Stochastic Dynamic Programming in DASH

Koffka Khan
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: koffka.khan@gmail.com
Wayne Goodridge
Department of Computing and Information Technology The University of the West Indies, Trinidad and Tobago, W.I
Email: wayne.goodridge@sta.uwi.edu.com

ABSTRACT

Stochastic Dynamic Programming (SDP) is an important class of optimization which has recently been used in the area of dynamic adaptive streaming over HTTP (DASH). Though DASH is very popular method of video delivery in recent years it is plagued with problems when multiple players share a bottleneck link. Thus, this area has become a very active area of research. Two works which implement SDP in DASH are selected and their performances compared against the Conventional DASH player. It is shown that SDP works well for various network conditions.

Keywords – **stochastic; dynamic; programming; SDP; optimization; DASH; video; bottleneck; network.**

Date of Submission: Aug 21, 2019 Date of Acceptance: Dec 23, 2019

I. INTRODUCTION

Many DASH approaches require choosing the best option among a range of competing video segments [29], [27], [21], [18]. These so-called optimization problems can be solved using mathematical methods such as linear programming to evaluate maximum advantages or minimum expenses given some goals and under certain limitations for equilibrium-assumed deterministic systems. These goals are formalized in a utility function that prioritizes certain desired results by assessing the benefits of any system decisions. Stochastic dynamic programming (SDP) models [42], [20], [45] show the trade-off between acquiring present utility and changing future events to obtain future usefulness. Such video streaming issues abound because segment selection choices often have significant consequences for influencing future user quality of experience (user-QoE) [48], [24].

Stochastic dynamic programming is an optimization technique suitable for multiple video streaming systems involving nonlinear and random processes. While in optimization processes such as classical linear or nonlinear programming, the time dimension is often ignored, SDP determines ideal state-dependent choices that differ over time. Finally, SDP is recognized as one of the finest instruments for making recurrent choices when dealing with streaming system-inherent uncertainty. The SDP principle is based on the partitioning of a complicated problem into easier sub-problems in several steps, which once solved are combined to provide a general solution. SDP was first created and used in the fields of applied mathematics, economics and engineering. In recent years it has gained much attention in video streaming.

SDP's goal is to find a solution to an optimization problem based on the 'principle of optimality' [5], [37], [39], [23] which states that 'an optimal policy has the property that regardless of the initial state and decision, the remaining decisions must constitute an optimal policy resulting from the first decision'. The principle of optimality enables us to consider a static problem for the present era, assuming that all future choices are made in the best possible way. The impact of the present action therefore adds to both present utility and future utility by affecting the future state of the system. SDP discovers a strategy in this manner that balances present rewards/benefits with future opportunities/possibilities. The method used to solve a Markov decision problem [16] is stochastic dynamic programming. There are six stages of stochastic dynamic programming. The first step describes the problem's optimization goal. An objective must be specific to the problem, acceptable to the decision makers involved, achievable, defined over a period of time also called time horizon, and measurable with a function related to the state and actions of the system. This function, called utility, provides the reward of any action applied to a particular state. Depending on the sort of environment and streaming problem, several goals can be defined, but an optimization goal must be described as maximizing or minimizing a function over a horizon of time [19]. The horizon of time may be defined as finite [34] or infinite [13].

The second step is to identify the set of states at each time step that reflects the feasible system arrangement. Further, let X_t be the system's state variable at the moment t. In the third stage, the decision variable must be defined. This is the system dynamic element that can be controlled to fulfill the goal. The fourth stage is to construct a transition model that describes the dynamics of the system and its behavior with regard to the impact of a decision on the state variables. This transformation model follows a Markov process in which the future X_{t+1} state depends on the present X_t state and the action taken A_t, but not on the system's previous state and action pairs. In the fifth step, the utility function Ut must also be defined at the time t which is called the immediate reward. This function referred to as U_t (X_t, A_t), which pertains to the Markov chain formulation [12], represents the desirability of acting in a given system state and is defined in terms of the state variable X_t (step 2) and A_t (step 3) decision. Depending on the objectives formalized in step 1, the utility values can accumulate over either a finite or an infinite time horizon. The final stage is to determine the optimum solution to the optimization problem. The optimal solution, also referred to as the optimal strategy or policy, maximizes our opportunity to achieve our goal over time. An optimal solution maps each state to the optimal action for that state.

A discussion on solving SPDs is given in Section II. Then the problem in dynamic adaptive streaming over HyperText Transfer Protocol (HTTP) (DASH) based systems with multiple competing players is given in Section III. DASH approaches using SDP is explored in section IV. Experimental setup is given in section V. Results are given in section VI and finally, the conclusion is given in Section VII.

II. Stochastic Dynamic Programming (SDP)

How to select the most suitable algorithms relies primarily on the goal of optimization (step one)? In time-reversed fashion, reverse iteration is the run over a finite horizon. It leads to an optimal time- and state-specific solution. Value iteration and policy iteration are used to resolve problems with the infinite time horizon. Both methods provide optimal action expressed as a feature that is independent of time.

According to the principle of optimality [4], reasoning back in time is an effective way to find an optimal decision. More specifically, it involves assuming that the last decision taken at horizon time T is optimal and choosing what to do in each remaining step of the time. T is the time it takes to get the optimal solution. Let V(X) be the value function [30] of states that quantify the reward or the penalty after each state transition following the choice made. Let Pi* be a vector mapping the best horizon time decision for each state. Pi* is the collection of choices (A) related to the state set highest value function [V(X)]. Let β be the variable of discount, representing the value of the reward earned in the following period relative to the reward received in the present era. It may also represent a level of confidence in the dynamic model's predictions. Overall, predictions for the near future are more certain than those made for the remote future. The problem with the finite horizon can be expressed formally as:

$$V_t(X_t) = \max_{\{A_t\}_{\tau=t}^{T}} \sum_{\tau=t}^{T} \beta^{\tau-t} U(X_\tau, A_\tau) + \beta^{T+1} R(X_{T+1})$$

The term includes two parts, the sum of the discounted utility values from time t to horizon T and the discounted terminal reward ($R(X_T+1)$), which is a function of the state the system is left in, X_T+1 after the last decision has been made.

The starting point in the backward iteration algorithm is to realize that a recursive relationship exists that identifies a value function for step t, denoted $V_t(X_t)$, for each state, since step Vt+1(Xt+1) has already been solved.

$$V_t(X_t) = \max_{A_t}[U_t(X_t, A_t) + \beta V_{t+1}(X_{t+1})]$$

As proposed by the principle of optimality, in terms of the present choice alone, the Bellman equation reflects the problem of optimization. The first part of this equation consists of the immediate reward represented by the utility function, while the second part consists of the value function $V_{t+1}(X_{t+1})$ for the next period. Setting $V_{T+1}(X_{T+1}) = R(X_{T+1})$ initializes the procedure. Then calculate the prior value $V_T(X_T)$, then $V_{T-1}(X_{T-1})$, etc… The optimal action, which is the action associated with each initial state X_0, is obtained by repeated retro-recursions from horizon [3] time T to present time 0 and the maximum initial value argument $V_0(X_0)$. In addition to choosing the horizon T and the system's terminal value, $R(X_{T+1})$, an important issue is the choice of a discount factor β between 0 and 1. In terms of a discount rate r, discounting is often indicated with the discount factor provided by $\beta = 1/(1+r)$. Conservative measures are more likely to use a β of 1, which means that there is discount on the value of future system states. Future utility adds as much to the general goal as present utility in such circumstances. Although future utility discounting does not comply with the principle of sustainability, most streaming algorithms use a discount factor < 1. One reason is that many individuals place greater emphasis on current than future rewards, particularly if future rewards are risky.

With infinite horizon problems [13], both the value function and the optimal strategy are time independent. It is possible to write the problem to be solved as:

$$V(X_t) = \max_{A(X)} \sum_{\tau=0}^{\infty} \beta^{\tau-t} U(X_\tau, A(X_\tau))$$

Starting with an arbitrary value function and iterating over an infinite horizon model with policy or value iteration leads the appropriate action to converge to a time-independent function also called a stationary strategy with the optimal solution only depending on the state of the system and not on time. The first algorithm used to solve an MDP over an infinite horizon, called value iteration, follows the same procedure as earlier outlined except that it iteratively applies the Bellman equation until a convergence criterion is met. A typical convergence criterion is:

$$\| V(X_{t+1}) - V(X_t) \| \leq \frac{\varepsilon(1-\beta)}{2\beta}$$

where the norm $V(X_{t+1})$-$V(X_t)$ is the maximum absolute value of the difference between the two consecutive choice values for all possible states. Usually the value of E is selected to be small, so that when the condition in the previous equation is satisfied, the value function is within E of its optimal value.

Another algorithm called policy iteration [14] includes alternating between discovering the best policy (or strategy) given the present value function guess and determining the value function connected with the present policy. One benefit of the algorithm for policy iteration is that it usually runs quicker than the value iteration algorithm [36]. Policy iteration can be broken down in two steps.

A value function is calculated from a guessed policy in the first stage (evaluation) [36]. Let A_t be any policy describing the actions taken for any state X_t, so that X_{t+1} is a function of state and action variables that can be written as $X_{t+1} = g(X_t, A_t)$. The value function associated with this policy can be determined by solving a linear equation system, one for each state variable value.

$$V_t(X_t) = U_t(X_t, A_t) + \beta V_{t+1}(g(X_t, A_t))$$

In the second stage (improvement), we discover the policy A' that satisfies each state value.

$$\max_{A'} U(X_t, A'_t) + \beta V_{t+1}(g(X_t, A_t))$$

The same procedure is carried out again (back to the first stage) until there is no change in the two policies A and A'. Here we address how to cope with uncertainty in dynamic programming. Because of the action taken and the present state, there are several feasible next states in SDP, and each of them has a certain probability of being achieved. Let P be a transition matrix showing the system's conditional probabilities at X_t at moment t and at (in rows) shift to states X_{t+1} (in columns) given the action. The transition matrix [6] is a stochastic matrix comprising non-negative components with rows summing up to 1. The Bellman equation can be rewritten as the sum of the utility value at the present state (holding in the deterministic variant) and the sum of the anticipated future rewards produced by the transition probabilities and values of all possible future states. For example, the stochastic version of the equation is in the backward iteration procedure:
The distinction is that the transition probability matrix is added. In fact, the Bellman equation's deterministic version can be rewritten as a special SDP case, where P is a 0s matrix with a single 1 in each row. In SDP, P comprises of probabilities of change based on stochastic occurrences linked to population and/or environmental stochasticity or action that may have an uncertain impact. We identify various kinds of uncertainty that can be accounted for.
First, there is the natural uncertainty in the system and its environment that is linked to the natural and intrinsic procedures that occur. It's hard to assess and even harder to decrease, if not impossible. Second partial controllability is the result of the failure to apply the choice being taken correctly. Sometimes, actions are taken in an unclear manner themselves. The third sort of uncertainty deals with that which comes from the partial understanding of state variable's value. To cope with such uncertainty, the partially observed Markov decision

$$V_t(X_t) = \max_{A_t}\left[U_t(X_t, A_t) + \beta \sum_{X_{t+1}} P(X_{t+1} \mid X_t, A_t) V_{t+1}(X_{t+1})\right]$$

process (POMDP) [28], [31], [35], [44] may be used, a procedure that can solve stochastic dynamic problems if we are unable to fully observe the state of the system. Another type of uncertainty is model uncertainty, referring to the absence of certainty regarding the structural framework that shapes the system's behaviour. The last form of uncertainty, called parametric uncertainty, is related to our limited knowledge of system dynamics parameters. One strategy to this problem uses the unknown parameters of conjugating priors [10]. If the type of uncertainty is unknown, reinforcement learning is an alternative optimization strategy to reverse iteration, strategy or value iteration. This technique makes sequential decisions when the transition probabilities or rewards are unknown and cannot be estimated by simulation. The Q-learning algorithm [47] is used in which the optimal value V_0* and the associated action are estimated through a learning method of observed transitions and values captured with approximation function.

III. DASH and Multiple Competing Players Problem

The concept of adaptive video streaming is based on the idea to adapt the bandwidth required by the video stream to the throughput available on the network path from the stream source to the client [32]. These algorithms can live at the server [26], at an intermediate network device [25] or at the client [27], [21]. With client-side protocols it is the player that decides what bitrate to request for any fragment, improving server-side scalability [1]. A benefit of this protocol is that the player can control its playback buffer size by dynamically adjusting the rate at which new fragments are requested. The adaptation is performed by varying the quality of the streamed video. Multiple video segments constitute a video stream lasting from as little as 2 seconds to as much as having a 10 second chunk delivery rate. Segments are encoded and stored on the server in numerous quality versions, termed representations. Each version has a unique resolution, bitrate and/or quality. A client downloads segments using HTTP GET statements [7]. However, with adaptive streaming a client might request subsequent segments at different quality levels to manage varying network conditions, based on an estimation bandwidth. To do this it uses a manifest file that contains information about the video segments. Protocols and standards such as MPEG Dynamic Adaptive Streaming over HTTP (DASH), Apple HTTP Live Streaming (HLS), Microsoft Smooth Streaming (MSS) or Adobe HTTP Dynamic Streaming (HDS) typically use a media playlist that contains a list of uniform resource identifiers (URIs) that are addresses to media segments [38]. The process of determining the ideal representation for each segment to enhance the user’s experience is pivotal to adaptive streaming. The controller algorithm estimates the network bandwidth and chooses the next bitrate level corresponding to the available network bandwidth. Variations in the available bandwidth will result in jerky playback and disruption of the video playback if the throughput falls below the bit rate requirement of the video. This is the major challenge in adaptive video streaming. Selecting appropriate bitrate levels helps to maximize the user experience. Generally, higher bitrates and resolutions will give better user experience. For example, if a client approximates that there is 9.5Mb/s available in the network, it might request the server to stream video compressed to the highest video rate available, 9.5Mb/s, or the next rate below, 9.3Mb/s. If the client picks a video rate that is too high, the viewer will experience annoying re-buffering events; if they pick a streaming rate that is too low, the viewer will experience poor video quality.

Adaptive streaming uses the HTTP/TCP protocol stack to transmit video Web traffic. Thus, the development of this wave of HTTP-based streaming applications is not referred to as adaptive streaming over HTTP. The use of HTTP/TCP protocols for video streaming is because of the advantages that HTTP/TCP offers. It allows standard web servers and caches to be used increasing its' cost effectiveness. Another advantage is that all firewalls are configured to support HTTP connections [46]. In addition, is allows better scaling as HTTP is stateless and the streaming session is managed by the client, thus reducing the load on the server. However, HTTP/TCP use reveals further challenges as adaptation is on top of TCPs congestion control algorithm, which forms nested control loops. As the throughput of the TCP connection depends on both the link capacity and the amount of congestion, the throughput can vary significantly over time [2].

In the presence of competing HTTP-based adaptive streaming (HAS) clients the TCP throughput does not always faithfully represent the fair-share bandwidth [29]. Three performance issues that can take place when two or more adaptive streaming players share a network bottleneck and compete for available bandwidth are instability, unfairness and utilization [27]. It is shown that in the case of two competing video flows Adaptive video streaming players provide a received video rate that oscillates around the fair share, but with an increased number of video level switches [18]. Depending on the temporal overlap of the ON-OFF [38] periods among competing players, they may not estimate their fair share correctly [21]. In the case where both players overestimate their fair share, they may request a video representation with a higher bitrate than the fair share, which causes network congestion. Consequently, the players measure that their TCP throughput is lower than their previous fair share estimate, and so switch down to a lower video bitrate level. This creates a repeating oscillatory scenario, so inducing instability. A scenario can also occur where some players are requesting chunks with lower bitrates than the other players. This can occur as some players observe a throughput lower than the fair share, while others observe a throughput that is more than the fair share. This means that some players overestimate its fair share. When some players overestimate their fair share, it can be that the system of players converge to a stable equilibrium, but unfair. This occurs as the players with the larger fair share estimates request higher bitrate video levels. Even in the case where two players estimate their fair share correctly, bandwidth underutilization can still be prevalent. This occurs as both players request the same lower video bitrate level, which causes underutilization, even though stability and fairness still exist. In reality, several other factors can play an important role in the appearance and extent of instability, unfairness and underutilization, such as the exact player adaptation algorithm, TCP dynamics, bandwidth fluctuations, and the variability of the video encoding rate [21]. We group these problems into three categories: The first relates to the stability of the players in terms of requested bitrates and video quality. The second is the unfairness among competing players. The third is the potential bandwidth underutilization when multiple adaptive players compete. They are described as follows:

Instability: The instability for player i at time t is given in Equation 7, where $w(d) = k - d$ is a weight function that puts more weight on more recent samples. k is selected as 20 seconds [18].

$$Instability = \frac{\sum_{d=0}^{k-1} \left| r_{i,t-d} - r_{i,t-d-1} \right| * w(d)}{\sum_{d=0}^{k-1} r_{i,t-d} * w(d)}$$

Unfairness: Let $JainFair_t$ be the Jain fairness index (cf. Equation 10) calculated on the average received rates , r_i, (cf. Equation 9) at time t over all players [17]. The unfairness at time t is defined as $\sqrt{1 - JainFair_t}$. A lower value implies a fairer allocation.

$$r_i = \frac{downloaded\ bytes}{time\ interval}$$

$$JFI = \frac{(\sum_{i=1}^{n} r_i)^2}{n \sum_{i=1}^{n} r_i^2}$$

The utilization metric is defined as the aggregate throughput during an experiment divided by the available bandwidth in that experiment (cf. Equation 6, where tp_i is the throuput at time i and bw is the experimental available bandwidth) [29].

$$Utilization = \frac{\sum_{i=0}^{n-1} tp_i}{bw}.$$

IV. SDP DASH-Based Approaches

A. *Method A [11]*

Dynamic Adaptive Streaming over HTTP (DASH) is a recent MPEG standard for IP video delivery whose aim is the convergence of existing adaptive-streaming proprietary solutions. However, it does not impose any adaptation logic for selecting the quality of the media segments requested by the client, which is crucial to cope effectively with bandwidth fluctuations, notably in wireless channels. Authors therefore propose a solution to this control problem through Stochastic Dynamic Programming (SDP). This approach requires a probabilistic characterization of the system, as well as the definition of a cost function that the control strategy aims to minimize. This cost function is designed taking into account factors that may influence the quality perceived by the users. Unlike previous works, which compute control policies online by learning from experience, our algorithm solves the control problem offline, leading promptly to better results. In addition, we compared our algorithm to others during a streaming simulation and we analyzed the objective results by means of a Quality of Experience (QoE) oriented metric. Moreover, we conducted subjective tests to complete the evaluation of the performance of our algorithm. The results show that our proposal outperforms the other approaches in terms of both the QoE-oriented metric and the subjective evaluation.

B. Method B [9]

HTTP Streaming has been a new trend for video delivering via IP. Currently, most of the adaptation algorithms developed for HTTP Streaming are qualitative, which means the performance metrics could only be shown after the streaming session. In this paper, we embark on this problem by discretizing the whole system. We then formulate an infinite horizon problem (IHP) and solve it by Stochastic Dynamic Programming (SDP). To deal with the time-varying characteristics of video bitrate, we estimate the instant bitrate from the previous bitrate. We also develop mathematical models that predict the average performance of the adaptation policy, which helps adjust the settings before a streaming session. In the evaluation, we compare our method with a previous work which applies bitrate estimation and another method which only applies SDP.

V. EXPERIMENTAL SETUP

A virtual network is setup on the same host machine creating a custom emulation framework. Our setup consists of client players, video servers, and a bottleneck link. The server resides on a Windows 10 machine. All experiments are performed on a Windows 10 client with an Intel(R) Core(TM)i7-5500U CPU 2.40GHz processor, 16.00 GB physical memory, and an Intel(R) HD Graphics processor. It serves video data to the client(s) who are on a Ubuntu operating system hosted on VMware. The virtual machine is allocated 12GB of physical memory.

TAPAS [8] is installed on Ubuntu 15.04 Linux. The TAPAS Adaptive Video Controller client makes different video segment bitrate level requests to the Apache server. TAPAS allow multiple instances of the player to be created enabling multi-client scenarios. This work involves the interaction between adaptive streaming algorithm at the controller and TAPAS player (cf. Figure 6). All traffic between clients and servers go through the bottleneck, which uses VMware settings which allow bandwidth limits to be set during the experiment. TAPAS support both the HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) format. Algorithms that uses Method A and Method B was tested and shown to work on both MPEG-DASH [40], and Apple HTTP Live Streaming (HLS) [33]. This makes it useful for video on demand (VOD) [15] and live streaming [41], for example, real-time video chats. However, the MPEG-DASH standard is used for testing in this research paper, because it makes the experiments more comparable to the ones in the research literature, for example, [29], [18].

The ten-minute-long MPEG-DASH video sequence "Elephant's Dream"[1] is encoded at twenty different bitrates, between 46 Kbps to 4200Kbps and five different resolutions, between 320x240 to 1920x1080, is used to run the experiments (cf. Table II). The video is encoded at 24 frames per second (fps) using the AVC1 codec [43]. Fragment duration of 2s is used and is recorded in the mpd playlist accordingly. All the DASH files (.m4s fragments and MPD playlists) are placed on the Apache server. We implemented three client-side algorithms in the TAPAS controller. The conventional approach is present by default and is used as a baseline in which to compare against other algorithms. TAPAS is lightweight in built, thus allowing the same receiving host to run a large number of separate video player instances at the same time at different command line interfaces. Thus, it allows the multi-client scenarios which are essential to the work in this paper.

VI. RESULTS

The first experiment illustrates five players competing at a 20Mbps bottleneck link. Table 1 gives the results. Method A outperforms Method B and the Conventional.

	Method A	Method B	Conventional
Utilization	0.92	0.86	0.68
Unfairness	0.020	0.059	0.124
Instability	0.151	0.210	0.311

The second experiment illustrates five players competing at a 20Mbps bottleneck link and stopping or pausing during the experiment. Table 2 gives the results. Method A outperforms Method B and the Conventional.

	Method A	Method B	Conventional
Utilization	0.93	0.84	0.71
Unfairness	0.015	0.067	0.136
Instability	0.146	0.243	0.356

The third experiment illustrates five players competing at a 100Mbps bottleneck link with increasing number of players up to 20. Table 3 gives the results. Method A outperforms Method B and the Conventional.

	Method A	Method B	Conventional
Utilization	0.85	0.77	0.59
Unfairness	0.037	0.068	0.173
Instability	0.175	0.258	0.384

The fourth and final experiment illustrates five players competing at a 20Mbps bottleneck link in bandwidth varying conditions. Table 4 gives the results. Method A outperforms Method B and the Conventional.

	Method A	Method B	Conventional
Utilization	0.83	0.73	0.55
Unfairness	0.045	0.073	0.398
Instability	0.194	0.280	0.401

VII. CONCLUSION

Stochastic Dynamic Programming (SDP) is an important class of optimization which has recently been used in the area of dynamic adaptive streaming over HTTP (DASH).

Though DASH is very popular method of video delivery in recent years it is plagued with problems when multiple players share a bottleneck link. Thus, this area has become a very active area of research. Two works which implement SDP in DASH are selected and their performances compared against the Conventional DASH player. It is shown that SDP works well for various network conditions. However, one method outperforms the others in the experiments conducted. Future work could involve the use of multipath routing protocols [22] between the home gateway router and the server for improving current performance.

REFERENCES

[1] Akhshabi, Saamer, Sethumadhavan Narayanaswamy, Ali C. Begen, and Constantine Dovrolis. "An experimental evaluation of rate-adaptive video players over HTTP." Signal Processing: Image Communication 27, no. 4 (2012): 271-287.

[2] Allman, Mark, Vern Paxson, and Wright Stevens. "TCP congestion control." (1999).

[3] Ang, James S., and Shaojun Zhang. "Choosing benchmarks and test statistics for long horizon event study." In EFMA 2002 London Meetings. 2002.

[4] Bellman, Richard. "A Markovian decision process." Journal of mathematics and mechanics (1957): 679-684.

[5] Carbone, Enrica, and John D. Hey. "A Test of the Principle of Optimality." Theory and Decision 50, no. 3 (2001): 263-281.

[6] Craig, Bruce A., and Peter P. Sendi. "Estimation of the transition matrix of a discrete-time Markov chain." Health economics 11, no. 1 (2002): 33-42.

[7] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "Elastic: a client-side controller for dynamic adaptive streaming over http (dash)." In 2013 20th International Packet Video Workshop, pp. 1-8. IEEE, 2013.

[8] De Cicco, Luca, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "TAPAS: a Tool for rApid Prototyping of Adaptive Streaming algorithms." In Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming, pp. 1-6. ACM, 2014.

[9] Duong, Anh H., Thoa Nguyen, Thang Vu, Tung T. Do, Nam Pham Ngoc, and Truong Cong Thang. "SDP-based adaptation for quality control in adaptive streaming." In 2015 International Conference on Communications, Management and Telecommunications (ComManTel), pp. 194-199. IEEE, 2015.

[10] Fink, Daniel. "A compendium of conjugate priors." See http://www. people. cornell. edu/pages/df36/CONJINTRnew% 20TEX. pdf 46 (1997).

[11] García, S., Cabrera, J. and García, N., 2014. Quality-control algorithm for adaptive streaming services over wireless channels. IEEE Journal of Selected Topics in Signal Processing, 9(1), pp.50-59.

[12] Gibson, Gavin J., and Eric Renshaw. "Estimating parameters in stochastic compartmental models using Markov chain methods." Mathematical Medicine and Biology: A Journal of the IMA 15, no. 1 (1998): 19-40.

[13] Halkin, Hubert. "Necessary conditions for optimal control problems with infinite horizons." Econometrica: Journal of the Econometric Society (1974): 267-272.

[14] Howard, Ronald A. "Dynamic programming and markov processes." (1960).

[15] Huang, Cheng, Jin Li, and Keith W. Ross. "Can internet video-on-demand be profitable?." In ACM SIGCOMM Computer Communication Review, vol. 37, no. 4, pp. 133-144. ACM, 2007.

[16] Jaakkola, Tommi, Satinder P. Singh, and Michael I. Jordan. "Reinforcement learning algorithm for partially observable Markov decision problems." In Advances in neural information processing systems, pp. 345-352. 1995.

[17] Jain, Raj, Arjan Durresi, and Gojko Babic. "Throughput fairness index: An explanation." In ATM Forum contribution, vol. 99, no. 45. 1999.

[18] Jiang, Junchen, Vyas Sekar, and Hui Zhang. "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." IEEE/ACM Transactions on Networking (ToN) 22, no. 1 (2014): 326-340.

[19] Keeney, Ralph L., and Howard Raiffa. Decisions with multiple objectives: preferences and value trade-offs. Cambridge university press, 1993.

[20] Kelman, Jerson, Jery R. Stedinger, Lisa A. Cooper, Eric Hsu, and Sun-Quan Yuan. "Sampling stochastic dynamic programming applied to reservoir operation." Water Resources Research 26, no. 3 (1990): 447-454.

[21] Khan, Koffka, and Wayne Goodridge. "B-DASH: broadcast-based dynamic adaptive streaming over HTTP." International Journal of Autonomous and Adaptive Communications Systems 12, no. 1 (2019): 50-74.

[22] Khan, Koffka, and Wayne Goodridge. "Energy aware Ad-Hoc on demand multipath distance vector routing." International Journal of Intelligent Systems and Applications 7, no. 7 (2015): 50-56.

[23] Khan, Koffka, and Wayne Goodridge. "Future DASH Applications: a Survey." International Journal of Advanced Networking and Applications 10, no. 2 (2018): 3758-3764.

[24] Khan, Koffka, and Wayne Goodridge. "QoE in DASH." International Journal of Advanced Networking and Applications 9, no. 4 (2018): 3515-3522.

[25] Khan, Koffka, and Wayne Goodridge. "SAND and Cloud-based Strategies for Adaptive Video Streaming." International Journal of Advanced Networking and Applications 9, no. 3 (2017): 3400-3410.

[26] Khan, Koffka, and Wayne Goodridge. "Server-based and network-assisted solutions for adaptive video streaming." International Journal of Advanced Networking and Applications 9, no. 3 (2017): 3432-3442.

[27] Khan, Koffka, and Wayne Goodridge. "S-MDP: Streaming with Markov Decision Processes." IEEE Transactions on Multimedia (2019).
[28] Krishnamurthy, Vikram, and Dejan V. Djonin. "Structured threshold policies for dynamic sensor scheduling—A partially observed Markov decision process approach." IEEE Transactions on Signal Processing 55, no. 10 (2007): 4938-4957.
[29] Li, Zhi, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. "Probe and adapt: Rate adaptation for HTTP video streaming at scale." IEEE Journal on Selected Areas in Communications 32, no. 4 (2014): 719-733.
[30] Littman, Michael L. "Value-function reinforcement learning in Markov games." Cognitive Systems Research 2, no. 1 (2001): 55-66.
[31] Lovejoy, William S. "A survey of algorithmic methods for partially observed Markov decision processes." Annals of Operations Research 28, no. 1 (1991): 47-65.
[32] Miller, Konstantin, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. "Adaptation algorithm for adaptive streaming over HTTP." In 2012 19th international packet video workshop (PV), pp. 173-178. IEEE, 2012.
[33] Müller, Christopher, Stefan Lederer, and Christian Timmerer. "An evaluation of dynamic adaptive streaming over HTTP in vehicular environments." In Proceedings of the 4th Workshop on Mobile Video, pp. 37-42. ACM, 2012.
[34] Mundhenk, Martin, Judy Goldsmith, Christopher Lusena, and Eric Allender. "Complexity of finite-horizon Markov decision process problems." Journal of the ACM (JACM) 47, no. 4 (2000): 681-720.
[35] Nguyen, Dong, and Thinh Nguyen. "Network coding-based wireless media transmission using POMDP." In 2009 17th International Packet Video Workshop, pp. 1-9. IEEE, 2009.
[36] Pashenkova, Elena, Irina Rish, and Rina Dechter. "Value iteration and policy iteration algorithms for Markov decision problem." In AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning. 1996.
[37] Porteus, Evan. "An informal look at the principle of optimality." Management Science 21, no. 11 (1975): 1346-1348.
[38] Seufert, Michael, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. "A survey on quality of experience of HTTP adaptive streaming." IEEE Communications Surveys & Tutorials 17, no. 1 (2014): 469-492.
[39] Sniedovich, M. "A new look at Bellman's principle of optimality." Journal of Optimization Theory and Applications 49, no. 1 (1986): 161-176.
[40] Sodagar, Iraj. "The mpeg-dash standard for multimedia streaming over the internet." IEEE multimedia 18, no. 4 (2011): 62-67.
[41] Sripanidkulchai, Kunwadee, Bruce Maggs, and Hui Zhang. "An analysis of live streaming workloads on the internet." In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pp. 41-54. ACM, 2004.
[42] Stedinger, Jery R., Bola F. Sule, and Daniel P. Loucks. "Stochastic dynamic programming models for reservoir operation optimization." Water resources research 20, no. 11 (1984): 1499-1505.
[43] Stockhammer, Thomas. "Dynamic adaptive streaming over HTTP--: standards and design principles." In Proceedings of the second annual ACM conference on Multimedia systems, pp. 133-144. ACM, 2011.
[44] Supancic III, James, and Deva Ramanan. "Tracking as online decision-making: Learning a policy from streaming videos with reinforcement learning." In Proceedings of the IEEE International Conference on Computer Vision, pp. 322-331. 2017.
[45] Tejada-Guibert, J. Alberto, Sharon A. Johnson, and Jery R. Stedinger. "The value of hydrologic information in stochastic dynamic programming models of a multireservoir system." Water resources research 31, no. 10 (1995): 2571-2579.
[46] Wang, Jia. "A survey of web caching schemes for the internet." ACM SIGCOMM Computer Communication Review 29, no. 5 (1999): 36-46.
[47] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8, no. 3-4 (1992): 279-292.
[48] Yoshimura, Yasuhiko, and Masao Masugi. "A QoS monitoring method for video streaming service based on presentation-timeline detection at user clients." In APCC/MDMC'04. The 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceeding, vol. 2, pp. 681-685. IEEE, 2004.

AUTHOR DETAILS:

Koffka Khan received the M.Sc., and M.Phil. degrees from the University of the West Indies. He is currently a PhD student and has up-to-date, published numerous papers in journals & proceedings of international repute. His research areas are computational intelligence, routing protocols, wireless communications, information security and adaptive streaming controllers.

Wayne Goodridge is a Lecturer in the Department of Computing and Information Technology, The University of the West Indies, St. Augustine. He did is PhD at Dalhousie University and his research interest includes computer communications and security.