$;
 6:     end for
 7:     for all $s \in \mathcal{S}$ and $s \neq s_r$ do
 8:         $V^{\hat{t}}(s) = V^{\hat{t}}(s) - V^{\hat{t}}(s_r)$;
 9:     end for
10:     $V^{\hat{t}}(s_r) = 0$;
11: end while
12: return The optimal value function $V(s)$.
```

Algorithm 1 Value Iteration

The number of global states is determined by $\mathcal{V}_t$ and $\mathcal{R}_t$, therefore there are $C_N^V \times N \approx O(V^{N+1})$ states. The number of actions for states in case 1 is only one. However, for states in case 2 and 3, there are at most $C_1^{V-N} C_1^{N+1} + C_2^{V-N} C_2^{N+2} \approx O((VN)^2)$ actions, since the proxy can replace one view or two views at a time. Therefore, the total number of state-action pair is of $O(V^{N+2}N^2)$, which is an exponential function. Note that the real feasible action will all be determined by the cost components in the cost function, e.g. when c_f is large, the proxy will not replace two view at a time. Thus, the number of state-action pair will be smaller.

The MDP is designed to find the optimal policy for small cases. In the next section,

doi:10.6342/NTU201601753

we design an effective and efficient heuristic algorithm to solve the problem for large cases.

doi:10.6342/NTU201601753

Chapter 3 Heuristic Algorithm

3.1 Algorithm EVEA

Most conventional cache replacement algorithms exploit the request recency [6] or frequency [25], such as the Least Recently Used (LRU-based) policy and Least Frequently Used (LFU-based) policy, respectively. Correlations of nearby views have not been leveraged in those algorithms to support multi-view 3D videos. In this section, by contrast, we aim to extract the ideas behind the proposed MDP to design an effective and efficient heuristic algorithm. Our algorithm examines not only the preference/frequency of user requests but also the correlation between nearby views to boost the number cache hits and thus reduce the total cost, whereas a synthesize cost is also incorporate to ensure the video quality.

In the previous section, we have formulated the problem by MDP, which can be solved by value iteration or by linear programming. However, both methods need to evaluate the value function for each state and enumerate each action, and the above processes are computation intensive for large cases. Hence, we design a new algorithm, named *Efficient View Exploration Algorithm* (EVEA) by 1) simplifying the computation of the value function to consider only one-stage cost and, 2) shrinking the action space of

doi:10.6342/NTU201601753

each V_f, V_e. Note that in MDP, the proxy can fetch and evict at most two views to carefully examine all possible cases, and it may create a large action space when the number of view is large. In contrast, here we confine the proxy to fetch and replace at most one view at an instant to effectively reduce the action space.

More specifically, when the cache state is $\mathcal{V}_t$ and the current request $\mathcal{R}_t = r$, we consider the one-period cost and derive the expected miss probability for the next request[3]. The rationale behind EVEA is to minimize the expected miss probability of the next state as well as cost induced from the current request, including the fetch cost, transmission cost, and synthesize penalty. In other words, we aim to find the best (V_f, V_e, V_s) such that

$$\mathcal{C}(\mathrm{s},\mathrm{a}) = c_m \mathbf{1}_{\{r \notin \mathcal{V}_t \cup \mathcal{V}_t^s\}} + |V_f| c_f + |V_s| c_s + c_p + \beta \sum_{r'} q_{rr'} \mathbf{1}_{\{r' \notin \mathcal{V}_{t+1} \cup \mathcal{V}_{t+1}^s\}} \quad (3.1)$$

is minimized, where β is a weight factor and $\mathcal{V}_{t+1} = \mathcal{V}_t \cup V_f - V_e$. Therefore, when a new request arrives, we search all the possible combinations of the fetched, evicted, and delivered views to minimize the expected miss probability and total cost.

In the following, we consider the three cases in MDP for the algorithm design. First, when the cache can satisfy the user's request by directly transmitting the desired

[3] Note that the algorithm can be simply extended to consider the multi-stage cost and miss probability. Nevertheless, the complexity increases as analyzed is in Chapter 3.2.

doi:10.6342/NTU201601753

view, i.e., case 1 in MDP model, we do not replace any views in this case, the same as the MDP policy. Second, when the request can be synthesized by two views in the proxy, as the case 2, we examine every possible V_f and V_e to find the best (V_f, V_e) pair in order to minimize (3.1). Note that it is necessary to carefully examine the view correlation to estimate the average miss probability of the next state. In addition, the synthesized penalty can be avoided when we directly fetch the desired view. For example, if $|V| = 16$, $\mathcal{V}_t = \{2,5,9,11\}$ and $\mathcal{R}_t = 12$, the proxy can only access and replace one view at each time, e.g., access view 12 and replace view 11. Since the number of fetched view and evicted view are both limited to one, it is expected that the action space can be effectively reduced.

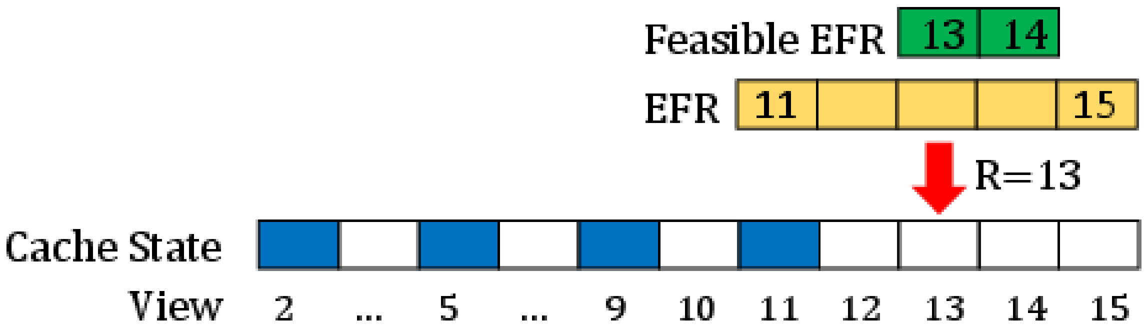


Fig. 3.1 An Example for *Effective Fetch Range*

The third case represents that the proxy is not able to satisfy the user request, and a cache miss thereby occurs. In this case, it is crucial to carefully fetch the view from the remote video server. Note that the feasible fetched views lie in the *Effective Fetch Range* (EFR), which is $[r - D + 1, r + D - 1]$, because only the views in this range can be

doi:10.6342/NTU201601753

employed for the request r. Consider the example in Fig. 3.1, when $|V| = 16$, $\mathcal{V}_t = \{2,5,9,11\}$ and $\mathcal{R}_t = 13$ and $D = 3$, the *effective fetch range* for view 13 is $[11,15]$, and the feasible fetched view is 13 and 14, because the proxy can serve the user by transmitting view 13 directly, or synthesize it by view 11 and view 14. After we identify the set of feasible fetch views, the algorithm carefully examines every view V_f in feasible *effective fetch range* and V_e as in the previous case.

After we acquire V_f and V_e, the corresponding V_s can be obtained as well. In addition, the cost induced in this state and operations can be derived accordingly by (2.1).

doi:10.6342/NTU201601753

```
Algorithm EVEA
Require: cache state $\mathcal{V}_t$ and request $r$.
Ensure: view to be fetched $V_f$, view to be evicted $V_e$, views to be sent $V_s$, and minimum induced
    cost $\mathcal{C}$.
 1: if $r$ in $\mathcal{V}_t$ then
 2:     $V_f^* = V_e^* = \emptyset$, $V_s^* = r$.
 3: else if $r$ in $\mathcal{V}_t^s$ then
 4:     for all $V_f$ in $V - \mathcal{V}_t$, $V_e$ in $\mathcal{V}_t \cup V_f$ do
 5:         Calculate (3.1) with each $V_f$ and $V_e$;
 6:     end for
 7:     $(V_f^*, V_e^*, V_s^*) = arg\min$ (3.1);
 8: else
 9:     Effective Fetch Range(EFR)= $[r - D + 1, r + D - 1]$;
10:     for all $v$ in EFR do
11:         check if $\mathcal{V}_t \cup v$ is feasible;
12:     end for
13:     for all $V_f$ in feasible EFR, $V_e$ in $\mathcal{V}_t \cup V_f$ do
14:         Calculate (3.1) with each $V_f$ and $V_e$;
15:     end for
16:     $(V_f^*, V_e^*, V_s^*) = arg\min$ (3.1);
17: end if
18: return $(V_f^*, V_e^*, V_s^*)$ and induced cost $\mathcal{C}$.
```

Algorithm 2 EVEA

Algorithm 2 is the pseudo code of EVEA. Line 1 to 2 is the case 1, line 3 to 7 is the case 2, line 8 to 17 is the case 3, and finally, the algorithm returns the view to be fetched, the view to be evicted, and the views to be sent with associate cost in line 18.

doi:10.6342/NTU201601753

3.2 Complexity Analysis

For case 1, the complexity is $O(1)$. The algorithm searches $O(|V| - N)$ V_f and (N) V_e, and calculates $O(|V|)$ requests to find the expected miss probability. Therefore, the complexity in case 2 is $O(N|V|^2 - N^2|V|) = O(N|V|^2)$, since $N < |V|$ typically. For case 3, the complexity of *effective fetch range* is $O(2D)$, and the algorithm then enumerates $O(2D)$ possible V_f and $O(N)$ possible V_e and sums up $O(|V|)$ requests when deriving the expected miss probability. Thus, the total complexity of case 3 is $O(DN|V|)$. The overall complexity of our algorithm is $O(N|V|(D + |V|)) = O(N|V|^2)$, where $D < |V|$ typically. For the multiple-stage cost model mentioned early in this section, the time complexity will be $O(N^2|V|^4)$ for the two-stage model and $O(N^3|V|^6)$ for the three-stage model.

doi:10.6342/NTU201601753

3.3 Extension of Multi-Video Scenario

To support multiple videos, a promising way is to add a superscript to each notation to represent the index of the video. For M videos, the global state of MDP now becomes $\mathcal{S}_t = \{\mathcal{V}_t^1, \mathcal{V}_t^2, \dots, \mathcal{V}_t^M, R_t^i\}$, to summarize the overall cache status of video $1,2, \dots M$ with the request for video i and view $\mathcal{R}_t$. The cache size constraint becomes $|\mathcal{V}_t^1| + |\mathcal{V}_t^2| + \cdots + |\mathcal{V}_t^M| \leq N$. Moreover, the action becomes $\{V_f, V_e, V_s\} = \bigcup_{i=1}^{M} \{V_f^i, V_e^i, V_s^i\}$ to represent the union of view sets fetched, evicted, and delivered for video i. Note that for each request R_t^i, V_s is selected only from $\mathcal{V}_t^i \cup V_f^i$. Nevertheless, V_f and V_e may be selected from another video j, $j \neq i$. Because the popularity of each video is different, some popular videos and views may be *prefetched*, whereas other videos and views may be replaced due to low popularity.

doi:10.6342/NTU201601753

Chapter 4 Performance Evaluation

In this section, we first describe the simulation settings and then evaluate EVEA with other cache replacement policies in different scenarios. Finally, we compare the MDP solution with EVEA and other algorithms.

Variable	Default Setting
Number of videos	5000
Number of views in a video	$\|V\| = 16$
Size of a view	30MB
Video popularity distribution	Zipf, 0.8
View popularity distribution	Zipf, 0.8 and Uniform
Number of requests	50000
DIBR synthesize range	$D = 3$
Cache size	450GB
(c_m, c_f, c_s, α)	$(10, 3, 2, 0.3)$

Table 4.1 Simulation Settings

doi:10.6342/NTU201601753

4.1 Simulation Settings

Table 4.1 summarizes the parameters in the simulation. We adopt the setting of the existing multi-view 3D videos with $|V| = 16$ [2], and the default value of the DIBR synthesize range $D = 3$ [17]. To the best of our knowledge, there has been no related work on caching multi-view 3D videos. Therefore, we compare the proposed MDP model and EVEA with two widely used categories of algorithms, LRU and LFU. The default settings are $c_m = 10, c_f = 3, c_s = 2$ and $\alpha = 0.3$. Note that $c_f > c_s$ in the simulation since a larger delay will be incurred for a remote video access.

Without losing the generality, we fix the size of all views for a video to be 30MB [26], which is about 3 minutes playback time, and there are 5000 multi-view videos with 16 views in the simulation (i.e., 80000 contents). Note that the proxy caches the whole 30MB data [10],[26] of each selected view. The video popularity follows the Zipf distribution [27] as follows,

$$f(i, z, N) = \frac{1/_{i^z}}{\sum_{n=1}^{N} 1/_{n^z}},$$

where i is the preference rank of an object, z is the zipf factor, and N is the total number of objects. Here we let $= 0.8$ [27]. The view angle popularity follows 1) Zipf distribution and 2) Uniform distribution, denoted by Z and U respectively. In the simulation, the results with different cache sizes, quality constraint D, and numbers of

doi:10.6342/NTU201601753

views are evaluated.

4.2 Simulation Results

4.2.1 Scenario 1: Cache Size

We change the cache size from 150GB to 750GB[4] to observe the hit rate and the average cost induced from 50000 requests in Fig. 4.1 and Fig. 4.2. The performance of LRU and LFU is quite similar. As the cache size grows, the performance of the three algorithms improves. However, the gap between EVEA and LRU (LFU) remains the same, because the proposed algorithm always considers the view correlation and examine the expected miss probability. When more views can be cached, EVEA has larger flexibility to store more proper views, and it always outperforms the other two schemes.

[4] When the cache size is set to 300GB, about 12.5% of the views can be cached in the simulation.

doi:10.6342/NTU201601753

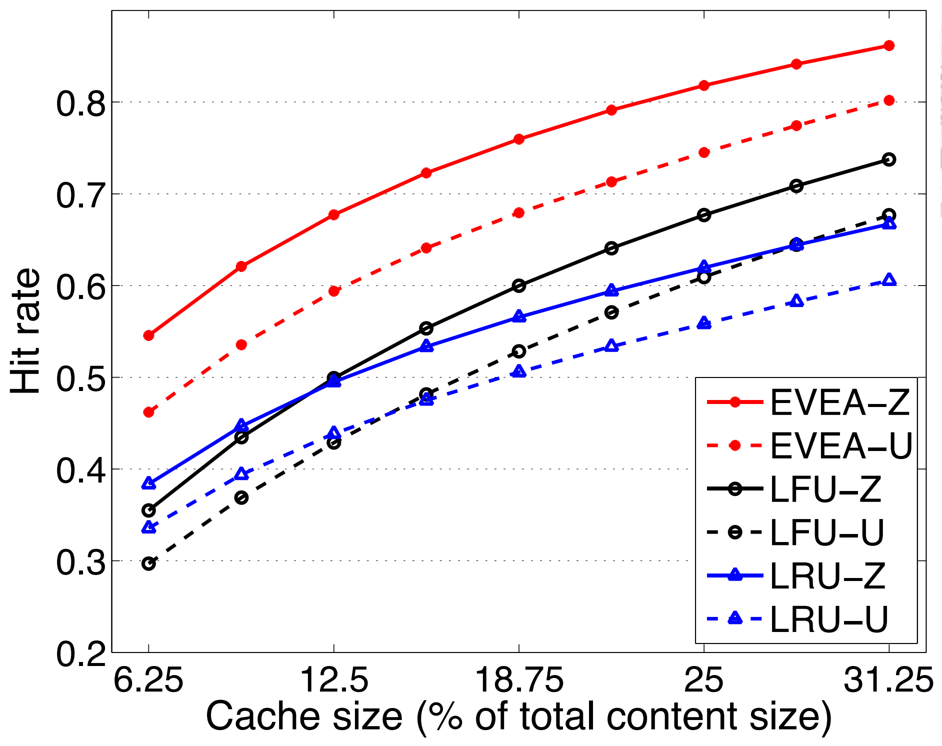


Fig. 4.1 Cache Size vs. Hit Rate

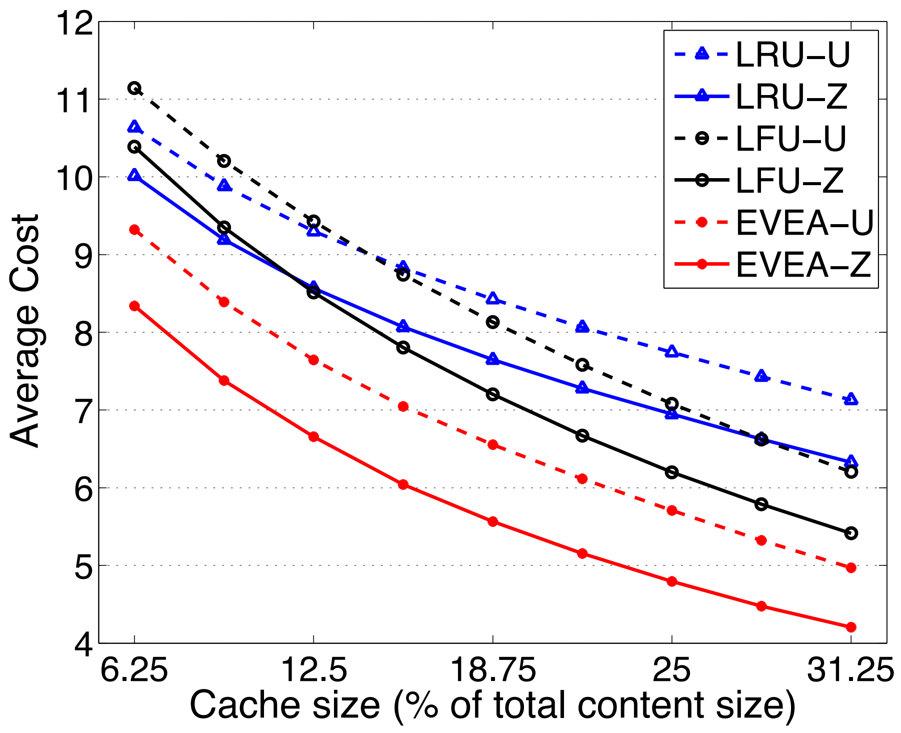


Fig. 4.2 Cache Size vs. Average Cost

doi:10.6342/NTU201601753

4.2.2 Scenario 2: Synthesize Range

Fig. 4.3 and Fig. 4.4 present the impact of different synthesize ranges D on the hit rate and the percentage of users receiving two views, respectively. The default cache size is set to 450GB. When $D = 1$, (i.e. no DIBR), EVEA acquires a better hit rate, because it carefully derives the miss probability for the cases with and without DIBR. As D grows, more and more users can be satisfied because it becomes easier to synthesize a view. However in Fig. 4.4, note that the percentage of users receiving two views in EVEA first rises and then declines. For a small D, users are not able to synthesize their desired view from two distant views. Nevertheless, since EVEA selects the views for caching according to the synthesis cost $c_p = \alpha\left|V_{s,l} - V_{s,r}\right|$, the proxy will be more inclined to serve the users by directly transmitting the view or synthesizing the view from a close view pair even for a large D.

doi:10.6342/NTU201601753

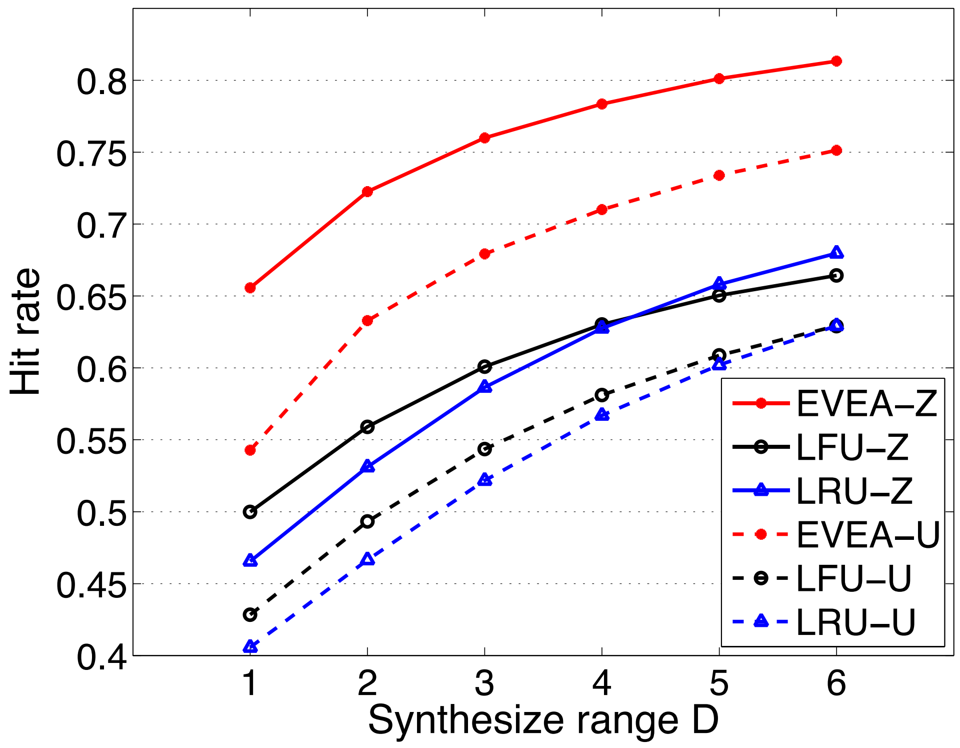


Fig. 4.3 Synthesize Range vs. Hit Rate

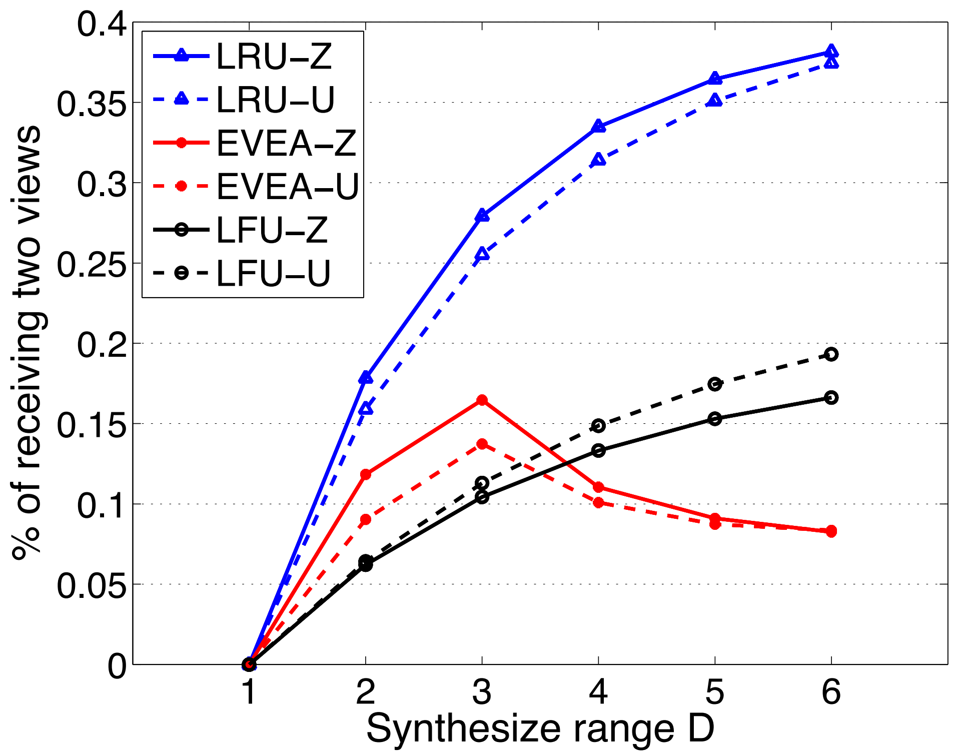


Fig. 4.4 Synthesize Range vs. Percentage of Users Receiving Two Views

doi:10.6342/NTU201601753

4.2.3 Scenario 3: Number of Views

When the number of views is small, the proxy can cache most popular views in a video, resulting a high hit rate and a low average cost, as shown in Fig. 4.5 and Fig. 4.6. The gap in Fig. 4.5 between EVEA and the other two baseline algorithms becomes more significant. When a smaller percentage of views are cached (i.e., more views in a video), cache misses occur more often if the view correlation is not carefully considered. As a result, when there are 32 views, our design can improve at least 30% of the hit rate compared with LRU and LFU in both distributions. The total cost can be effectively reduced as well.

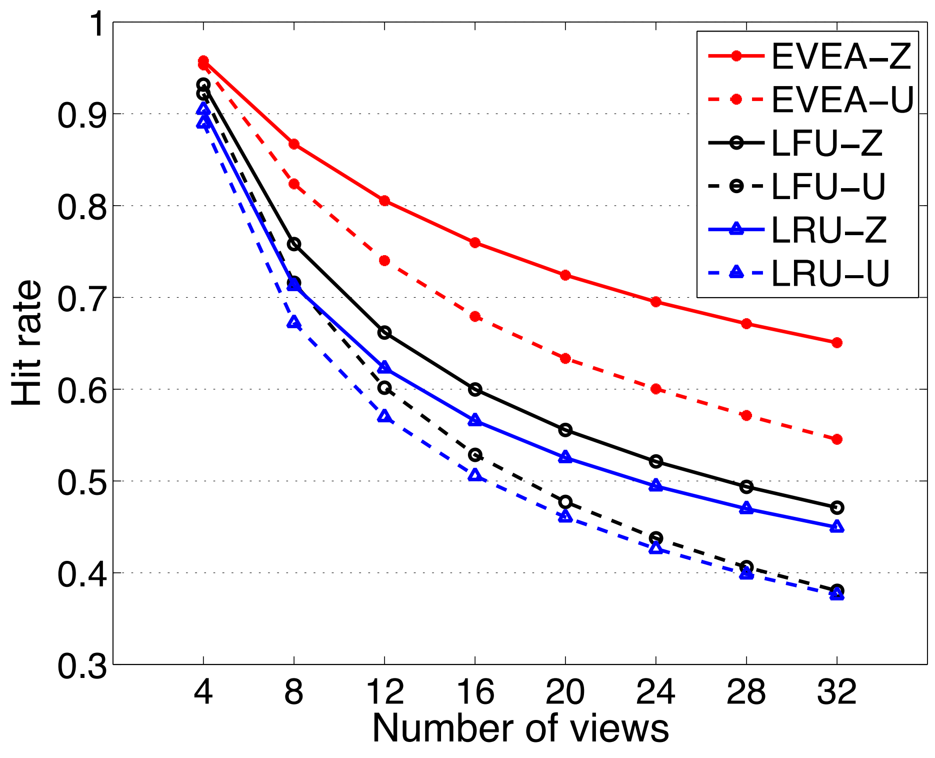


Fig. 4.5 Number of Views vs. Hit Rate

doi:10.6342/NTU201601753

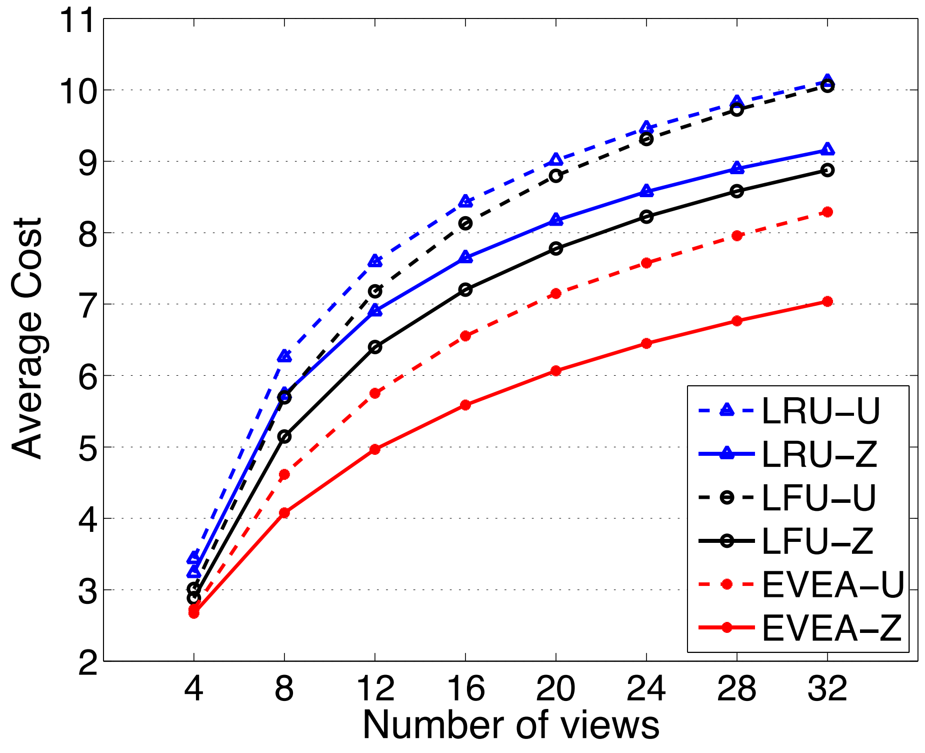


Fig. 4.6　Number of Views vs. Average Cost

4.2.4　Scenario 4: Distribution of view popularity

Fig. 4.1 to Fig. 4.6 also summarize the results with different distributions on video and view popularity. The results indicate that the cache performance improves as the user requests are more concentrated in only a few views. Moreover, our proposed algorithm performs better in both distributions, implying that it is able to support a wide ranged of applications with different preferences.

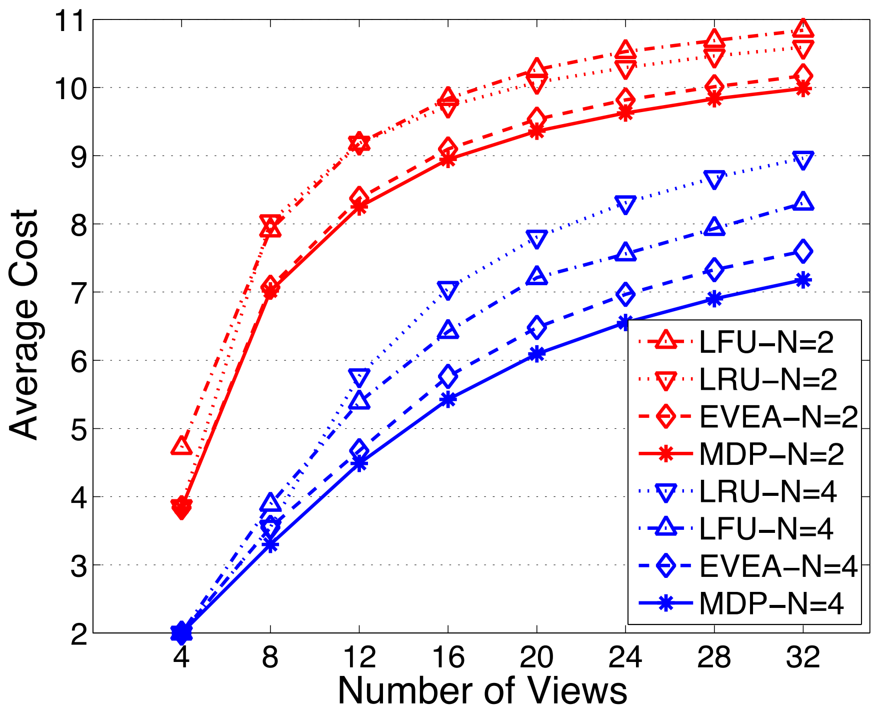
doi:10.6342/NTU201601753

4.2.5 Scenario 5: Comparison between MDP and EVEA

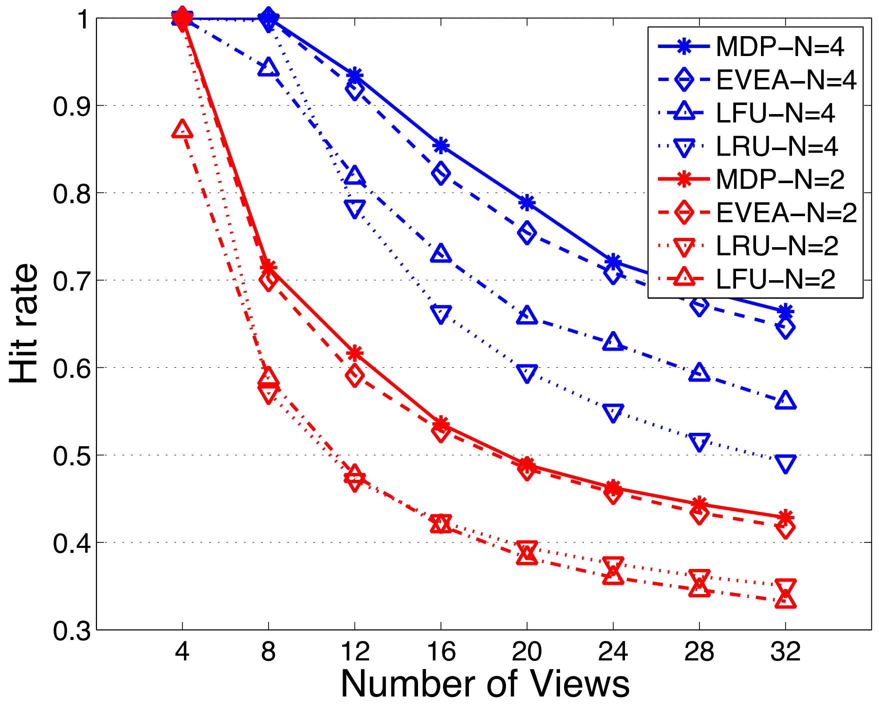
Fig. 4.7 (a), (b) show the comparison between MDP and EVEA. Here smaller cases considered a single video with $|V| = 4$ to $|V| = 32$, and $N = 2, 4$, because there are about a million states and over a billion state-action pairs when $|V| = 32$ and $N = 4$. The simulation result shows that the performance of EVEA nearly approach MDP, which is the optimal solution of VSCO.

Fig. 4.8 and Fig. 4.9 show the impact on average cost and the percentage of users receiving two views with different synthesis unit cost α and the size of cache, with $|V| = 16$ and $D = 3$ and 4. The cost rises as the synthesis factor grows. Nevertheless, the curves of MDP and EVEA stop rising when $\alpha \geq 0.4$ and $\alpha \geq 0.3$ for $D = 3$ and 4, respectivly. The same trend is observed in Fig. 4.9. When the synthesis unit cost is large enough, MDP and EVEA will return views directly without synthesis since fetching an extra view for delivery does not produce more cost, compared with delivering two nearby views. Therefore, a larger synthesize range D should match a smaller synthesis unit cost α. Also, MDP and EVEA can effectively reduce the average cost compared with LRU and LFU.

doi:10.6342/NTU201601753



(a) Number of Views vs. Average Cost



(b) Number of Views vs. Hit Rate

Fig. 4.7 MDP vs. EVEA

doi:10.6342/NTU201601753

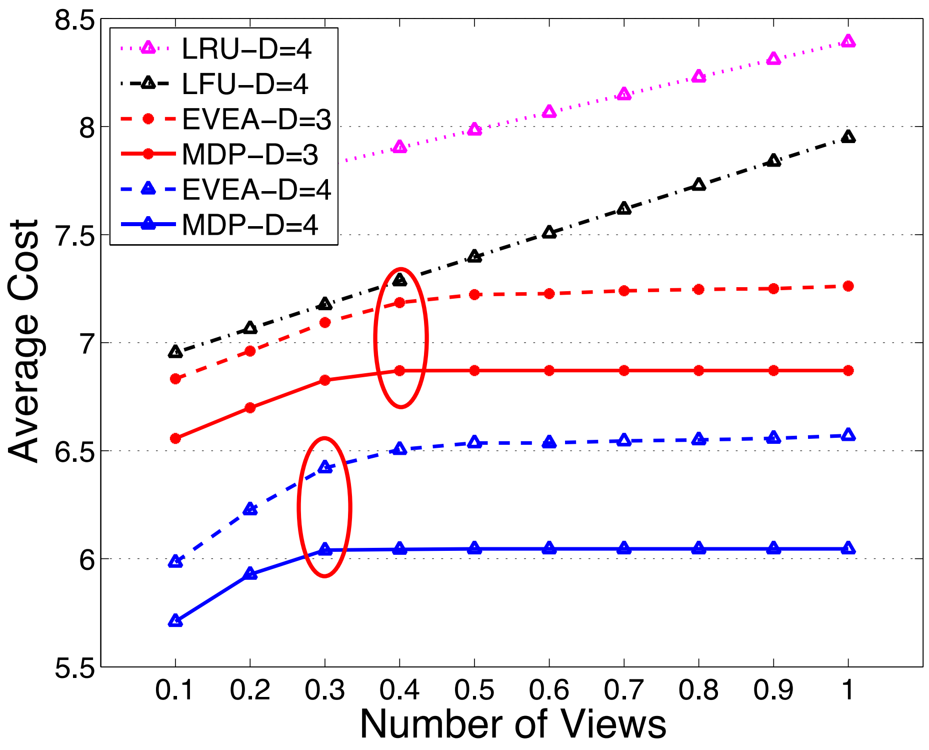


Fig. 4.8 Synthesis Unit Cost vs. Average Cost

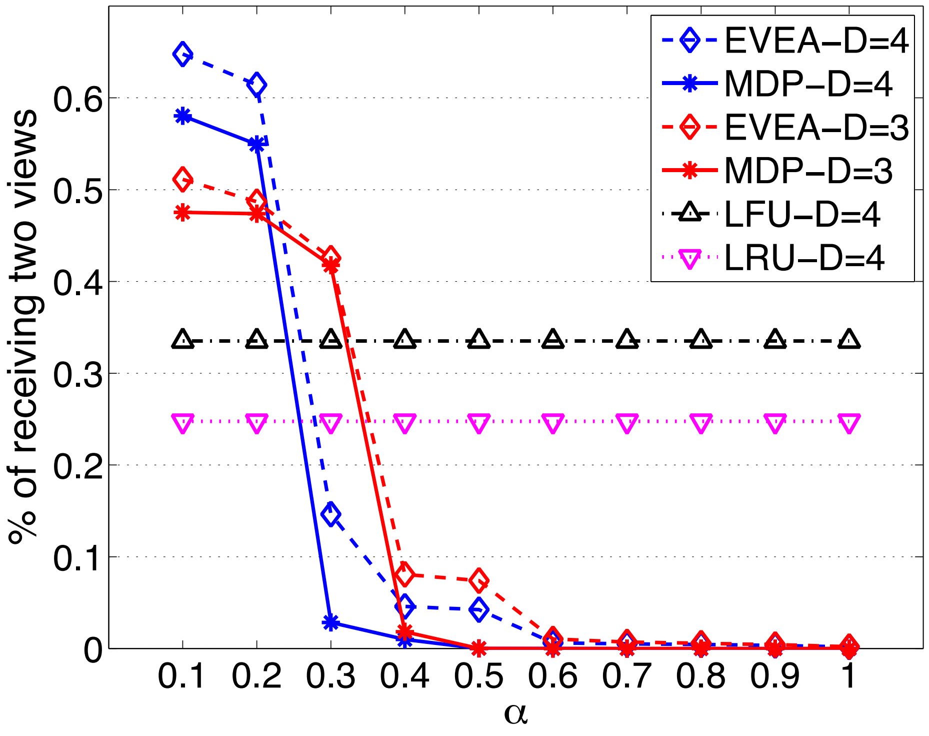


Fig. 4.9 Synthesis Unit Cost vs. Percentage of Users Receiving Two Views

doi:10.6342/NTU201601753

Chapter 5 Conclusion and Future Works

With the emergence of 3D video and VR device, this thesis investigated efficient caching for multi-view 3D videos. We propose to leverage DIBR, one of promising view rendering algorithm in multi-view 3D, for effectively increasing the cache hit rate and reducing the total cost in a proxy. We first formulate a new optimization problem (i.e., VSCO) and then derive the optimal policy based on MDP. Afterward, an effective and efficient algorithm EVEA is presented to support cache replace for large instances. Simulation results manifest that our proposed approaches can gain at least 30% hit rate and reduce 30% costs against traditional renowned cache replacement policies.

Due to the curse of dimensionality, solving MDP becomes more impractical for large cases. That is, the optimal solution of single-video scenario is solved while the performance of multi-view scenario is not guaranteed. Therefore, in the future works, we can solve VSCO by approximate MDP for large cases (i.e., multi-video), meanwhile providing the performance bound (i.e., approximation ratio).

doi:10.6342/NTU201601753

REFERENCE

[1] Cisco, Virtual Networking Index, “Global mobile data traffic forecast update, 2015-2020,” white paper, Feb. 2016, available online at http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking- index-vni/mobile-white-paper-c11-520862.html.

[2] I. Feldmann, M. Mueller, F. Zilly, R. Tanger, K. Mueller, A. Smolic, P. Kauff, and T. Wie-gand, “HHI Test Material for 3-D Video,” Proc. 84th Meet. ISO/IEC JTC1/SC29/WG11, document M15413, Apr. 2008.

[3] C. Fehn, “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV,” in Proceedings of SPIE Conference Stereoscopic Displays and Virtual Reality Systems XI, 2004, pp. 93–104.

[4] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, “Femtocaching: Wireless video content delivery through distributed caching helpers,” in INFOCOM, 2012 Proceedings IEEE, March 2012, pp. 1107–1115.

[5] K.-Y. Wong, “Web cache replacement policies: a pragmatic approach,” IEEE Network, vol. 20, no. 1, pp. 28–34, Jan 2006.

[6] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, and E. A. Fox, “Caching

doi:10.6342/NTU201601753

proxies: Limitations and potentials," in WWW-4, Boston Conference, 1995.

[7] M. C. Lee, F. Y. Leu, and Y. P. Chen, "Cache Replacement Algorithms for YouTube," in Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, May 2014, pp. 743–750.

[8] A. Liu and V. K. N. Lau, "Cache-Enabled Opportunistic Cooperative MIMO for Video Streaming in Wireless Systems," IEEE Transactions on Signal Processing, vol. 62, no. 2, pp. 390–402, Jan 2014.

[9] H. Ahlehagh and S. Dey, "Hierarchical video caching in wireless cloud: Approaches and algorithms," in Communications (ICC), 2012 IEEE International Conference on, June 2012, pp. 7082–7087.

[10] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in Wireless Communications and Networking Conference (WCNC), 2012 IEEE, April 2012, pp. 2276– 2281.

[11] H. Ahlehagh and S. Dey, "Adaptive bit rate capable video caching and scheduling," in Wireless Communications and Networking Conference (WCNC), 2013 IEEE, April 2013, pp. 1357–1362.

[12] H. Ahlehagh and S. Dey, "Video-Aware Scheduling and Caching in the Radio Access Network," IEEE/ACM Transactions on Networking, vol. 22, no. 5, Oct

doi:10.6342/NTU201601753

2014.

[13] A. Lobzhanidze and W. Zeng, “Proactive caching of online video by mining mainstream media,” in Multimedia and Expo (ICME), 2013 IEEE International Conference on, July 2013, pp. 1–6.

[14] F. Shao, G. Jiang, M. Yu, K. Chen, and Y.-S. Ho, “Asymmetric Coding of Multi-View Video plus Depth based 3D Video for View Rendering,” Multimedia, IEEE Trans. on, vol. 14, no. 1, pp. 157–167, Feb. 2012.

[15] G. Cheung, V. Velisavljevic, and A. Ortega, “On dependent bit allocation for multiview image coding with depth-image-based rendering,” IEEE Transactions on Image Processing, vol. 20, no. 11, pp. 3179–3194, Nov 2011.

[16] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand, “Depth image-based rendering with advanced texture synthesis for 3-d video,” IEEE Transactions on Multimedia, vol. 13, no. 3, pp. 453–465, June 2011.

[17] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, “View Generation with 3D Warping Using Depth Information for FTV,” Signal Processing: Image Communication, vol. 24, no. 1-2, pp. 65–72, Jan. 2009.

[18] D. Heyman and M. Sobel, Stochastic Models in Operations Research, Volume II: Stochastic Optimization. New York: McGraw-Hill, 1984.

doi:10.6342/NTU201601753

[19] W. Tu, E. Steinbach, M. Muhammad, and X. Li, “Proxy caching for video-on-demand using flexible starting point selection,” IEEE Transactions on Multimedia, vol. 11, no. 4, pp. 716–729, June 2009.

[20] J. Z. Wang and P. S. Yu, “Fragmental proxy caching for streaming multimedia objects,” IEEE Transactions on Multimedia, vol. 9, no. 1, pp. 147–156, Jan 2007.

[21] A. Hamza and M. Hefeeda, “Energy-Efficient Multicasting of Multiview 3D Videos to Mo- bile Devices,” ACM Trans. on Multimedia Computing, Communications, and Applications, vol. 8, no. 3s, pp. 45:1–45:25, Sep. 2012.

[22] Y. Aksoy, O. Sener, A. Alatan, and K. Ugur, “Interactive 2D-3D Image Conversion for Mobile Devices,” in IEEE International Conference on Image Processing, Sep. 2012, pp. 2729–2732.

[23] “Information Technology–Dynamic Adaptive Streaming over HTTP (DASH),” ISO/IEC 23009-1, Dec. 2014.

[24] S. Alcock and R. Nelson, “Application Flow Control in YouTube Video Streams,” in ACM SIGCOMM Computer Communication Review, Apr. 2011.

[25] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, “Evaluating content management techniques for web proxy caches,” ACM SIGMETRICS Performance Evaluation Review, vol. 27, no. 4, pp. 3–11, Mar. 2000.

doi:10.6342/NTU201601753

[26] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through coded distributed caching," in Communications (ICC), 2012 IEEE International Conference on, June 2012, pp. 2467–2472.

[27] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," IEEE/ACM Transactions on Networking, vol. 17, no. 5, pp. 1357–1370, Oct 2009.

doi:10.6342/NTU201601753