ECE4016 Assignment 2 Report

Jiarui Chen 120090361

I. Algorithm Analysis

In this project, referring to paper "BOLA: Near-Optimal Bitrate Adaptation for Online Videos", we write a BOLA algorithm. BOLA (Buffer Occupancy based Lyapunov Algorithm) is a buffer-based ABR (Adaptive Bitrate Streaming) algorithm using Lyapunov optimization. While building system model, we need to define some variables:

1. The video file is segmented into *N* segments, each segment represents *p* seconds in the video.
2. Each segment is available in *M* different bitrates.
3. The size of the segment encoded at bitrate index *m* is S_m and suppose the utility derived by the user from viewing it is given by
   v_m where $m \in \{1, 2, \ldots, M\}$. Let segment bitrates be non-decreasing in index *m* then

$$v_1 \leq v_2 \leq \ldots \leq v_m \Leftrightarrow S_1 \leq S_2 \leq \ldots \leq S_m.$$

4. The available bandwidth (in bits/second) is assumed to vary continuously in time according to a stationary random process $\omega(t)$.
5. The video player have a finite buffer of size Q_{max} segments.
6. The timeline is divided into nonoverlapping consecutive slots of variable length and indexed by $k \in \{1, 2, \ldots\}$. Slot k starts at time t_k and is $T_k = t_{k+1} - t_k$ seconds long.
7. We define

$$a_m(t_k) = \begin{cases} 1, \text{if the player downloads a segment of bitrate index } m \text{ in slot } k \\ 0, \text{otherwise} \end{cases}$$

   and denote the buffer level at the start of slot *k* by $Q(t_k)$. Then

$$Q(t_{k+1}) = max[Q(t_k) - \frac{T_k}{p}, 0] + a_m(t_k)$$

II. Implementation

The algorithm we implemented in this assignment is BOLA-FINITE.

1: **for** n in $[1, N]$ **do**
2: $t \leftarrow \min[\text{playtime from begin}, \text{playtime to end}]$
3: $t' \leftarrow \max[t/2, 3p]$
4: $Q^{\mathrm{D}}_{\max} \leftarrow \min[Q_{\max}, t'/p]$
5: $V^{\mathrm{D}} \leftarrow (Q^{\mathrm{D}}_{\max} - 1)/(\upsilon_M + \gamma p)$
6: $m^*[n] \leftarrow \arg\max_m (V^{\mathrm{D}}\upsilon_m + V^{\mathrm{D}}\gamma p - Q)/S_m$
7: **if** $m^*[n] > m^*[n-1]$ **then**
8: $r \leftarrow$ bandwidth measured when downloading segment $(n-1)$
9: $m' \leftarrow \max m$ such that $S_m/p \le \max[r, S_1/p]$
10: **if** $m' \ge m^*[n]$ **then**
11: $m' \leftarrow m^*[n]$
12: **else if** $m' < m^*[n-1]$ **then**
13: $m' \leftarrow m^*[n-1]$
14: **else if** some utility sacrificed for fewer oscillations **then**
15: pause until $(V^{\mathrm{D}}\upsilon_{m'} + V^{\mathrm{D}}\gamma p - Q)/S_{m'} \ge (V^{\mathrm{D}}\upsilon_{m'+1} + V^{\mathrm{D}}\gamma p - Q)/S_{m'+1}$ ▷ BOLA-O
16: **else**
17: $m' \leftarrow m' + 1$ ▷ BOLA-U
18: **end if**
19: $m^*[n] \leftarrow m'$
20: **end if**
21: pause for $\max[p \cdot (Q - Q^{\mathrm{D}}_{\max} + 1), 0]$
22: download segment n at bitrate index $m^*[n]$, possibly abandoning
23: **end for**

Fig 1. Pseudo code of BOLA-FINITE

In actual implementation, codes in line 14,15 (BOLA-O) and line 21 will not be implemented. We set υ_m to $\ln(S_m/S_1)$. The buffer level Q is calculated by $\lceil t/p \rceil$, where t is the time remain in the buffer. Despite to the arguments we introduce in part I, there is an argument γ, which is an input weight parameter. In actual implementation, we set $\gamma = 5/p$.

III. Evaluation

We run the program by *python grader.py*. We have the score of given example program and the score of BOLA:

```
testHDmanPQtrace:
Results:Average bitrate:500000.0buffer time:91.115switches:0
Score:4669.348620686024


testPQ:
Results:Average bitrate:500000.0buffer time:91.115switches:0
Score:4669.348620686024


testALTsoft:
Results:Average bitrate:4100000.0buffer time:5.948000000000001switches:9
Score:1426836.8172498895


testHD:
Results:Average bitrate:4700000.0buffer time:0.101switches:1
Score:4301656.912826439


badtest:
Your trace file is poorly formed!Results:Average bitrate:2600000.0buffer time:19.751switches:28
Score:91422.1523596769


testALThard:
Results:Average bitrate:2600000.0buffer time:19.751switches:28
Score:91422.1523596769


```

Fig 2. The score of Example Program

```
testHDmanPQtrace:
Results:Average bitrate:983333.3333333334buffer time:245.21300000000002switches:1
Score:3.1190835791810283


testPQ:
Results:Average bitrate:983333.3333333334buffer time:245.21300000000002switches:1
Score:3.1190835791810283


testALTsoft:
Results:Average bitrate:3833333.3333333335buffer time:3.1610000000000014switches:8
Score:1672875.1145839747


testHD:
Results:Average bitrate:4700000.0buffer time:0.101switches:1
Score:4301656.912826439


badtest:
Your trace file is poorly formed!Results:Average bitrate:2350000.0buffer time:9.295000000000002switches:19
Score:299210.9568638165


testALThard:
Results:Average bitrate:2350000.0buffer time:9.295000000000002switches:19
Score:299210.9568638165


```

Fig 3. The score of BOLA Algorithm

We found that BOLA Algorithm does more better than algorithm in example program except PQ test. PQ test is a test case with extremely bad network quality. That means BOLA Algorithm does not perform well when bandwidth value is extremely low.

To improve this situation, we add a code block to check if the buffer only remains 1 segment. If it is, set the next bitrate to the lowest.

```
if buffer_info['time']<=p:
    opt = 0
```

Fig 4. The modification made to improve the algorithm

After this improvement, we have a new score for BOLA, which performs better in PQ test:

```
testHDmanPQtrace:
Results:Average bitrate:500000.0buffer time:91.115switches:0
Score:4669.348620686024


testPQ:
Results:Average bitrate:500000.0buffer time:91.115switches:0
Score:4669.348620686024


testALTsoft:
Results:Average bitrate:3833333.3333333335buffer time:3.1610000000000014switches:8
Score:1672875.1145839747


testHD:
Results:Average bitrate:4700000.0buffer time:0.101switches:1
Score:4301656.912826439


badtest:
Your trace file is poorly formed!Results:Average bitrate:2350000.0buffer time:9.295000000000002switches:19
Score:299210.9568638165


testALThard:
Results:Average bitrate:2350000.0buffer time:9.295000000000002switches:19
Score:299210.9568638165


```

Fig 5. The score of improved BOLA Algorithm