PiTree: Practical Implementations of ABR Algorithms Using Decision Trees

Paper # P5C-04

Zili Meng  Jing Chen  Yaning Guo  Chen Sun  Hongxin Hu  Mingwei Xu
Adaptive Bitrate (ABR) Algorithms

Partially borrowed from “Neural Adaptive Content-aware Internet Video Delivery” in USENIX NSDI 2018.

https://transys.io/pitree
Adaptive Bitrate (ABR) Algorithms

- **Deep Learning**
- **Complex Optimization**
- **Explicit Formula**

- Harmonic Mean (Rate)
- Piecewise Linear (Buffer)
- Linear Programming
- Hidden Markov Model
- Neural Networks
- Deeper NNs

Timeline:
- 2011
- 2012
- 2013
- 2014
- 2015
- 2016
- 2017
- 2018
- 2019

https://transys.io/pitree
Drawbacks – Heavyweight

Client-side Implementation

• Large page size.
  – Pensieve \([\text{SIGCOMM’17}]\) increased HTML page size by 4x.
  – Page load time is increased by \(\sim 10s\).

• Long decision latency.
  – RobustMPC \([\text{SIGCOMM’15}]\) increased decision latency to seconds.
  – Decision latency > chunk length.

Server-side Implementation

• High operating expenses.
  – Up to millions of concurrent viewers.

\[ \begin{align*}
\text{R} & = \text{RobustMPC [SIGCOMM’15]: ILP} \\
\text{P} & = \text{Pensieve [SIGCOMM’17]: DNN} \\
\text{H} & = \text{HotDASH [ICNP’18]: 2xDNN}
\end{align*} \]

https://transys.io/pitree

PSC-04

4
Our Contribution: PiTree

• Design & train ABR algorithms offline as usual.
• Convert the model into a decision tree.
• Deploy the decision tree online.

Sophisticated ABR model (e.g., DNN, ILP)

Decision tree

Network traces

Online deployment

Decisions
Design Choice: Why Decision Tree?

• Non-parametric and expressive.
Design Choice: Why Decision Tree?

• Non-parametric and expressive.
• Lightweight for video players.

A decision tree with 100 leaf nodes:

- Page size increase <1%
- Decision latency <1ms

https://transys.io/pitree
Design Choice: Why Decision Tree?

- Non-parametric and expressive.
- Lightweight for video players.
- Following the decision logic of ABR algorithms.

Decision tree of BBA [SIGCOMM'14].

https://transys.io/pitree
Design Challenge: Sequential Dependency

• ABR Control is a sequential decision-making process.
Design Challenge: Sequential Dependency

- ABR Control is a sequential decision-making process.
- One wrong prediction may drive the student off teacher’s trajectory.

https://transys.io/pitree
PiTree: Imitation Learning – Follow the Leader

teacher.predict() -> correct actions

states

(s,a) set

(in)correct actions

virtual player

traffic traces

videos

add into

add into

retain

deploy onto video clients

https://transys.io/pitree
For design details and theoretical analysis, please refer to our paper.
Evaluation – Quality of Experience (QoE) Ratio

• **Summary of experiments**
  – 3 QoE metrics.
  – 3 sets of bandwidth traces.
  – 3 ABR algorithms.

• QoE ratio = \( \frac{QoE_{PiTree}}{QoE_{Original}} \)

• Average QoE ratio > 97%.
• Median QoE ratio > 98%.
• Details in the paper.

https://transys.io/pitree
Evaluation – Page Size

Our experiments: <0.1s @ 1Mbps

RobustMPC [SIGCOMM’15]: ILP
Pensieve [SIGCOMM’17]: DNN
HotDASH [ICNP’18]: 2xDNN

https://transys.io/pitree
Evaluation – Latency

Intel Core i7-8550

Qualcomm Snapdragon 821

RobustMPC [SIGCOMM’15]: ILP
Pensieve [SIGCOMM’17]: DNN
HotDASH [ICNP’18]: 2xDNN
Takeaways

- Current ABR algorithms are increasingly heavyweight.
- PiTree converts complex ABR algorithms to decision trees to deploy them in a lightweight way.
  - Use imitation learning to address the action dependency.
- PiTree can significantly reduce the algorithm overhead with negligible QoE loss.
  - Page size reduced by up to 5x, decision latency reduced by up to 1000x.
Thank you!

Questions and comments?

Try your ABR algorithms with PiTree!

https://transys.io/pitree

zilim@ieee.org