ECE 50863
Project 3 Report Template

Name and email: Yuan-Yao Lou / yylou@purdue.edu
Collaborator (if applicable):

Submit as a pdf, labeled <firstname.lastname.Project3.first.pdf>,
<firstname.lastname.Project3.final.pdf>

Collaborations should be restricted to brainstorming ideas, and sharing experiences, but each student should implement her/his own code, and the report should be individual

Instruction

- Please follow this template for all milestones, but update the material with each milestone.
- Answers should be typed. Handwritten documents are not permitted.
- Your report should be submitted in pdf format only.
- Please keep your answers to the point.
- **Graphs should be carefully plotted, with the X and Y axis clearly labeled with appropriate legends.**
- **For a slide with a graph have a 1 line “take-home” message (what does the result show?)**
- Avoid 2 graphs on the same slide to ensure clarity.

Base Algorithm 1

- What base algorithm did you implement? **RobustMPC**
- Present a bulleted list of the key details of the implementation of the base algorithm. The description should be self-sufficient to see all your key decisions, but please keep it succinct.
  - **Calculate QoE score of total quality** for all possible quality sequences
  - **Calculate QoE score of quality variation** for all possible quality sequences
  - **Use "previous_throughput" as the throughput** for the 5 look-ahead windows
  - **Get chunk size sequences for all possible quality sequences** by "quality_bitrates" & "upcoming_quality_bitrates"
  - **Calculate download time** by the throughput and chunk size sequences
  - **Calculate rebuffer time** by download time and "buffer_seconds_until_empty"
  - **Calculate the composite QoE score** for all possible quality sequences to make the decision

Variant of Base Algorithm 1

- Discuss the variant of the base algorithm that you implemented. If a standard variant (BBA-0 or BBA-1, FastMPC), this can be brief. But if a more novel variant, expand a bit more on what the variant seeks to achieve, and provide more details.
  - When calculating the QoE score of quality variation for all possible quality sequences, I also **consider the previous chunk's quality**
  - Since rebuffer time has the highest penalty coefficient, **I add a new metric "amount of client buffer" into consideration when choosing the quality of the next chunk**
  - **I extend the look-ahead window size to 9** for calculating the QoE score for all possible quality sequences (not choosing 10 due to much longer execution time)
  - By setting the look-ahead window size to ~10, **I set the QoE coefficient of "amount of client buffer" to 2** (same as the total quality). This is because the MAX client buffer size now approximately equals MAX chunk quality (i.e., [3, 3, 3, 3, 3, 3, 3, 3, 3]).

Results: Base Algorithm 1 and Variant

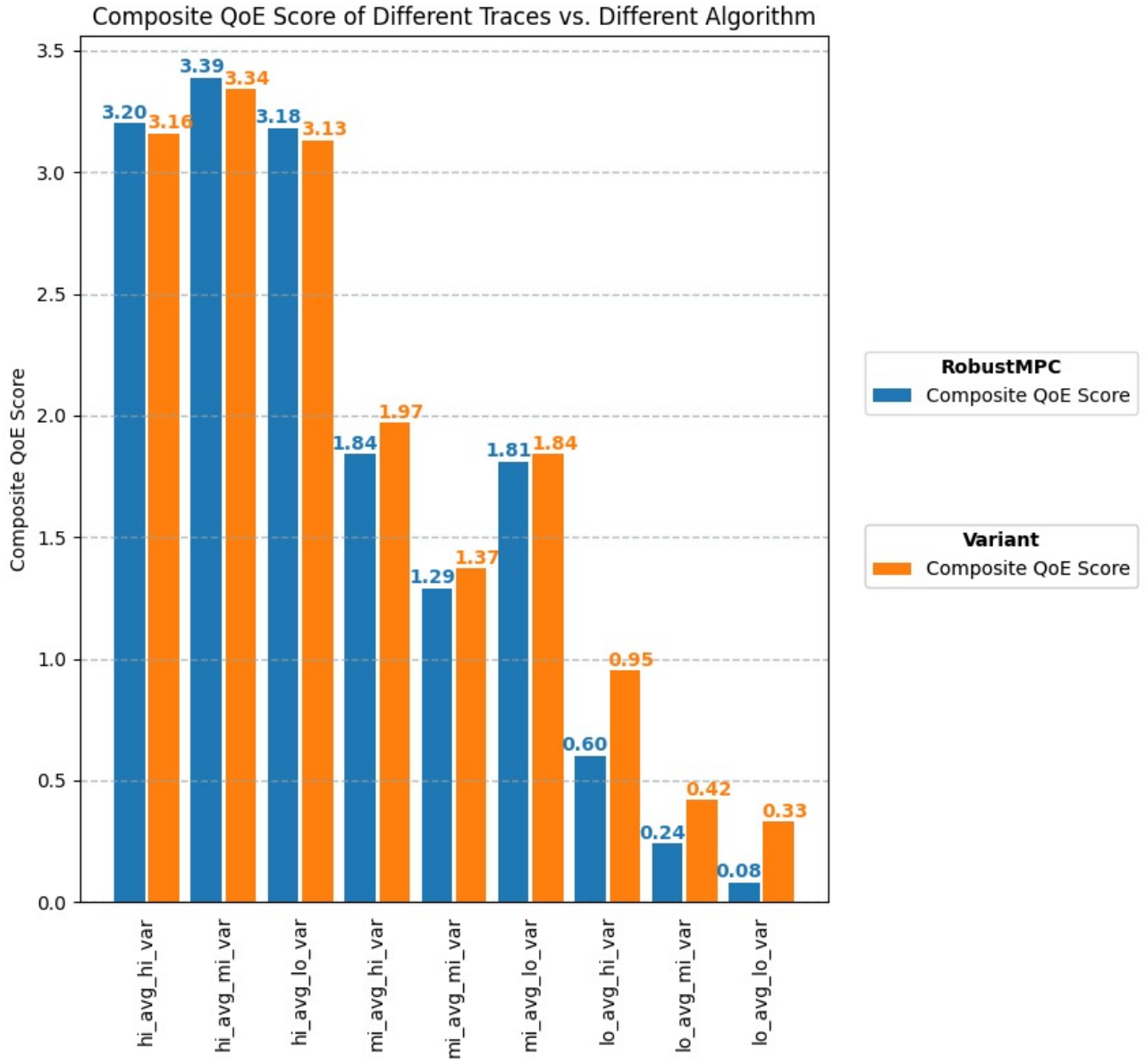
- Present **comparison results of BaseAlgorithm1 and its variant** on the set of traces/configurations that we provided.
- Please give some thought to **what graphs to present** in as compact yet informative/insightful a manner as possible.
- Reporting the composite score is useful, but it may be helpful to also **report other metrics especially when they can help provide more insights.**

Composite Score

The variant performs slightly worse than the base algorithm (RobustMPC) in “hi_avg_*” traces.

The variant performs slightly better than the base algorithm (RobustMPC) in “mi_avg_*” traces.

The variant performs significantly better than the base algorithm (RobustMPC) in “lo_avg_*” traces.

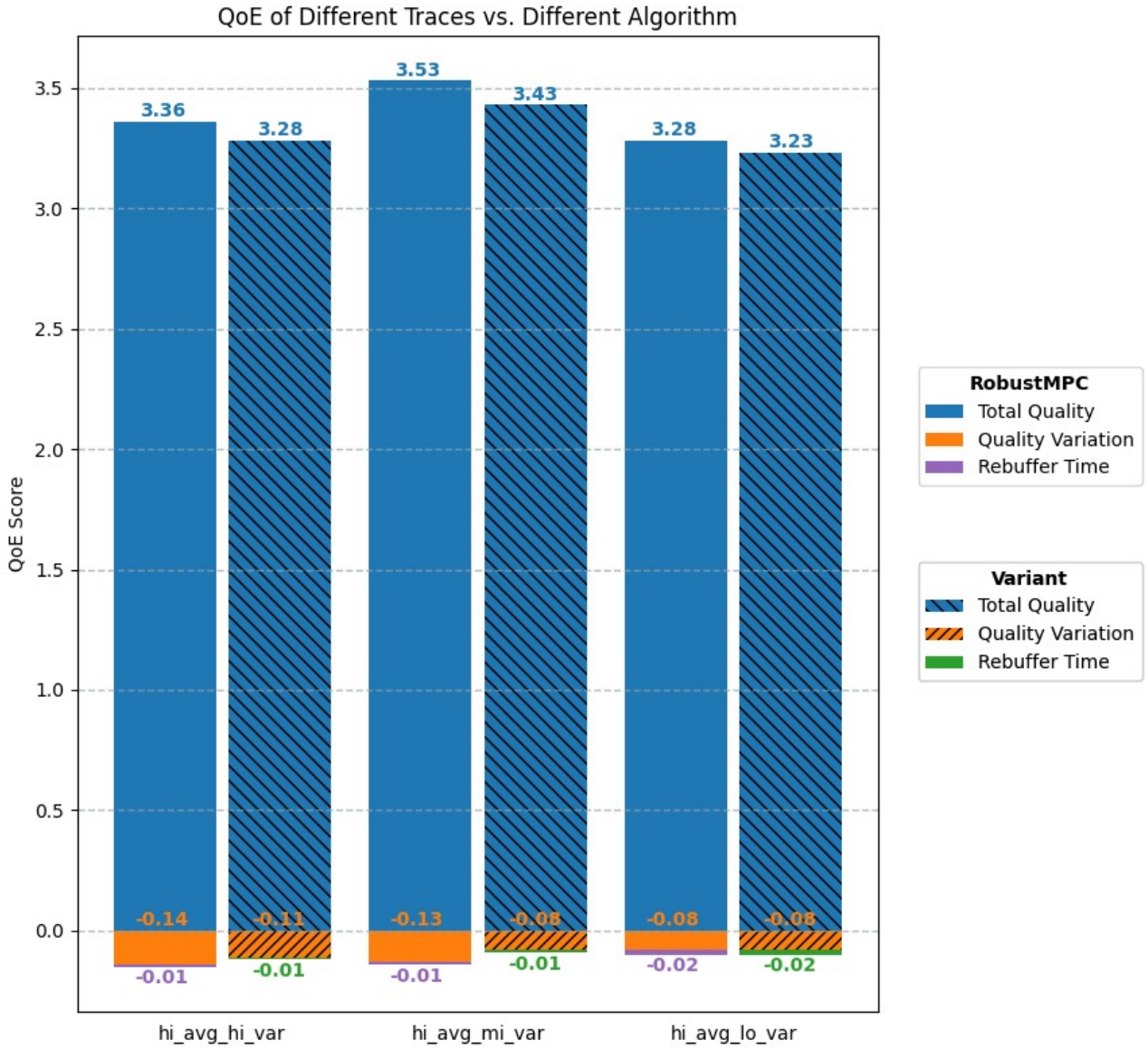


Individual Score: “hi_avg_*” traces

The variant performs slightly worse than the base algorithm (RobustMPC).

The variant gets:

- Lower QoE score of total quality (from 0.05 to 0.1)
- Slightly higher QoE score of quality variation (from 0.03 to 0.05)

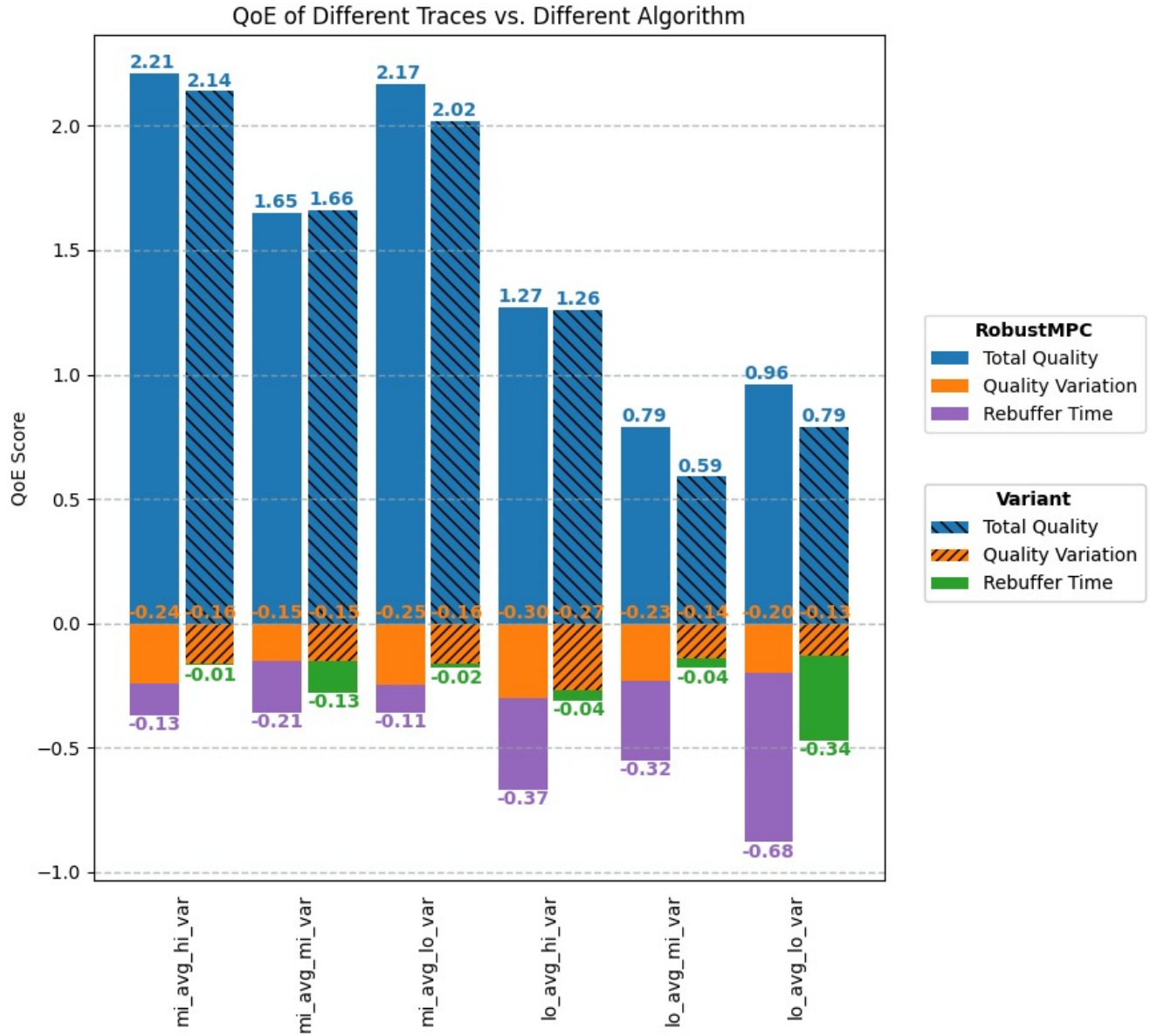


Individual Score: “lo_avg_* & mi_avg_*” traces

The variant performs better than the base algorithm (RobustMPC).

The variant gets:

- Lower QoE score of total quality (from 0.01 to 0.17)
- Slightly higher QoE score of quality variation (from 0.03 to 0.12)
- Much higher QoE score of rebuffer time (from 0.08 to 0.34)



For interim report, you may stop here

For final report, update slide deck to..

- Discuss the **details of the second algorithm** implemented
- Discuss the **variant of the second algorithm** you implemented: rationale for it and details.
- Present results first **comparing the second algorithm and its variant.**
- Then, present results comparing the better performing variant of the first algorithm, and the better performing variant of the second algorithm **(you may compare all algorithms and their variants if the results can be presented compactly).**
- **Update your discussions of results in the interim report** to cover both algorithms and variants

Base Algorithm 2

- What base algorithm did you implement? **BBA-2**
- Present a bulleted list of the key details of the implementation of the base algorithm. The description should be self-sufficient to see all your key decisions, but please keep it succinct.
  - **Use previous throughput as the calculation of the delta of buffer** in the startup phase
  - **Increase video rate in the startup phase** if the delta of buffer is larger than 0.875
  - **Terminate startup phase** if the buffer is decreasing or the chunk map suggests higher rate
  - **Dynamically adjust the reservoir** according to next X (i.e., size of playout buffer) seconds
  - **Adopt chunk map** instead of rate map
  - **Increase/Decrease video rate** if the chunk size suggested by chunk map pass R_minus/R_plus

Variant of Base Algorithm 2

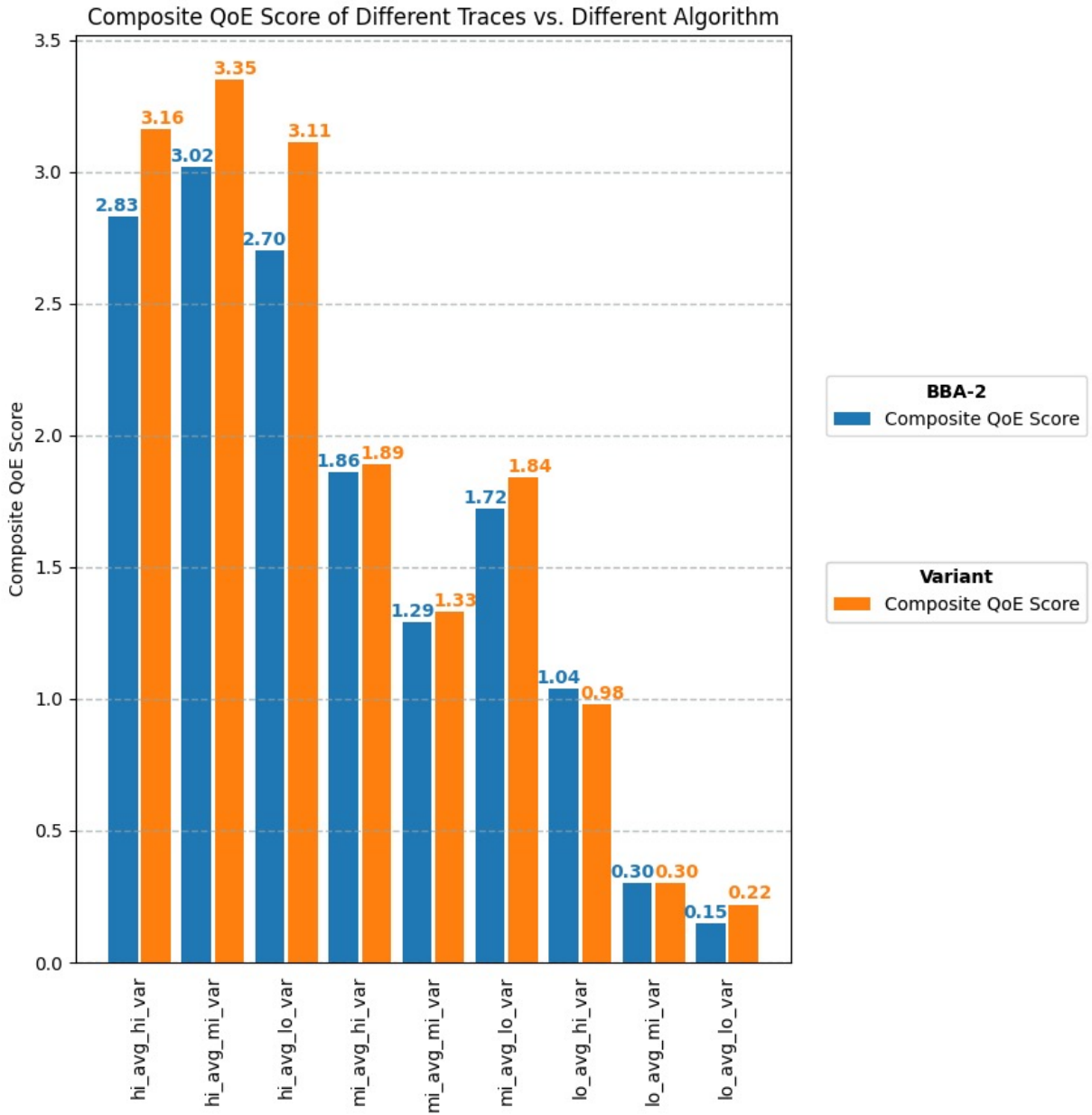
- Discuss the variant of the base algorithm that you implemented. If a standard variant (BBA-0 or BBA-1, FastMPC), this can be brief. But if a more novel variant, expand a bit more on what the variant seeks to achieve, and provide more details.
  - **Modify the threshold of delta buffer to 0.45 instead of 0.875** due to different environments (compared with Netflix's environments)
  - **Use the chunk size of the lowest available video rate as the reservoir size** since if current buffer is larger than this value, we could safely pickup the rate larger than R_min
  - **Modify upper reservoir size** using the factor 0.4 since the playout buffer size is only 30 seconds in this project, if use 0.9 as the factor, it would be too conservative.
  - **Increase my RobustMPC variant's coefficient of total quality and the current buffer size to 4**
  - **Replace the linear mapping with my RobustMPC variant within the cushion region** since my RobustMPC variant considers the current buffer size and thus could help to safely ramp up the amount of buffer while maintaining relative high video quality

Results: Base Algorithm 2 and Variant

- Present **comparison results of BaseAlgorithm2 and its variant** on the set of traces/configurations that we provided.
- Please give some thought to **what graphs to present** in as compact yet informative/insightful a manner as possible.
- Reporting the composite score is useful, but it may be helpful to also **report other metrics especially when they can help provide more insights.**

Composite Score

The variant performs better than the base algorithm (BBA-2) in all traces, except the “lo_avg_hi_var” trace.

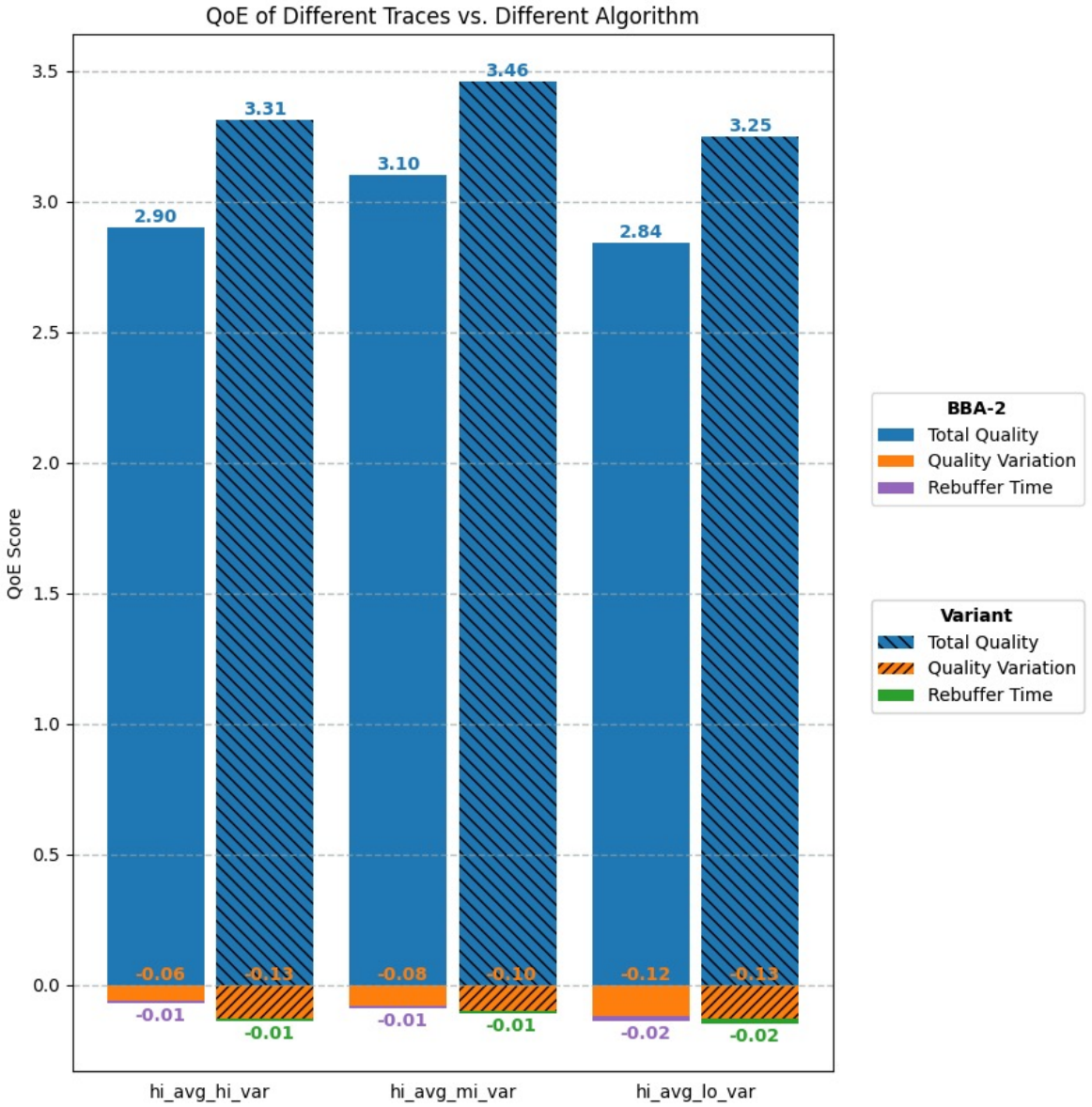


Individual Score: “hi_avg_*” traces

The variant performs better than the base algorithm (BBA-2).

The variant gets:

- Much higher QoE score of total quality (from 0.36 to 0.41)
- Slightly lower QoE score of quality variation (from 0.01 to 0.07)
- Same QoE score of rebuffer time

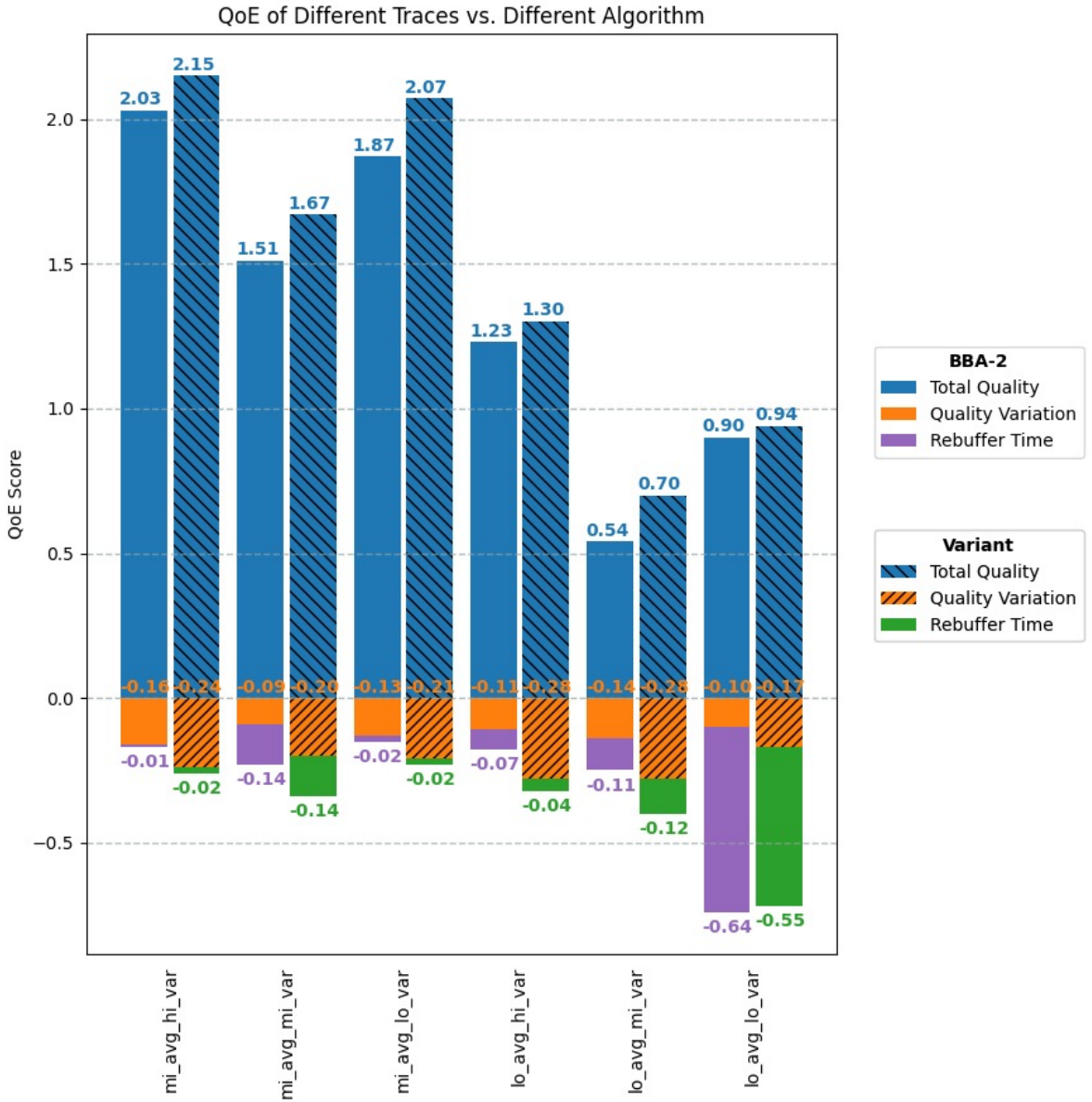


Individual Score: “lo_avg_* & mi_avg_*” traces

The variant performs better than the base algorithm (BBA-2), except “lo_avg_hi_var” trace.

The variant gets:

- Higher QoE score of total quality in all listed traces (from 0.04 to 0.20)
- Lower QoE score of quality variation in all listed traces (from 0.07 to 0.17)
- Slightly higher QoE score of rebuffer time in “lo_avg_hi_var” and “lo_avg_lo_var” traces (from 0.03 to 0.09)
- Slightly lower QoE score of rebuffer time in “mi_avg_hi_var” and “lo_avg_mi_var” traces (0.01)

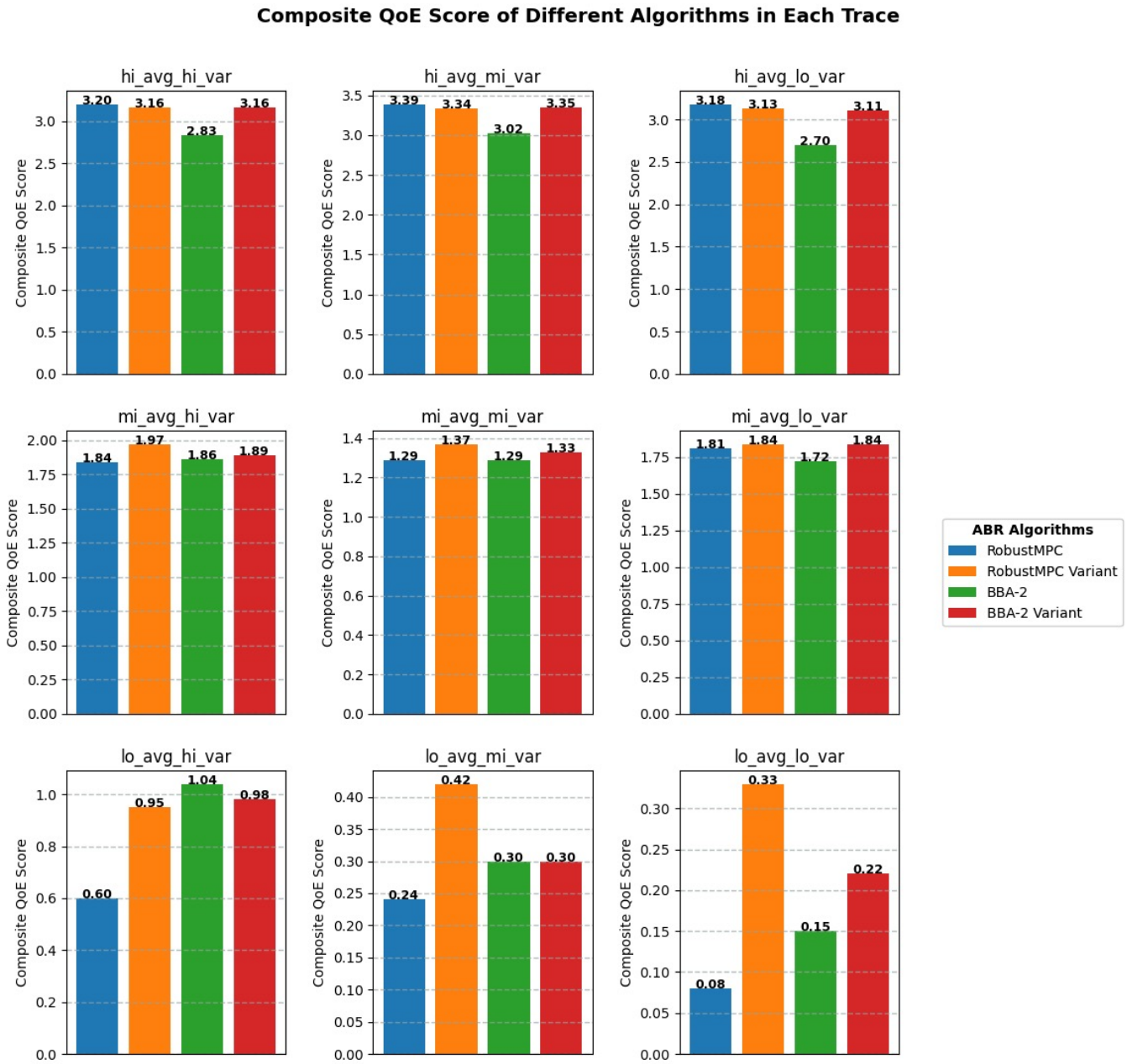


Composite Score

RobustMPC performs the best in the "hi_avg_*" traces.

RobustMPC variant performs the best in the "mi_avg_*" and "lo_avg_*" traces, except the "lo_avg_hi_var trace".

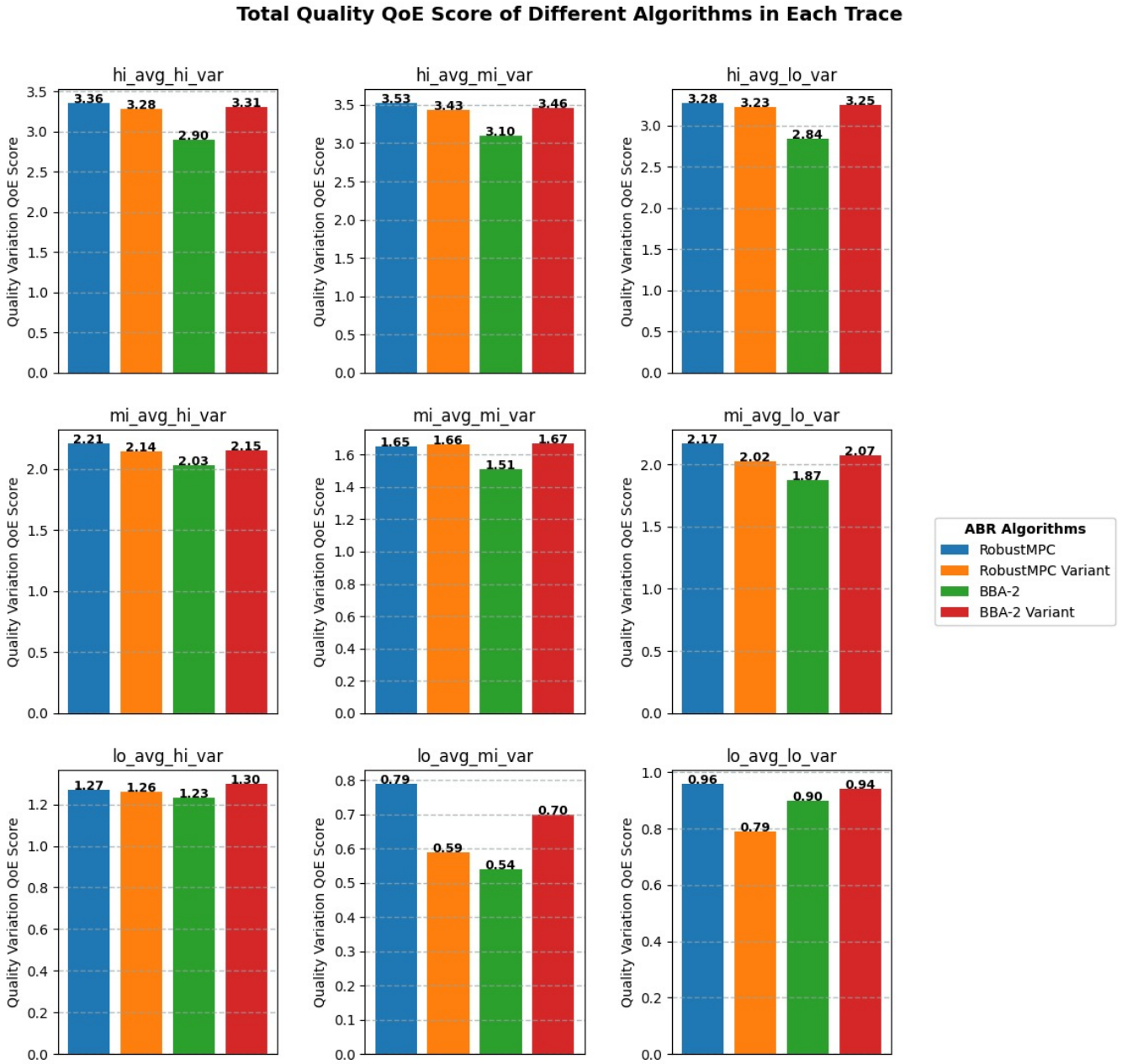
BBA-2 performs the best in the "lo_avg_hi_var" trace.



Individual Score: Total Quality

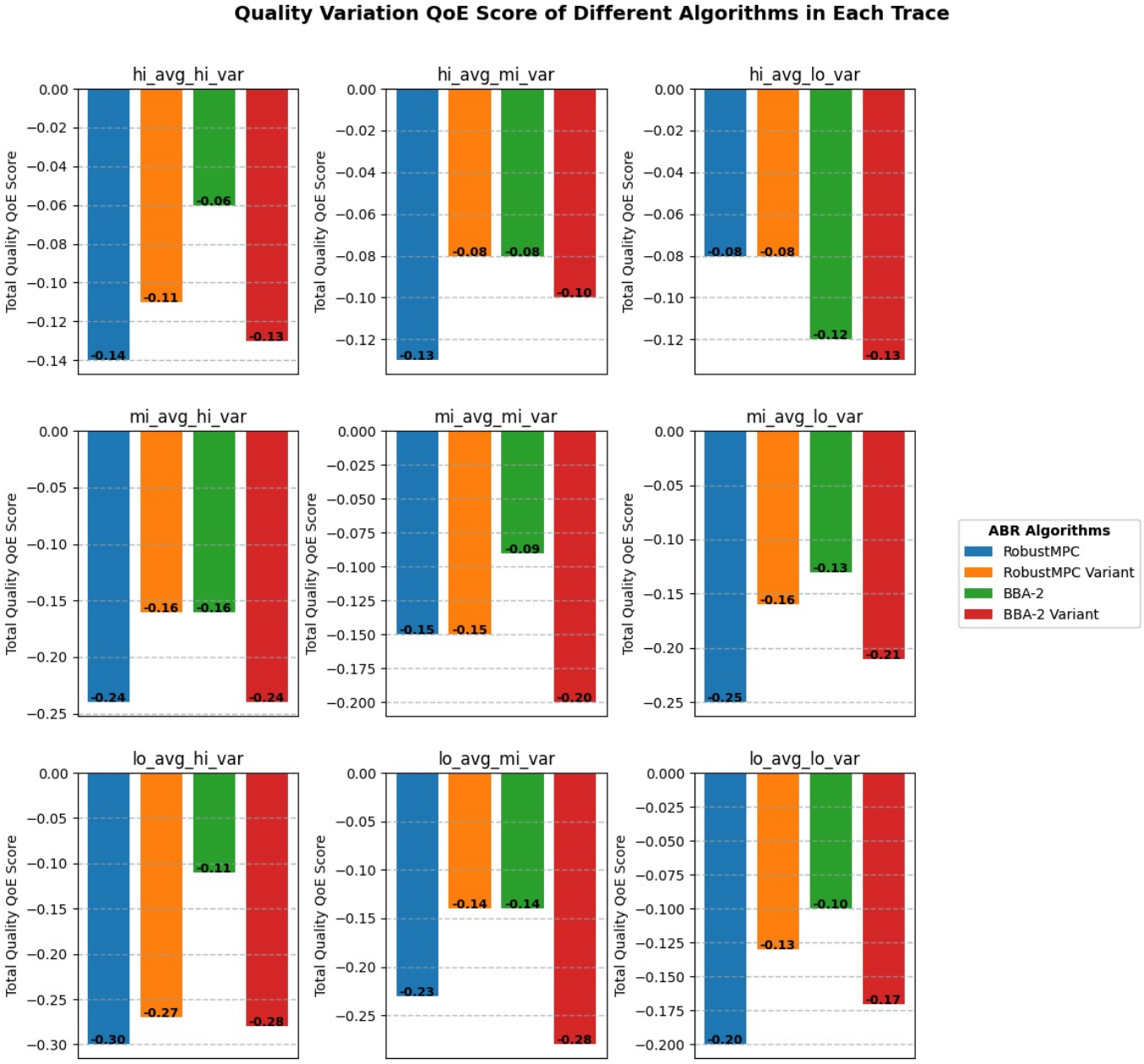
RobustMPC performs the best in all traces, except “mi_avg_mi_var” and “lo_avg_hi_var” traces.

BBA-2 variant performs the best in the “mi_avg_mi_var” and “lo_avg_hi_var” traces.



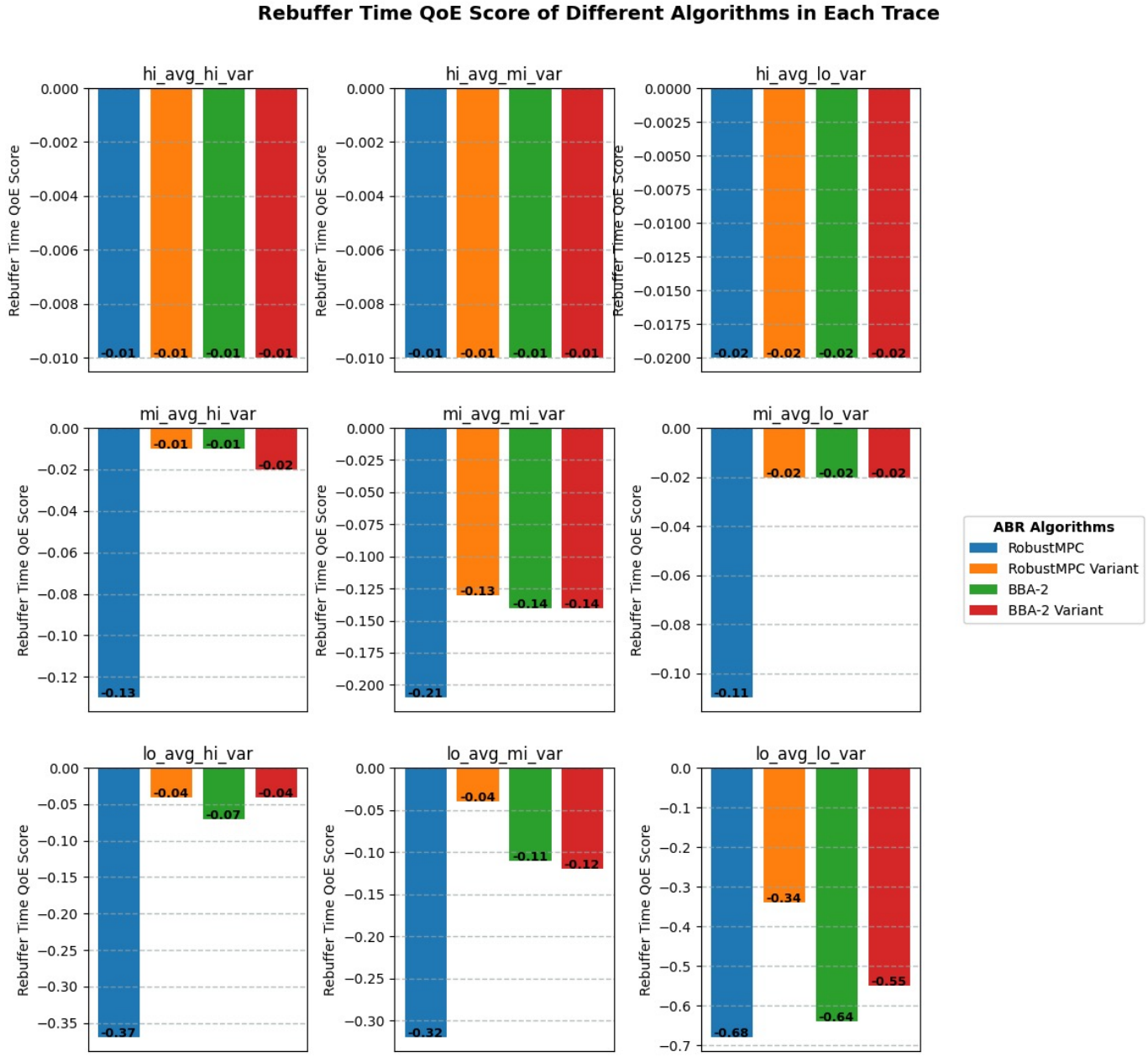
Individual Score: Quality Variation

BBA-2 performs the best in all traces, except the “hi_avg_lo_var” trace



Individual Score: Rebuffer Time

RobustMPC variant performs the best in the all traces.



Results Discussion

- What are your conclusions?
  - Does an algorithm (or variant) **generally work better in all cases?**
  - Do the algorithms have **trade-offs,** with one working better in certain environments than others?
  - Do some algorithms **favor one metric (e.g. quality) more than others?**
  - In **what settings does an algorithm perform the best or the worst, and why?**
- Please do not just say “Algorithm 1 performs better In Trace 1 and 2, while Algorithm 2 performs better in Traces 3 to 5”. Instead, seek to generalize and see what characteristics those traces have, and try to connect it to how the algorithm works.

Results Discussion

- **Does an algorithm (or variant) generally work better in all cases? Do the algorithms have trade-offs, with one working better in certain environments than others?**

  RobustMPC variant generally works better in all cases, however, each algorithm does have trade-offs in different traces.

  Take **RobustMPC** as an example, as the previous slide shows, it performs the best in “hi_avg_*” traces since it tends to be aggressive in obtaining a high total quality, and higher throughput could support this target. However, it generally performs the worst in all other traces since the average throughput now becomes lower.

Results Discussion

- **Do some algorithms favor one metric (e.g. quality) more than others?**

  For the metric “Rebuffer Time”, the **RobustMPC variant** performs the best of all different algorithms in all traces. This is because the **RobustMPC variant** considers the amount of buffer in the prediction scheme with a large look-ahead window. Thus, the **RobustMPC variant** optimizes the amount of buffer in all possible quality combinations and further avoids rebuffering events.

- **In what settings does an algorithm perform the best or the worst, and why?**

  I’d like to discuss the reservoir setting in the BBA-2 and its variant. Apparently, the fixed size performs the worst. Other than that, I found out the method described in the paper to dynamically adjust reservoir size doesn’t work the best. This is because, in this project, the playout buffer is much smaller, also, the video length is much shorter.

  Thus, I use the chunk size of the next lowest available video rates as the reservoir size. According to the results, this way performs the best compared to other settings.

Other open-ended explorations

- **Discuss any other variants you may have implemented/explored and summarize results.**

  During implementing the BBA-2 variant, I tried to adopt the RobustMPC variant in the startup phase since I'd like to quickly fill up the buffer to enter the safe area. However, I got the same composite QoE score. I think it is because the trend of all traces is easily matched with the termination condition of the startup phase.

  Besides, I also make a variant that goes back to the startup phase when the current buffer is less than reservoir size. The motivation is to imitate the congestion control algorithm. However, the results are slightly worse. I think if my algorithm gets more aggressive when entering into the startup phase again, I might get better results.

- If you came up with additional traces of your own to better distinguish different algorithms, discuss the rationale, and what you found.