Meta Reinforcement Learning for Rate Adaptation

I. DPPO Algorithms

The pseudocode for the distributed PPO used by Ahaggar is provided in Algorithm 1 for chief and Algorithm 2 for workers. In these algorithms, the hyperparameter KL_{target} represents the desired changes in the policy per time episode. The scaling term $\tilde{\alpha}$ controls the adjustment of the KL-regularization coefficient if the actual change in the policy stayed significantly below or significantly exceeded the target KL, *i.e.*, falls outside the interval $[\bar{\beta}_{low} \times \text{KL}_{target}, \bar{\beta}_{high} \times \text{KL}_{target}]$.

Algorithm 1 Ahaggar DPPO (Central Agent)

for Each agent $c \in \{1, \ldots, N\}$ **do**
  while not done **do**
    Wait until N gradient parameters for actor (θ_π) and critic (θ_v) are available
    Average gradients and update global θ_π and θ_v
    Update all the workers with global θ_π and θ_v
  end while
end for

Algorithm 2 Ahaggar DPPO (Workers)

for Each agent $c \in \{1, \ldots, N\}$ **do**
  while not done (for every $t = [1, \ldots, T_{\pi_\theta^c}]$) **do**
    for Each $\kappa \in \{0, \ldots, \Theta - 1\}$ **do**
      Run policy $\pi^c_{\theta_\kappa}$ and collects $\{b_t^c, a_t^c, r_t^c\}$
      Estimate discounted expected reward G_t^c
      Estimate advantages $A_t^{\pi^c_{\theta_\kappa}}$
      Store partial trajectory information
    end for
    $\pi^c_{\theta_{old}} \leftarrow \pi^c_\theta$
    Compute $\mathcal{L}^{KLPEN}_{\theta_\kappa}(\theta)$ using (**??**)
    if $\text{KL}[\pi^c_{\theta_{old}}|\pi^c_\theta] > 4\text{KL}_{target}$ **then**
      Break and continue with next time epoch $t+1$
    end if
    Compute $\bigtriangledown_\theta \mathcal{L}^{KLPEN}_{\theta_\kappa}$
    Send gradient actor parameters (θ_{π^c}) to chief
    Send gradient critic parameters (θ_{v^c}) to chief
    Wait until parameters are accepted or dropped
    Update parameters of worker c
    if $\text{KL}[\pi^c_{\theta_{old}}|\pi^c_\theta] > \bar{\beta}_{high}\text{KL}_{target}$ **then**
      $\bar{\beta} \leftarrow \tilde{\alpha}\bar{\beta}$
    else if $\text{KL}[\pi^c_{\theta_{old}}|\pi^c_\theta] < \bar{\beta}_{low}\text{KL}_{target}$ **then**
      $\bar{\beta} \leftarrow \bar{\beta}/\tilde{\alpha}$
    end if
  end while
end for

II. Network Traces

We used a total of 28 network traces for evaluation. Table I shows the summary of all the network traces which are extracted from four datasets:

1) Traces B-4G-1 to B-4G-6 are taken from Belgium 4G/LTE [5] with mobility types of bus, car, bicycle, train, foot, and train.
2) Traces N-LTE-1 to N-LTE-6 are LTE network traces from NYU LTE [2] with mobility types of bus, car, ferry, and train.
3) Traces L-4G-1 to L-4G-6 are taken from Lumous 4G [3] with mobility types of drive and walk.
4) Traces L-5G-1 to L-5G-6 are taken from Lumous 5G [3] with mobility types of drive and walk.

The bandwidth values in each trace are varied every second interval, and each client-server video session is used one of the traces as shown in Table I (1^{st} column: Session ID). We note that traces X-X-1 to X-X-6 are used for multiple clients scenario. Next, we show an example (Listing 1) of one of the traces, *i.e.*, L-4G-1, as follows:

Listing 1: An example of a network trace (L-4G-1).

```
rate 68.0mbit
wait 5s
rate 59.0mbit
wait 5s
rate 65.0mbit
wait 5s
...
```

The *tc* command (Listing 2) to throttle the bandwidth between the client and server on network interface (`eth1`) and port (`8080`) as per the network trace is described as follows:

Listing 2: An example of *tc* command.

```
echo "Setting network speed to ${rate}mbit
for ${wait}s via tc"
sudo tc qdisc add dev eth1 root handle 1:
htb default 10
sudo tc class add dev eth1 parent 1:
classid 1:1 htb rate ${rate}mbit
sudo tc class add dev eth1 parent 1:1
classid 1:10 htb rate ${rate}mbit
sudo tc qdisc add dev eth1 parent 1:10
handle 10: sfq perturb 10
sudo tc filter add dev eth1 parent 1:0
protocol ip prio 1 u32 \
match ip sport 8080 0xffff flowid 1:10
sleep ${wait}s
sudo tc qdisc del dev eth1 root
```

III. Video Dataset

We used a 4K DASH dataset from [4] for evaluation. This video dataset was not using during Ahaggar training and consists of three video samples: *Big Buck Bunny* (BBB), *Sintel* (Sintel) and *Tears of Steel* (TOS). Each video is encoded using FFmpeg (H.264/MPEG- 4 AVC codec) into 13 bitrate levels {0.24, 0.375, 0.57, 0.75, 1.05, 1.75, 2.35, 3.0, 3.85, 4.3, 15.0, 25.0, 40.0} Mbps with content resolutions {320x180, 384x216, 512x288, 512x288, 640x360, 960x540, 1280x720, 1280x720, 1920x1080, 1920x1080, 3840x2160, 3840x2160, 3840x2160} p at 30 fps. In our experiments,

we used BBB video sample with customized MPD file and a segment duration of 4 seconds. The customized MPD file includes additional meta-data for each segment, *i.e.*, segment size (in byte) and segment VMAF values for different device resolutions (Phone, HDTV, and UHDTV) and bitrate levels. To compute the VMAF values, we used two VMAF models with FFmpeg, namely: `vmaf_v0.6.1.pkl` (for Phone and HDTV) and `vmaf_ 4k_v0.6.1.pkl` (for UHDTV). Each VAMF model takes a source reference and encoded videos as input and outputs a per-frame VMAF value in a range of [0...100]. We downloaded the source reference video of BBB from Blender website (https://download.blender.org/demo/movies/BBB/) with highest possible 4K resolution (2160p) at 30 fps. For each device resolution and each encoded video with `bitrateX`, we run the following commands:

Phone:

```
ffmpeg -i bbb_encoded_bitrateX.mp4
-i bbb_reference.mp4 -filter_complex
"[0:v]scale=3840x2160:flags= bicubic[main];
[1:v]scale=3840x2160:flags=bicubic,
format=pix_fmts=yuv420p,fps=fps=30/1[ref];
[main] [ref] libvmaf=log_fmt=json:log_path=
VMAF/bitrateX_Phone.json:model_path=ffmpeg/
model/vmaf_v0.6.1.pkl:phone_model=1" -f null -
```

HDTV:

```
ffmpeg -i bbb_encoded_bitrateX.mp4 -i
bbb_reference .mp4 -filter_complex
"[0:v]scale=3840x2160:flags=
bicubic[main]; [1:v]scale=3840x2160:
flags=bicubic, format=pix_fmts=yuv420p,
fps=fps=30/1[ref]; [main][ref] libvmaf=
log_fmt=json:log_path=VMAF/bitrateX_HD.
json:model_path=ffmpeg/model/vmaf_v0.6.1
.pkl" -f null -
```

UHDTV:

```
ffmpeg -i bbb_encoded_bitrateX.mp4
-i bbb_reference.mp4 -filter_complex
"[0:v]scale=3840x2160:flags=bicubic[main];
[1:v]scale=3840x2160:flags=bicubic,
format=pix_fmts=yuv420p,fps=fps=30/1[ref];
[main] [ref] libvmaf=log_fmt=json:log_path=VMAF
/bitrateX_UHD.json:model_path=ffmpeg/model/vmaf
_4k_v0.6.1.pkl" -f null -
```

The per-frame VMAF values are stored in JSON files (*e.g.*, `bitrateX_Phone.json`, `bitrateX_HD.json`, and `bitrateX_ UHD.json`). Considering the output JSON files, we computed the per-segment VMAF value as the average of 120 (fps $\times$ segment duration = 30 $\times$ 4) per-frame VMAF values. Then, we create a customized MPD. An example of this MPD is highlighted in Listing 3.

Listing 3: An example of modified MPD used in our experiments.

```
<MPD xmlns="urn:mpeg:dash:schema:mpd:2011"
    minBufferTime="PT1.500S" type="static"
    mediaPresentationDuration="PT0H10M34.600S"
    maxSegmentDuration="PT0H0M4.000S" profiles="
    urn:mpeg:dash:profile:full:2011">
<Period duration="PT0H10M34.600S">
   <AdaptationSet segmentAlignment="true"
       maxWidth="3840" maxHeight="2160"
       maxFrameRate="30" par="16:9" lang="und"
       startWithSAP="1">
      <Representation id="1" mimeType="video/mp4"
           codecs="avc1.64000D" width="320"
          height="180" frameRate="30" sar="1:1"
          bandwidth="227077">
         <SegmentList duration="61440" timescale=
             "15360" initialization="240k/init.
              mp4" startNumber="1">
            <Initialization sourceURL="240k/init.
                mp4"/>
            <SegmentURL media="240k/1.m4s" size="
                74542" phone="5" hdtv="4" uhdtv="
                15"/>
            <SegmentURL media="240k/2.m4s" size="
                130980" phone="11" hdtv="10"
                uhdtv="16"/>
            ...
```

The characteristics of BBB video sample including bitrate levels, content resolution, per-segment VMAF values and sizes for different device resolutions are highlighted in Table II.

IV. Artifacts

Our code and its materials (including network and video datasets) are publicly available in [1]. Instructions on setup are given in README.

References

[1] Anonymous. Ahaggar Bitrate Guidance. [Online] Available: tobeadded. Accessed on June 25, 2022.

[2] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, and J. Li. Realtime Mobile Bandwidth Prediction Using LSTM NN. In *Springer PAM*, 2019.

[3] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, , et al. A Variegated Look at 5G in the Wild: Performance, Power, and QoE Implications. In *ACM SIGCOMM*, 2021.

[4] J. J. Quinlan and C. J. Sreenan. Multi-profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets. In *ACM MMSys*, 2018.

[5] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters*, 20(11):2177–2180, Nov. 2016.

TABLE I: Characteristics of network traces used in the experiments. StD: Standard Deviation.

Session ID	Trace Name	Mobility Type	Inter-variation Duration (s)	Min. Bandwidth (Mbps)	Max. Bandwidth (Mbps)	Mean Bandwidth (Mbps)	StD Bandwidth (Mbps)
Belgium 4G/LTE [5]							
S1	B-4G-1	Bus	5	0.5	7.88	4.11	1.50
S2	B-4G-2	Car	5	0.5	7.96	4.02	1.97
S3	B-4G-3	Bicycle	5	0.5	7.48	3.95	1.57
S4	B-4G-4	Bus	5	0.5	7.00	3.45	1.29
S5	B-4G-5	Car	5	0.5	12.88	4.49	2.68
S6	B-4G-6	Train	5	0.5	7.57	2.92	1.69
NYU LTE [2]							
S1	N-LTE-1	Bus	5	1.19	31.0	9.08	5.27
S2	N-LTE-2	Car	5	0.5	26.2	10.33	6.82
S3	N-LTE-3	Ferry	5	0.69	27.7	11.0	6.47
S4	N-LTE-4	Train	5	0.66	23.0	10.70	4.94
S5	N-LTE-5	Car	5	0.77	37.3	12.24	9.14
S6	N-LTE-6	Train	5	1.01	21.7	7.37	4.22
Lumous 4G [3]							
S1	L-4G-1	Drive	5	0.5	264.0	38.76	41.07
S2	L-4G-2	Walk	5	0.5	194.0	27.81	28.32
S3	L-4G-3	Drive	5	0.5	157.0	33.36	25.87
S4	L-4G-4	Walk	5	2.0	198.0	60.02	33.85
S5	L-4G-5	Drive	5	0.5	137.0	33.90	25.16
S6	L-4G-6	Walk	5	2.5	176.0	84.96	29.39
Lumous 5G [3]							
S1	L-5G-1	Drive	5	0.5	969.0	306.10	276.65
S2	L-5G-2	Walk	5	0.5	1750.0	587.63	456.73
S3	L-5G-3	Drive	5	0.5	1720.0	535.05	525.36
S4	L-5G-4	Walk	5	0.5	1010.0	341.12	278.29
S5	L-5G-5	Drive	5	0.5	1840.0	663.46	525.65
S6	L-5G-6	Walk	5	0.5	1860.0	704.90	584.89

TABLE II: Characteristics of BBB content.

Bitrate Level (Mbps)	Content Resolution (p)	Min. VMAF	Max. VMAF	Mean VMAF	StD VMAF	Min. Size (MB)	Max. Size (MB)	Mean Size (MB)	StD Size (MB)
Phone									
0.24	320x180	0	37	4.16	7	0.05	0.14	0.11	0.01
0.375	384x216	0	47	9.70	11	0.07	0.20	0.18	0.02
0.57	512x288	0	68	26.86	18	0.10	0.31	0.26	0.03
0.75	512x288	0	71	29.40	19	0.10	0.41	0.35	0.04
1.05	640x360	0	82	47.72	21	0.12	0.57	0.48	0.06
1.75	960x540	45	94	76.30	10	0.18	0.95	0.76	0.11
2.35	1280x720	62	98	88.06	6	0.22	1.21	0.99	0.14
3.00	1280x720	67	99	90.26	5	0.24	1.55	1.26	0.19
3.85	1920x1080	77	100	95.42	4	0.28	1.94	1.54	0.24
4.30	1920x1080	79	100	96.17	4	0.29	2.20	1.72	0.28
15.0	3840x2160	95	100	99.89	0.5	0.56	7.98	5.95	1.04
25.0	3840x2160	100	100	0.00	0	0.75	13.18	9.96	1.78
40.0	3840x2160	100	100	0.00	0	0.91	21.05	15.95	2.94
HDTV									
0.24	320x180	0	28	2.66	5	0.05	0.14	0.11	0.01
0.375	384x216	0	33	5.97	8	0.07	0.20	0.18	0.02
0.57	512x288	0	50	17.13	13	0.10	0.31	0.26	0.03
0.75	512x288	0	53	18.93	13	0.10	0.41	0.35	0.04
1.05	640x360	0	65	32.50	16	0.12	0.57	0.48	0.06
1.75	960x540	39	79	57.52	11	0.18	0.95	0.76	0.11
2.35	1280x720	43	86	72.27	8	0.22	1.21	0.99	0.14
3.00	1280x720	48	87	74.16	7	0.24	1.55	1.26	0.19
3.85	1920x1080	57	100	82.92	8	0.28	1.94	1.54	0.24
4.30	1920x1080	60	100	84.21	7	0.29	2.20	1.72	0.28
15.0	3840x2160	82	100	96.19	2	0.56	7.98	5.95	1.04
25.0	3840x2160	92	100	97.96	1	0.75	13.18	9.96	1.78
40.0	3840x2160	96	100	98.56	1	0.91	21.05	15.95	2.94
UHDTV									
0.24	320x180	0	39	17.58	8	0.05	0.14	0.11	0.01
0.375	384x216	0	48	24.29	10	0.07	0.20	0.18	0.02
0.57	512x288	3	63	37.31	12	0.10	0.31	0.26	0.03
0.75	512x288	4	65	38.72	12	0.10	0.41	0.35	0.04
1.05	640x360	21	74	50.82	11	0.12	0.57	0.48	0.06
1.75	960x540	50	85	69.81	8	0.18	0.95	0.76	0.11
2.35	1280x720	60	91	79.96	6	0.22	1.21	0.99	0.14
3.00	1280x720	63	92	81.91	5	0.24	1.55	1.26	0.19
3.85	1920x1080	71	100	88.43	5	0.28	1.94	1.54	0.24
4.30	1920x1080	73	100	89.30	5	0.29	2.20	1.72	0.28
15.0	3840x2160	87	100	98.33	2	0.56	7.98	5.95	1.04
25.0	3840x2160	94	100	99.53	1	0.75	13.18	9.96	1.78
40.0	3840x2160	98	100	99.88	0.3	0.91	21.05	15.95	2.94