Report

Yifu TIAN 121090517

Algorithm Analysis

Abstract

In Assignment 2, I implemented **BOLA-BASIC** Algorithm after reading the article, “BOLA: Near-Optimal Bitrate Adaptation for Online Videos”. This article formulates bitrate adaptation as a utility maximization problem that incorporates both key components of QoE(Quality of Experience): the average bitrate of the video experienced by the user and the duration of the rebuffer events. An increase in the average bitrate will increase utility, whereas rebuffering will decrease it.

Using Lyapunov optimization, it derives an online bitrate adaptation algorithm called **BOLA (Buffer Occupancy based Lyapunov Algorithm)** that achieves utility which is within an additive factor of the maximum possible utility. This algorithm is the first to provide a theoretical guarantee on the achieved utility.

Analysis

BOLA operates on the principle of utility-based decision-making, where each available bitrate is assigned a utility value based on the logarithm of the bitrate, normalized by the size of the video chunk. The algorithm then adjusts the bitrate by maximizing this utility while considering the current buffer occupancy and a Lyapunov optimization function that ensures system stability and prevents excessive rebuffering.

The pseudocode is as below:

for n in $[1, N]$ **do**
  $t \leftarrow \min$[playtime from begin, playtime to end]
  $t' \leftarrow \max[t/2, 3p]$
  $Q^{\mathrm{D}}_{\max} \leftarrow \min[Q_{\max}, t'/p]$
  $V^{\mathrm{D}} \leftarrow (Q^{\mathrm{D}}_{\max} - 1)/(v_1 + \gamma p)$
  $m^*[n] \leftarrow \arg\max_m (V^{\mathrm{D}} v_m + V^{\mathrm{D}} \gamma p - Q)/S_m$
  if $m^*[n] < m^*[n-1]$ **then**
    $r \leftarrow$ bandwidth measured when downloading chunk $(n-1)$
    $m' \leftarrow \min m$ such that $S_m/p \leq \max[r, S_M/p]$
    if $m' \leq m^*[n]$ **then**
      $m' \leftarrow m^*[n]$
    else if $m' > m^*[n-1]$ **then**
      $m' \leftarrow m^*[n-1]$
    else if some utility sacrificed for fewer oscillations **then**
      pause until $(V^{\mathrm{D}} v_{m'} + V^{\mathrm{D}} \gamma p - Q)/S_{m'} \geq (V^{\mathrm{D}} v_{m'-1} + V^{\mathrm{D}} \gamma p - Q)/S_{m'-1}$ ▷ BOLA-O
    else
      $m' \leftarrow m' - 1$ ▷ BOLA-U
    end if
    $m^*[n] \leftarrow m'$
  end if
  pause for $\max[p \cdot (Q - Q^{\mathrm{D}}_{\max} + 1), 0]$
  download chunk n at bitrate index $m^*[n]$, possibly abandoning
end for

Fig. 4. The BOLA Algorithm.

1. The algorithm iterates over each chunk n from 1 to N, where N is the total number of chunks.

2. t is set to the minimum of the playback time from the beginning and the end, ensuring that only the current buffer is considered; t' is the half of t or thrice the parameter p, whichever is larger, which seems to be a threshold or a decision parameter.

3. Q^D_max is the dynamic Q value, computed as the minimum of Q_max and t'/p. This represents a dynamic buffer size; V^D is a value derived from Q^D_max, V, and p, influencing the utility for each bitrate.

4. m*[n] finds the bitrate that maximizes the utility function, which is a log function of the bitrate adjusted by V^D, p, and the buffer size.

5. If the selected bitrate for the current chunk is lower than the previous chunk's bitrate, the algorithm checks if it should adjust the bitrate based on the bandwidth measured when downloading the previous chunk.

6. m' finds a bitrate that fits the current estimated bandwidth.

7. The algorithm then either keeps the selected bitrate, adjusts it based on utility comparison, or decides to pause if the utility sacrificed for fewer oscillations is too great, as per BOLA-O and BOLA-U thresholds.

8. Finally, the algorithm may pause for a certain time based on buffer levels before downloading the next chunk.

Implementation

Firstly we need to implement calculating the utility of each available bitrate. In this article, the utility function is as below:

$$v_m = \ln(S_m/S_M)$$

Then implement the objective formula we want to maximize is as below:

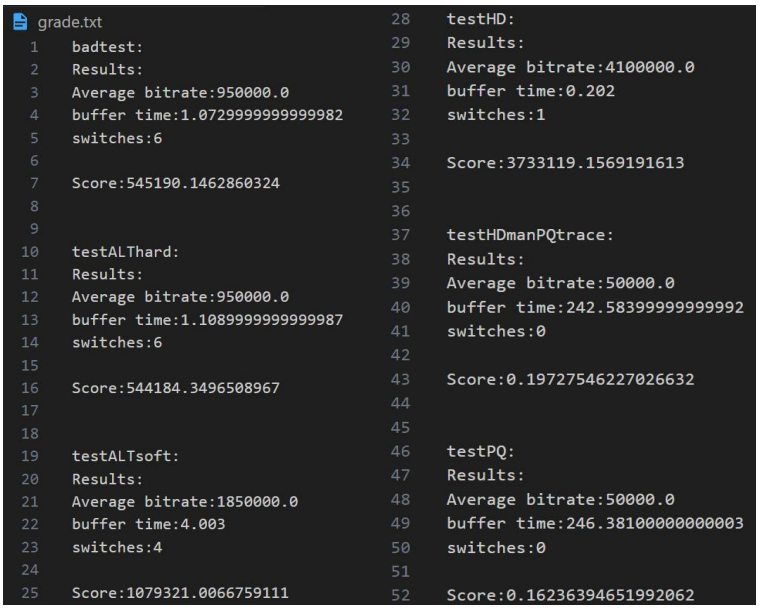
$$(Vv_m + V\gamma p - Q)/S_m \text{ for } 1 \leq m \leq M.$$

Part of my program implementation is as below:

```python
def BOLA(bitrates_array, buffer_size):
    '''
    I only implemented BOLA-BASIC here

    V: A parameter that affects the trade-off between high bitrate and rebuffering risk
    p: A parameter related to buffer content
    gamma: A parameter that translates the utility function to avoid rebuffering
    CHUNK_TIME: The duration of each video chunk in seconds.
    S_m: A dictionary that maps each bitrate to its corresponding chunk size
    Q: A parameter that indicates the number of buffered chunks.(This spends me a lot of time to figure out)

    list_of_utility: A list to store the result of each goal_formula
    '''
    V = 0.93
    p = 5
    gamma = 1
    CHUNK_TIME = 2
    S_m = {br: size for br, size in bitrates_array}
    list_of_utility = []

    for br, sz in bitrates_array:

        # Calculate utility for each bitrate, we will use this in goal_formula
        utility = np.log(sz / S_m[br])

        # Q indicated the number of buffered chunks
        Q = buffer_size["time"] / CHUNK_TIME

        # goal_formula is the formula we want to maximize
        goal_formula = (V * utility + V * p * gamma - Q) / S_m[br]

        list_of_utility.append(goal_formula)

    # Select the index of the bitrate with the maxium utility
    optimal_bitrate_index = np.argmax(list_of_utility)
    optimal_bitrate = bitrates_array[optimal_bitrate_index][0]
```

Evaluation

The grade result of my program is as below:



grade.txt

```
badtest:
Results:
Average bitrate:950000.0
buffer time:1.0729999999999982
switches:6

Score:545190.1462860324


testALThard:
Results:
Average bitrate:950000.0
buffer time:1.1089999999999987
switches:6

Score:544184.3496508967


testALTsoft:
Results:
Average bitrate:1850000.0
buffer time:4.003
switches:4

Score:1079321.0066759111


testHD:
Results:
Average bitrate:4100000.0
buffer time:0.202
switches:1

Score:3733119.1569191613


testHDmanPQtrace:
Results:
Average bitrate:50000.0
buffer time:242.58399999999992
switches:0

Score:0.19727546227026632


testPQ:
Results:
Average bitrate:50000.0
buffer time:246.38100000000003
switches:0

Score:0.16236394651992062
```

The evaluation of BOLA-BASIC is centered around its performance in achieving a balance between video quality and the possibility of rebuffering events.

1. The algorithm assigns utility values to different bitrates based on the logarithmic function, which inherently values a smooth playback experience instead of aggressive quality maximization. The utility-based approach ensures that the quality of the video adapts dynamically, ensuring that the viewer's experience is optimized within the constraints of the current buffer state.
2. In terms of performance metrics, BOLA-BASIC's effectiveness is demonstrated through comparisons with the offline optimal bound and other adaptive bitrate algorithms. The comparisons highlight BOLA-BASIC's strength in maintaining a high utility value while minimizing rebuffering events, which is a critical aspect of the viewer's quality of experience.
3. The adaptability of BOLA-BASIC is particularly notable. It shows resilience to fluctuating network conditions and reacts solely based on buffer thresholds, which eliminates the need for complex bandwidth prediction mechanisms. This buffer-only

decision-making process simplifies the algorithm's implementation while still delivering a robust streaming experience.

4. BOLA-BASIC also showcases its efficacy through simulations that reveal its competence in handling various network profiles and video lengths. Its utility values remain consistently close to the optimal, signifying that it can effectively maximize the perceived quality of the video content across different scenarios.

5. Furthermore, the BOLA-BASIC algorithm is easy to tune with a single parameter, V, which adjusts the trade-off between bitrate and buffer occupancy. By tuning V, providers can customize the aggressiveness of bitrate selection to cater to specific content or network environments.