ECE 50863
Project 3 Report Template

Name and email: Junyoung Kim
Kim3722@purdue.edu

Instructions

- Please follow this template for all milestones, but update the material with each milestone.
- For checkpoint, comparisons only need to present implemented algorithms.
- 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: RobustMPC

Feature	What you did? Focus on your implementation, and mainly how it might have simplified/modified what was in the paper. Detailed description of paper not needed.
What predictor did you use?	I used a simple predictor that uses the last observed throughput. New throughput = past throughput if past throughput > 0 else 1000.0
How did you adjust for prediction error?	I did not implement anything that will consider prediction error. However, the algorithm handles this error by simulated lookahead and re-evaluating plan at every chunk.
What look-ahead window did you use?	I used 5 which is recommended from the paper.
How do you solve the optimization for each look-ahead ?	The algorithm calculates all possible combinations of quality levels for N chunks and each sequence of N chunks is evaluated with QoE equation chunk by chunk.
Do you recompute bitrate choices at each chunk or every few chunks? [E.g., chunks 1-5, 2-6, 3-7 etc. or 1-5,6-10,11-15]	The algorithm recomputes at every new chunk using the sliding window approach.

BaseAlgorithm: RobustMPC

Use a bulleted list to explain any thing else you think important for the base implementation not covered in the previous slide. Include any simplifications you made, or other significant features you implemented

- This algorithm assumes that the bandwidth stays constant for all chunks in the lookahead window. This reduces the complexity, but it does not correctly represent real world network.
- The optimization does not consider which chunks are cheap and which are not. This implementation reduces the potential performance.
- The optimization considers all possible sequences, when there might be obvious bad sequences that do not need to be estimated.

Base Algorithm: BBA-2

Feature	What you did? Focus on your implementation, and mainly how it might have simplified/modified what was in the paper. Detailed description of paper not needed.
How was initial period handled to avoid being conservative?	The algorithm uses reservoir threshold. If the buffer is below this, it will pick the lowest quality. So, it will not handle conservative at initial period.
How did you map buffer levels to chunk sizes?	The algorithm uses a cushion zone and linearly maps the buffer level to the quality level. This keeps the transition smooth and prevent quality oscillations.

BaseAlgorithm: BBA-2

Use a bulleted list to explain any thing else you think important for the base implementation not covered in the previous slide. Include any simplifications you made, or other significant features you implemented

- BBA-2 only considers buffer level and ignores other information available. It might be better if it starts to consider more information available.
- Reservoir and cushion is a constant value and not adapting to changing conditions.
- No lookahead or future awareness were used which makes ba ig difference compared to previous Robust MPC.

Your approach [Use more slides if needed]

For checkpoint, this is a proposed approach. For final replace with what you actually did.

- What is the goal of your approach and why do you think it might do better?

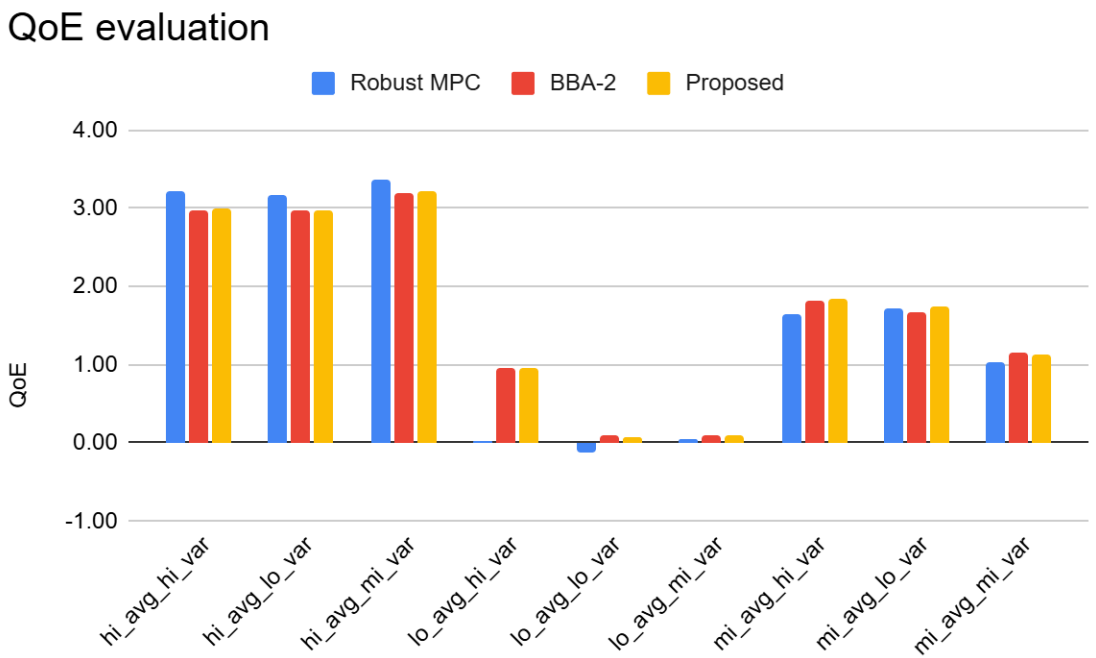
My first approach was to upgrade Robust MPC algorithm as I thought it makes more sense to try to predict the future conditions and make plans accordingly. However, after few tests, it was clear that the MPC approach struggled a lot on low average test cases as it had a very high rebuffer time. I aimed to create an algorithm that performs well throughout the different scenarios by adding both algorithms together.

- Provide key details of your approach. [Avoid low-level implementation details] Justify key choices. Use a bulleted list.
  - Added buffer consideration on top of Robust MPC so when low buffer it will act like BBA-2.
  - To have accurate bandwidth estimation, added confidence checker using throughput history.
  - Linearly penalized buffer estimation when low confidence or recently had rebuffer period.
  - Added weight system to prefer cheaper high-quality chunks when evaluating QoE to increase performance on MPC part.

Results: [More slides if needed]

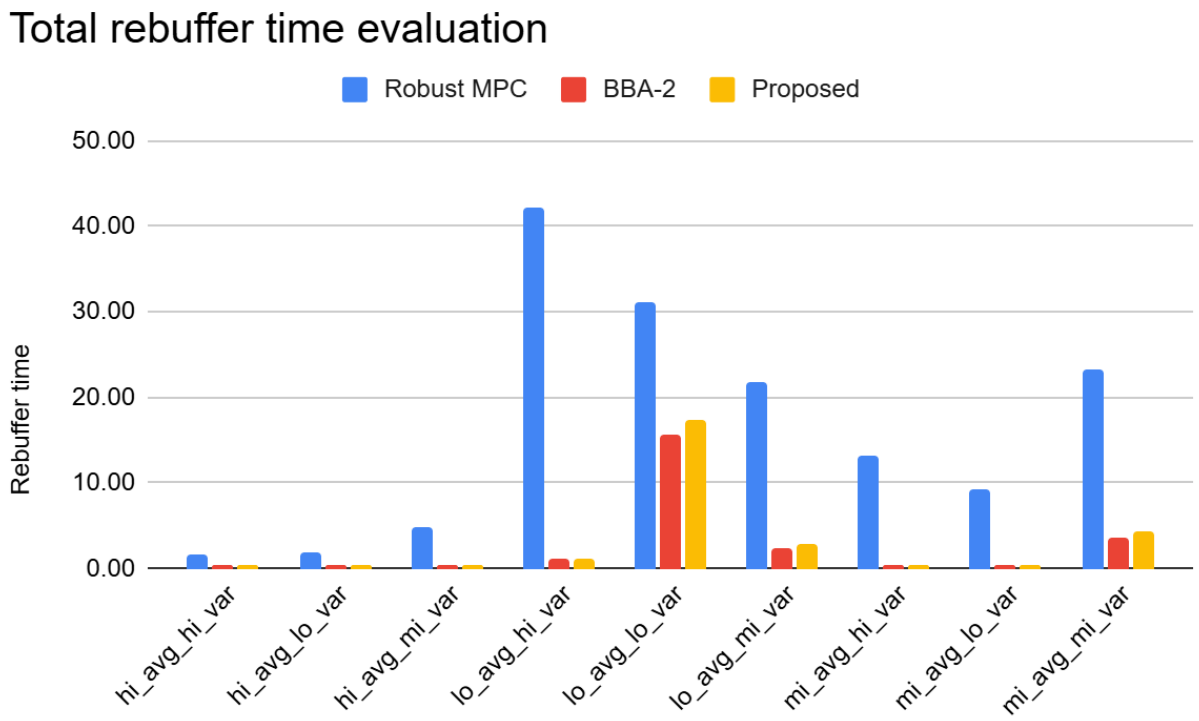
[For checkpoint, only focus on baseline algorithms, you can update for final]

- Present the settings where you compared?
  - All of the given testing files.
  - Lookahead = 5, reservoir = 4, cushion = 10 for all cases.
- Present comparison results of all algorithms on the set of traces/configurations that we provided.



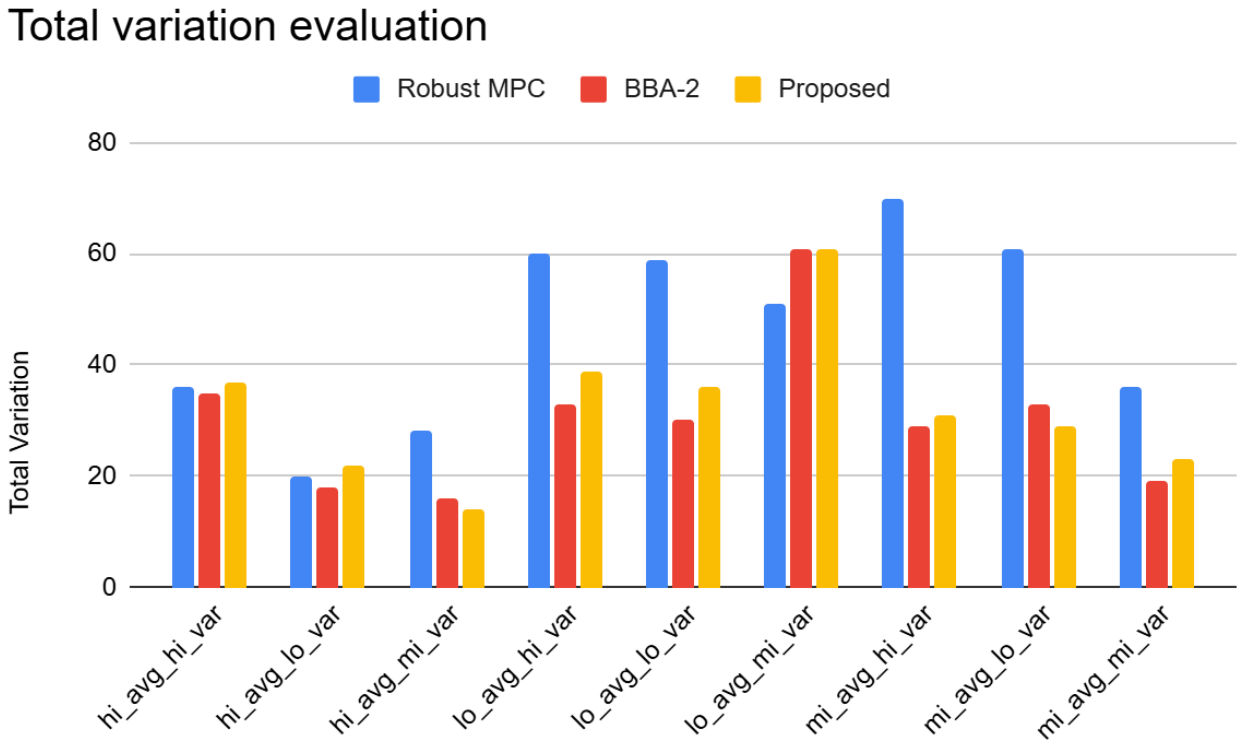
QoE	Robust MPC	BBA-2	Proposed
hi_avg_hi_var	3.21	2.96	2.99
hi_avg_lo_var	3.18	2.97	2.96
hi_avg_mi_var	**3.38**	**3.19**	**3.22**
lo_avg_hi_var	0.03	0.96	0.96
lo_avg_lo_var	-0.11	0.11	0.07
lo_avg_mi_var	0.05	0.10	0.11
mi_avg_hi_var	1.65	1.82	1.85
mi_avg_lo_var	1.73	1.68	1.74
mi_avg_mi_var	1.03	1.16	1.13

- From QoE graph, you can see that proposed methods have better QoE scores on one of the baseline if not both. The big improvement comes from low average cases where proposed methods maintain similar QoE to BBA-2 while it still has better performance on higher average cases.



Rebuffer time	Robust MPC	BBA-2	Proposed
hi_avg_hi_var	**1.76**	**0.33**	**0.33**
hi_avg_lo_var	2.00	0.47	0.47
hi_avg_mi_var	4.88	0.33	0.33
lo_avg_hi_var	42.29	1.12	1.12
lo_avg_lo_var	31.25	15.77	17.32
lo_avg_mi_var	21.76	2.27	2.95
mi_avg_hi_var	13.20	0.40	0.40
mi_avg_lo_var	9.37	0.50	0.50
mi_avg_mi_var	23.22	3.53	4.35

- From rebuffer time graph, proposed methods shows slightly higher rebuffer time than BBA-2 but it has significantly reduced time compared to Robust MPC. This gave a major advantage on low to mid average cases over Robust MPC.



Variation	Robust MPC	BBA-2	Proposed
hi_avg_hi_var	36	35	37
hi_avg_lo_var	**20**	18	22
hi_avg_mi_var	28	**16**	**14**
lo_avg_hi_var	60	33	39
lo_avg_lo_var	59	30	36
lo_avg_mi_var	51	61	61
mi_avg_hi_var	70	29	31
mi_avg_lo_var	61	33	29
mi_avg_mi_var	36	19	23

- From total variation graph you can see that proposed methods have lower variation than Robust MPC for most of the cases and sometimes lower than BBA-2.

The charts and graphs shows how versatile the proposed method is while Robust MPC and BBA-2 has clearly one-sided performances. Proposed methods can react to different scenarios while maintaining good performances which was the goal of the system.

Results Discussion [More slides if needed]

- What are your conclusions?
  - Does an algorithm (or variant) generally work better in all cases?

    Yes and no. The proposed model performs better overall, but in high average cases Robust MPC still outperforms and in lower average cases BBA-2 outperforms proposed model.
  - Do the algorithms have trade-offs, with one working better in certain environments than others?

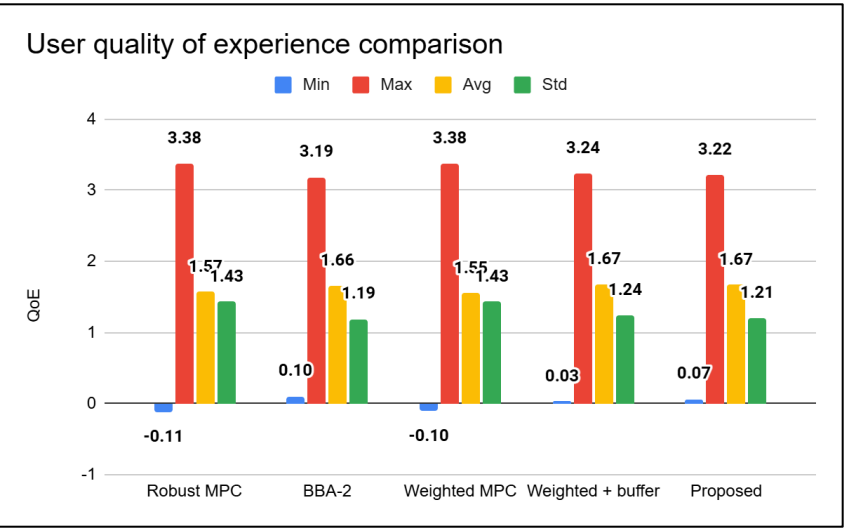
    Yes. Robust MPC worked better on high average network and BBA-2 worked better low average cases. However, as my goal was to create versatile algorithm, proposed model shows good performances on every conditions.
  - Do some algorithms favor one metric (e.g. quality) more than others?

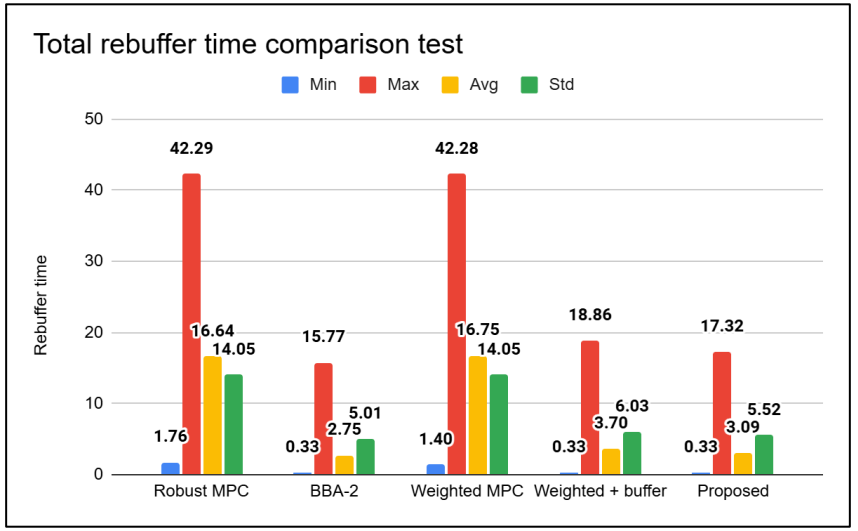
    Robust MPC seems to focuses on outputting highest quality possible and BBA-2 seems to consider more than just a quality as it considers buffer. Proposed methods added more consideration on many different way by adding different weight to chunks and penalize buffer prediction when it's needed.
  - In what settings does an algorithm perform the best or the worst, and why?

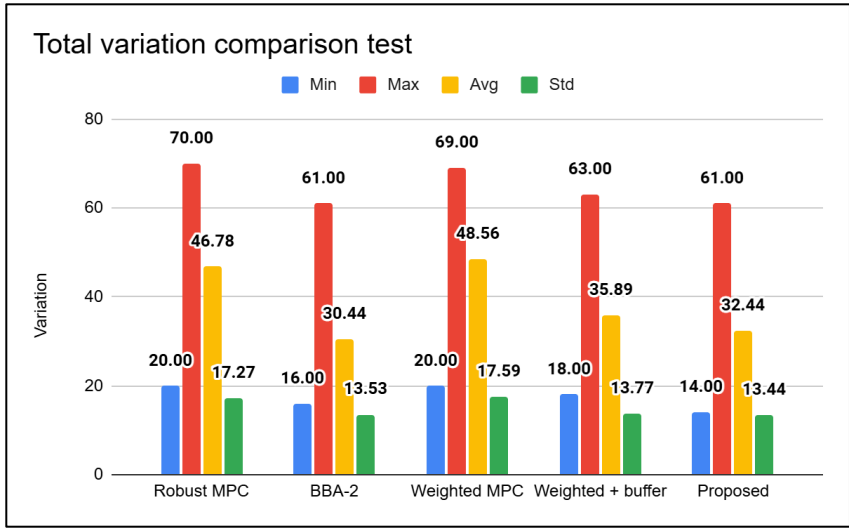
    Robust MPC: high average high variant. BBA-2: low average high variant. Proposed: mid average low variant.

    I believe this is because Robust MPC evaluates short-term future quality sequences optimizing for overall QoE, but it relies heavily on the accuracy of bandwidth prediction and does not account for longer-term trends or network variability. On the other hand, BBA-2 adapts based on current buffer occupancy without needing to estimate or predict bandwidth. This gives it stability and simplicity but limits its ability to respond to faster networks. Proposed methods shows good performance overall but compared to BBA-2 and robust MPC it shows the most difference on mid average cases. I believe this is because two baselines are having on sided strong point, proposed methods stands out more on mid average cases as both algorithms are not particularly strong on this section.

Other open-ended explorations if applicable







This data was generated by extracting minimum, maximum, average, and standard division from all the given test configures with same setting as previous result table.

- From the comparison test, you can see that Weighted MPC performed slightly better on worst case. However, it was not meaningful data as the difference is very small
- By adding buffer consideration on top of weighted MPC, the overall performance when up. It seems to have less peak performance on QoE, but it has meaning full performance gain on rebuffer time as well as number of variation. Also, minimum and average QoE score increased, indicating more reliable than Robust MPC.
- From proposed model, added buffer estimation confidence score penalty system, the rebuffer time and variation reduced but kept a similar QoE score as the previous version, even increasing performance on the worst case.
- Overall, the proposed method shows similar reliable performance as BBA-2 on lower average cases but has higher performance on high average cases which covers pros and cons of both Robust MPC and BBA-2.