Report of ECE4016 HW2 --- BOLA

Name: 韩铠赢 Student ID: 121090155

1. Introduction

I choose the task 1 for this assignment, and the ABR algorithm I implemented is BOLA (BOLA: Near-Optimal Bitrate Adaptation for Online Videos. (2016)).

ABR streaming requires that each video is partitioned into chunks, where each chunk corresponds to a few seconds of play. Each chunk is then encoded in a number of different bitrates to accommodate a range of device types and network connectivities. The decision of which chunks to download at what bitrates is made by a bitrate adaptation algorithm within the video player, while each chunk has a fixed playback time of p seconds, the size of the chunk (in bits) can be different depending on its bitrate. This can be seen in the manifest file for each chunks' size. Thus, the choice of bitrate for a chunk impacts its download time.

There are mainly two types of ABR algorithms, one uses both prediction of network throughput and buffer occupancy, the other primarily bases on buffer occupancy, and BOLA is the latter one.

2. BOLA algorithm analysis

BOLA, which is fully called Buffer Occupancy based Lyapunov Algorithm, is a classic algorithm for online video streaming. BOLA is a buffer-based bitate adapative algorithm which uses Lyapunov optimization techniques to minimize rebuffering and maximize video quality.

BOLA formulate bitrate adaptation as a utility maximization problem, they define the QoE as the average bitrate of the video experienced by the user `Vn` and the duration of the rebuffer events `Sn`. And BOLA designs a control algorithm that maximizes the joint utility `Vn+γSn` subject to the constraints of the model, where `γ` is for balance between utility and smoothness. The formulas for the utility are as follows:

$$\bar{v}_N \triangleq \frac{\mathbb{E}\{\sum_{k=1}^{K_N}\sum_{m=1}^{M} a_m(t_k)v_m\}}{\mathbb{E}\{T_{\text{end}}\}}$$

$$\bar{s}_N \triangleq \frac{Np}{\mathbb{E}\{T_{\text{end}}\}} = \frac{\mathbb{E}\left\{\sum_{k=1}^{K_N}\sum_{m=1}^{M} a_m(t_k)p\right\}}{\mathbb{E}\{T_{\text{end}}\}}$$

BOLA use the buffer level for the buffer stability maxmization problem, which denotes by `Q(tk)`, and the maintain of Q(tk) can be seen in the following figure, in which the sum of `am(tk)` is the download decision (if download equals 1, otherwise 0). BOLA proves that we achieve buffer stability by minimizing $Q(t_k)(\sum_{m=1}^{M} a_m - T_k/p)$ and the performance obiective to maximize `v_bar + γ*s_bar` is achieved by maximizing $(\sum_{m=1}^{M} a_m(t_k)(v_m + \gamma p))$. In conslusion, the control decision is solving the following deterministic optimization problem:

$$Q(t_{k+1}) = \max[Q(t_k) - T_k/p,\ 0] + \sum_{m=1}^{M} a_m(t_k)$$

$$\text{Maximize } \frac{\sum_{m=1}^{M} a_m(t_k)(Vv_m + V\gamma p - Q(t_k))}{\sum_{m=1}^{M} a_m(t_k)S_m}$$

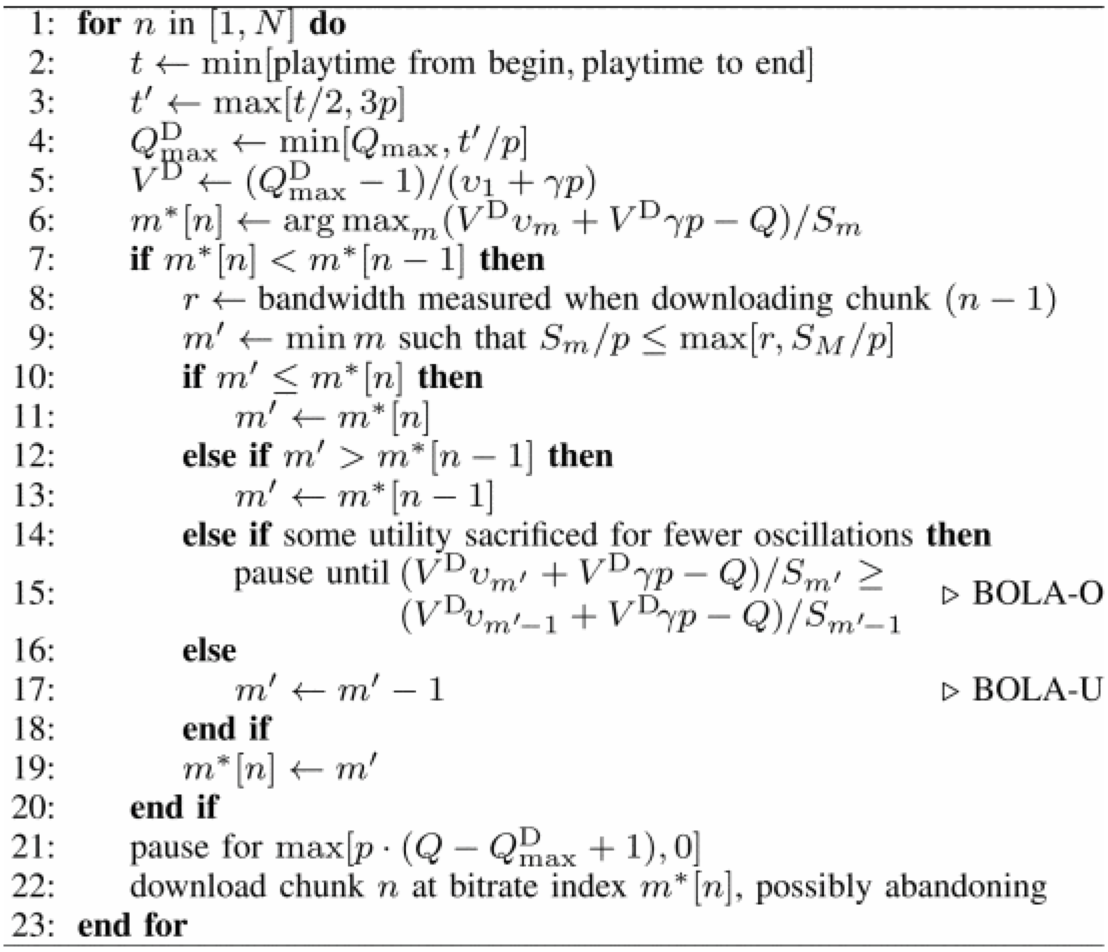
$$\text{subject to } \sum_{m=1}^{M} a_m(t_k) \leq 1, a_m(t_k) \in \{0,1\}$$

3. Implementation

To implement the ABR algorithm in the simulator, I first get to know the input parameters for the method.

```
===============================
Measured Bandwidth:  10000.0
Previous_Throughput:  9999.610667704887
Buffer_Occupancy:  {'size': 200000, 'current': 6421, 'time': 2}
Available_Bitrates,:  {'500000': 5764, '1000000': 14391, '5000000': 57155}
Video_Time 138.742
Chunk:  {'left': 2, 'time': 2, 'current': '28'}
Rebuffering_Time:  3.1369999999999996
Preferred_Bitrate None
===============================
```

After knowing the input parameters, I begin to write the code.In the paper's example, yp and V is given as the following: "let vm=ln(Sm/SM). Pick γ=5.0/p and V=0.93." And I follows this example to implement BOLA. BOLA provided psuedocode for easier implementing.



```
for $n$ in $[1, N]$ do
    $t \leftarrow \min[\text{playtime from begin, playtime to end}]$
    $t' \leftarrow \max[t/2, 3p]$
    $Q^{\mathrm{D}}_{\max} \leftarrow \min[Q_{\max}, t'/p]$
    $V^{\mathrm{D}} \leftarrow (Q^{\mathrm{D}}_{\max} - 1)/(v_1 + \gamma p)$
    $m^*[n] \leftarrow \arg\max_m (V^{\mathrm{D}} v_m + V^{\mathrm{D}} \gamma p - Q)/S_m$
    if $m^*[n] < m^*[n-1]$ then
        $r \leftarrow$ bandwidth measured when downloading chunk $(n-1)$
        $m' \leftarrow \min m$ such that $S_m/p \le \max[r, S_M/p]$
        if $m' \le m^*[n]$ 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 $(V^{\mathrm{D}} v_{m'} + V^{\mathrm{D}} \gamma p - Q)/S_{m'} \ge (V^{\mathrm{D}} v_{m'-1} + V^{\mathrm{D}} \gamma p - Q)/S_{m'-1}$    ▷ 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
```

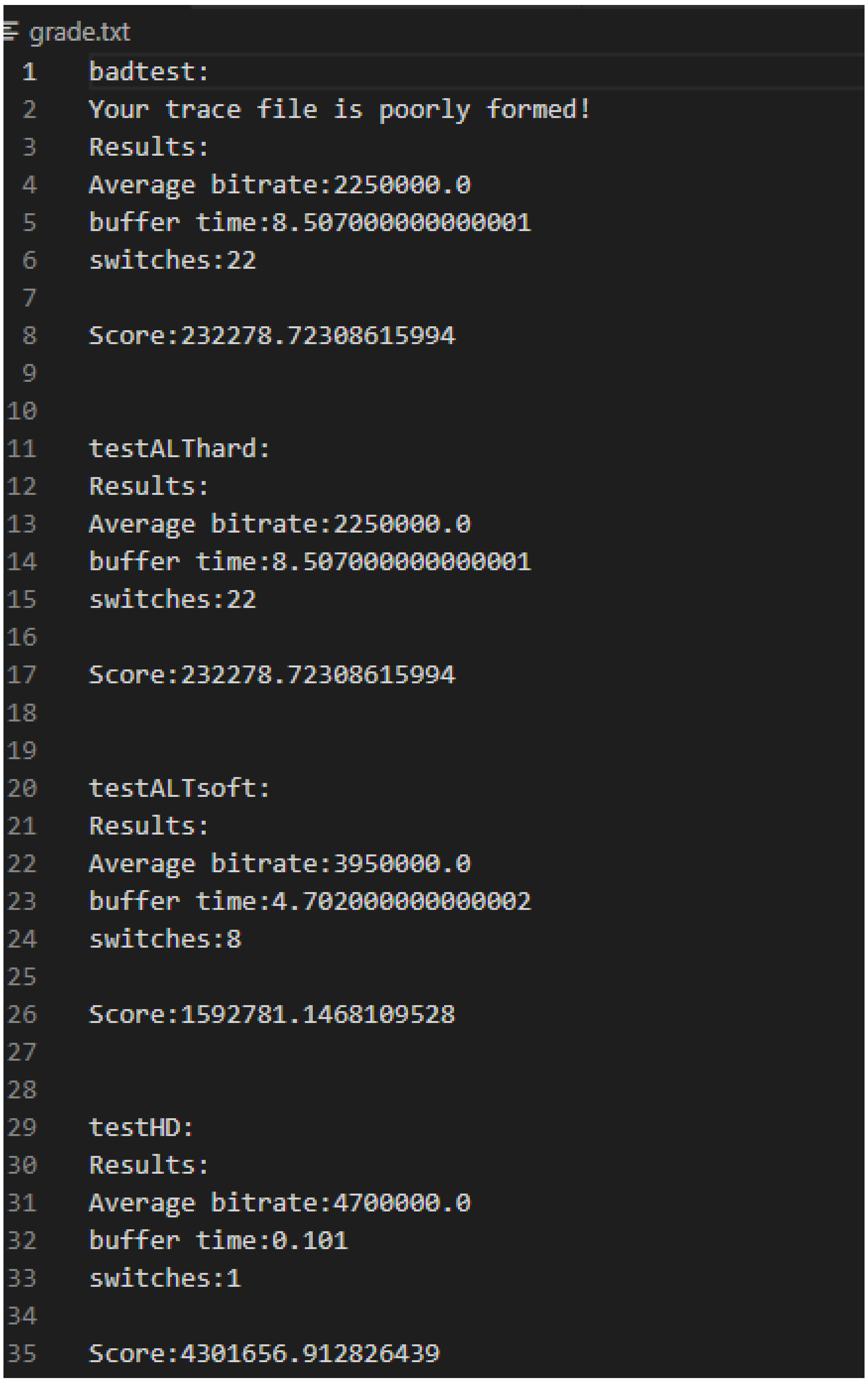
Since the provided entry is for each chunk, I do not need to do the for loop manually, and I define a global variable `prev_m` to store the previous download decision since I don't maintain a m array like the psuedocode. For the buffer size, level and minimum utility, the getting process is quite similar as the psuedocode.But for the argmax to get the optimal bitrate index, is more complex but still follows the formula in BOLA algorithm analysis.

```
# argmax index m
# calculate the optimal bitrate index that maximizes the utility function
m = max(range(len(S_m)) , key = lambda index : (V_D * math.log(int(S_m[index][0])
/ int(S_m[-1][0]), math.e) + V_D * 5 - Buffer_Occupancy['time'] / Chunk['time']) /
S_m[index][1])
```

For the next `if else` conditional judgement part, since there's no function for the simulator to pause or create oscillation, I just eliminate these parts. There are more detailed comments in the code file, please check for further information.

4. Evaluation

It's hard to conduct experiment on a simulator, but likely, the grader actually helps to evaluate the performance of the ABR algorithm. Here's my result for the grader, where the badtest, testALThard, TestALTsoft. etc has a signiificant improvement compareing with the ground truth buffer-based method:


grade.txt
```
badtest:
Your trace file is poorly formed!
Results:
Average bitrate:2250000.0
buffer time:8.507000000000001
switches:22

Score:232278.72308615994


testALThard:
Results:
Average bitrate:2250000.0
buffer time:8.507000000000001
switches:22

Score:232278.72308615994


testALTsoft:
Results:
Average bitrate:3950000.0
buffer time:4.702000000000002
switches:8

Score:1592781.1468109528


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

Score:4301656.912826439
```

5. Conclusion

This is a helpful task for learning ABR algorithm and improves my ability to imlement code from papers. And thanks for your patience for finishing readind this report.