A Bayesian Approach to Decentralized Video Conferencing

Undergraduate Thesis

Presented in Partial Fulfillment of the Requirements for the Undergraduate Research Distinction at The Ohio State University

By

Sanjeev Melchizedek Gunawardena,

Department of Computer Science and Engineering

The Ohio State University

2023

Undergraduate Research
Distinction Examination
Committee:

Shaileshh Bojja Venkatakrishnan, Ph.D.,
Advisor

Andrew Perrault, Ph.D.

Donald Williamson, Ph.D.

Abstract

Recent years have seen a significant increase in the adoption of video conferencing as a preferred means of communication. The centralized nature of the leading video conferencing platforms has brought with it various challenges and vulnerabilities. Previous research indicates that decentralized solutions offer improvements to centralized systems. This research continues past work by implementing DecVi++, a simulation for a novel protocol for providing decentralized video conferencing based on the Thompson sampling approach to the combinatorial multi-armed bandit problem, which can be used to model video conferencing clients and servers in a decentralized network. Thompson sampling, a Bayesian approach, allows for the addition of prior information into a system to improve future predictions for selecting a decentralized path for a video stream. An event-driven simulator was developed to test this protocol and compare it with the current leading video conferencing platform and a previous decentralized protocol, DecVi, which used a frequentist approach to the combinatorial multi-armed bandit problem. The simulator modeled the end devices of a video-conferencing network as nodes in a bipartite graph. Edges between nodes indicated the path of a video stream and weights on the edges were included as estimates of the bandwidth and latency of a given path. Using Thompson sampling, these values were sampled from normal distributions as inputs for the protocol to determine the optimal paths from a source client to recipient clients. Client feedback was then used to update the hyperparameters of these distributions. Experiments using the simulator showed that the incorporation of prior knowledge into the system produced more optimal video streaming paths. The DecVi++ algorithm tended to yield higher initial performance rewards than the DecVi algorithm but was sensitive to various hyperparameter initializations and experiment setups. The results of this research suggest that DecVi++ offers an encouraging approach to decentralized video conferencing.

This thesis is dedicated to my family, who support me in every endeavor.

Acknowledgments

I would like to thank my advisor Dr. Shaileshh Bojja Venkatakrishan, who has been an incredible mentor throughout this process. I thank Dr. Bojja Venkatakrishnan for all of his advice, constantly being available to help, always being calm and patient, and for answering my never-ending stream of questions.

I would also like to thank Dr. Andrew Perrault and Dr. Donald Williamson for being a part of my oral defense committee. I thank all three professors for their continuous support and for being high-quality instructors during my time in class with them.

Finally, I would like to thank my friends and family for always being supportive during the difficult moments of my life and for their constant encouragement which has driven me to get to this point and will continue to drive me as I continue learning and growing.

Vita

2019	Centerville High School
2023 (Anticipated)	B.S. Computer Science Engineering, The Ohio State University

Fields of Study

Major Field: Department of Computer Science and Engineering

Table of Contents

List of Figures

Figure **Page**

Chapter 1: Introduction

Largely due to the global COVID-19 pandemic, video conferencing has become a standard form of communicating amongst people from various countries, demographics, and occupations. The global video conferencing market has been valued at over 6 billion dollars and is projected to rise in the coming years [2]. However, significant issues have arisen due to the centralized nature of the current leading platforms, such as Zoom.

Centralized platforms maintain user data in private servers. When multiple parties communicate over a video conferencing platform, sender bits are first sent to a platform-owned private server with a public IP address before being transmitted to receivers from the private server. In some cases, the physical distances between senders and receivers are significantly smaller than the distances between senders and receivers and private servers. For example, Zoom has no data centers in Africa [3], which means that video conferencing sessions with all parties in Africa must have all bits routed to another continent before they are received.

As a result of the increased adoption of blockchain technology and cryptocurrencies, there has been a significant increase in the development of Web3 and the decentralized internet [6]. Decentralized video conferencing platforms, such as Huddle, are being developed as part of this wave [1].

Decentralized platforms offer the potential for significant improvements over their centralized counterparts. Decentralized platforms utilize peer-to-peer networks rather than centralized servers. In a decentralized video conferencing platform, any computer on the internet can volunteer to be an intermediary server between parties of a video conference in exchange for cryptocurrency tokens.

In a decentralized network, parties could connect to one or more intermediary servers that would be much closer to all parties when compared to a centralized server. This closer distance could improve the quality of experience because of lag times caused by packet delay from long distances. Furthermore, decentralized networks would not suffer from lag created due to server overloads. Decentralized networks would also be less susceptible to power outages, as power outages in centralized networks affect a large majority of users. Finally, decentralized networks offer improvements in pricing, privacy, and censorship.

While a decentralized video conferencing platform has the potential to be a marked improvement over a centralized platform, there are still problems that need to be addressed for such a platform to exist.

As there are no centralized servers in a decentralized network, there is no centralized information regarding the location of servers. Furthermore, individual volunteer servers have limited bandwidth capabilities, and different clients in a video conferencing session have different bandwidth requirements. Hence, a decentralized video conferencing network necessitates a protocol for determining efficient paths for clients to send streams to each other that balances client requirements, efficiently utilizes servers on the network, and keeps latency low.

For a client to send its stream to other clients in the network, a multicast tree can be generated with recipient clients and intermediary servers as nodes in the tree. Different trees can be constructed that reflect different possible paths by which the intermediary servers send the source client's stream to the recipient clients.

This thesis proposes the DecVi++ algorithm, which aims to find the optimal tree for each client to send its stream to the other clients in the network. Since each client constructs its tree independently of the others, the algorithm in this thesis aims to determine the optimal tree for a single source client and its associated network of SFUs and recipient clients.

Previous work from Dr. Bojja Venkatakrishnan's research group has proposed the DecVi protocol [7], a frequentist approach to constructing the optimal tree for a client. The protocol yielded promising results when compared with centralized video conferencing.

Chapter 2: Background

While often more compute-intensive and complex to implement, Thompson sampling is an alternative Bayesian approach used by DecVi++. While Thompson Sampling and comparable frequentist methods have the same theoretical guarantees, Thompson sampling often outperforms frequentist methods in practice. In Bayesian statistics, observed data rather than unobservable quantities contain all information necessary to make relevant inferences [5]. Thompson Sampling has the added flexibility of improving a system given additional knowledge about the system.

In current p2p video conferencing systems, rather than requesting a particular bit rate, clients request a number of scalable video coding "layers". In scalable video coding, a media server encodes a video stream into multiple bit stream layers with a base layer and enhancement layers – each providing a greater stream quality. Clients can decode the video stream with the base layer and any number of enhancement layers.

Instead of having to transcode into different bit rates for each client's particular requirement, media servers merely have to encode the base and enhancement layers and then forward the requested number of layers to each client.

Chapter 3: Methodology

3.1 Network Model

The simulator in this thesis follows the same model as the DecVi algorithm [7]. We consider a single source client c_0 in a video conferencing session, with a set of recipient clients C. c_0 encodes $Q \in N$ scalable video coding layers and recipient clients c_0 have a requirement for receiving at least $q^*(c) > 0$, $q^*(c) \leq Q$, $q^*(c) \in N$ layers with as low a packet latency as possible.

The latency of a path in the multicast tree from the source client to a recipient client c_0, s_0, $s1$, ... s_k, c is the overall delay of a packet on the path.

All clients are part of a network with a set of servers s known as scalable forwarding units (SFUs). In this model, each scalable video coding layer consumes one unit of bandwidth. An SFU $s \in S$ may forward at most $q^*(s) \leq q$, $q^*(s) \leq b_s$ layers, where b_s is the bandwidth capacity of the SFU and q is the number of layers received by the SFU.

DecVi++ aims to find an efficient multicast tree for the source client to send its video stream to the other clients via the SFUs in the network. A client aims to maximize the net bandwidth ratio $q(c)/q^*(c)$ of layers received $q(c)$ to layers requested $q^*(c)$ and minimize the net packet latency $d(s, c)$ it experiences. A control variable

α is used to reflect the priority tradeoff between latency and bandwidth for a given client. Hence, the reward for a given client and SFU is computed as

$$r_s(c) = -d(s,c) + \alpha q(c)/q^*(c). \tag{3.1}$$

An SFU in the tree aims to maximize the aggregate reward $\sum_{c \in C_s} r_s(c)$ for all clients it forwards the stream to. An optimal tree, therefore, maximizes the net reward for all SFUs in the tree.

To construct the multicast tree, the source client c_0 first selects an SFU close to it to disseminate c_0's stream to all the other clients in the video conferencing session. c_0 sends this "gateway" SFU its identifier, the number of scalable video coding layers sent, the list of recipient clients and IP addresses, and the number of layers requested by each recipient client in addition to its stream of video packets.

The gateway SFU is then responsible for forwarding the stream to the other clients, either directly or by selecting other SFUs in the network to forward the stream to as downstream nodes in the tree.

Subsequent SFUs, therefore, receive a subset of the client metadata for the clients they are responsible for. Hence, all SFUs in the tree, including the gateway SFU, receive a list of clients they are responsible for forwarding the stream to. For each client, an SFU decides whether to forward the stream directly to the client or to forward the stream and client metadata to a downstream SFU.

Each SFU must therefore select mappings between a set of SFUs, including itself and downstream SFUs, and the set of clients it is responsible for. For an SFU s, these selected mappings are known as the s's action, and the set of all possible mappings between clients and SFUs can be modeled as a bipartite graph $G(S, C)$ at s. A possible mapping between an SFU and a client is represented by an edge in the graph.

Each edge $s', c' \in G$ has two weights associated with it. The first edge weight, $\hat{d}(s, s', c')$ is an estimate of the total latency between s and c' if the stream is forwarded through s' and s' is responsible for c'. The second edge weight, $\hat{\phi}(s', c')$ is an estimate of the average number of layers c' receives from s' per unit layer received by s' from s, if s forwards the stream to s' and s' is responsible for c'.

Upon receiving a packet from a video stream, recipient clients send feedback back up the multicast tree which is then used to update the edge weights of the bipartite graphs at the SFUs.

3.2 Algorithm

For each SFU, finding the optimal set of mappings can be modeled as a combinatorial multi-armed bandit problem (CMAB). Rather than only selecting one arm as in a classical multi-armed bandit problem, each SFU selects multiple arms for a given round. Each of the k*n possible mappings in the bi-partite graph between k clients and n SFUs is considered an arm and the subset of k mappings between clients and the SFUs responsible for those clients, the SFU's action, is considered a "superarm".

To construct the optimal multicast tree given the current knowledge of the network, the CMAB problem is modified to consider two variables – the bipartite graph edge weights $\hat{d}(s, s', c')$ and $\hat{\phi}(s', c')$ reflecting the current bandwidth ratio and latency estimates of the arm, which are based on previous client feedback.

The Thompson Sampling algorithm proposed by Agrawal and Goyal for Bernoulli bandits [4] is adapted for the video streaming problem. This algorithm maintains a

Beta distribution for each possible arm and samples these distributions to decide the best arm to play. After playing this arm, the hyperparameters of the arms associated Beta distribution are updated to reflect the observed Bernoulli rewards.

This algorithm is adapted for the video streaming problem in two ways. First, the likelihood function is modified from a Bernoulli distribution to a Gaussian distribution to reflect continuous rewards. The associated conjugate prior is therefore modified to another Gaussian distribution with mean μ_0 and precision τ. Secondly, the estimated reward is no longer based on sampling from one distribution. Instead, two independent Gaussian distributions are sampled to reflect the estimated bandwidth ratio, $\hat{\phi}(s', c')$, and estimated latency, $\hat{d}(s, s', c')$, experienced by the client c':

$$\hat{\phi}(s', c'), \hat{d}(s, s', c') \sim N(\mu_0, \tau_0^{-1}) \tag{3.2}$$

Once reward estimates are computed for each possible mapping, the optimal action (superarm of mappings) is computed and played. The optimal action over all possible superarms for a given SFU is computed using the integer-quadratic-program used in DecVi, which maximizes the estimated reward for a superarm at an SFU [7].

In the DecVi algorithm, the weights are directly updated using the feedback from the clients with a learning rate. The algorithm also uses a parameter to randomly decide the probability of taking the estimated best action or a random action to encourage exploration. Using the Thompson sampling approach, the estimated best action is always played because exploration is already built into the model by sampling from distributions to compute the best action, rather than only using previous feedback.

Finally, after playing the estimated best action, the feedback received from the client is then used to update the hyperparameters of the associated Gaussian distributions for $\hat{\phi}(s', c')$ and $\hat{d}(s, s', c')$ using the posterior update rules for Gaussian distributions:

$$\tau_0 \leftarrow \tau_0 + n\tau \tag{3.3}$$

$$\mu_0 \leftarrow \frac{\tau_0 \mu_0 + \tau \sum_{i=1}^{n} x_i}{\tau_0 + n\tau} \tag{3.4}$$

Because these updates are performed every time the algorithm is invoked, n is set to 1. x_i represents the feedback received from the client for latency or bandwidth ratio for the respective distribution. τ reflects the step size by which the precisions of the distributions are narrowed given more client feedback.

3.3 Evaluation

DecVi++ was evaluated using a modified version of the event-driven simulator written in Python to evaluate DecVi. Packets sent or received by nodes in the network were considered events, and evaluating events in the simulators could cause other events to be triggered.

Events were evaluated by their order in a queue for a timestamp, which was based on simulated link delays. For a given simulation, the source client c_0 first sends a "*trigger_action*" packet to its gateway SFU along with the metadata for its list of recipient clients. Upon deciding its superarm of mappings given the other SFUs in the network, the gateway SFU forwards the *trigger_action* packet to its downstream nodes. The downstream SFUs then repeat this process, passing down the *trigger_action* packet and determining their superarms until the packet has been received by all the clients in the network. Upon receiving a packet, the clients send feedback back up the tree, which is used to update the edge weights of the bipartite graph at each SFU. This process of a packet sent down the tree and feedback received up the tree is known as one round in the simulation. Simulations during the evaluation were initialized to repeat this process for a given number of rounds.

3.3.1 Hyperparameter Tuning

The hyperparameters of both DecVi and DecVi++ were tuned before comparing the two algorithms to find the optimal hyperparameters for each algorithm. For the different hyperparameter settings, the algorithms were run and rewards were collected in increments of 10 rounds from 10 to 300. 3 trials of each algorithm were run and the rewards were averaged over the trials. These hyperparameter tests were run using a small network with 4 SFUs and 5 clients including the source client.

In particular, the precision increase step size τ was modified to determine the best step size for narrowing the Gaussian distributions of the edge weights given more client feedback over time.

The precision step size was varied between the following values: $0.25, 0.75, 1, 2, 5$. A plot of the reward for each step size is shown in Figure 3.1.

Higher values of τ led to more variation in the reward per round. A value of $\tau = 1$ was selected as the optimal value for further experiments because, for this value of τ, there was minimal variation and a continuous increase in reward over the rounds.

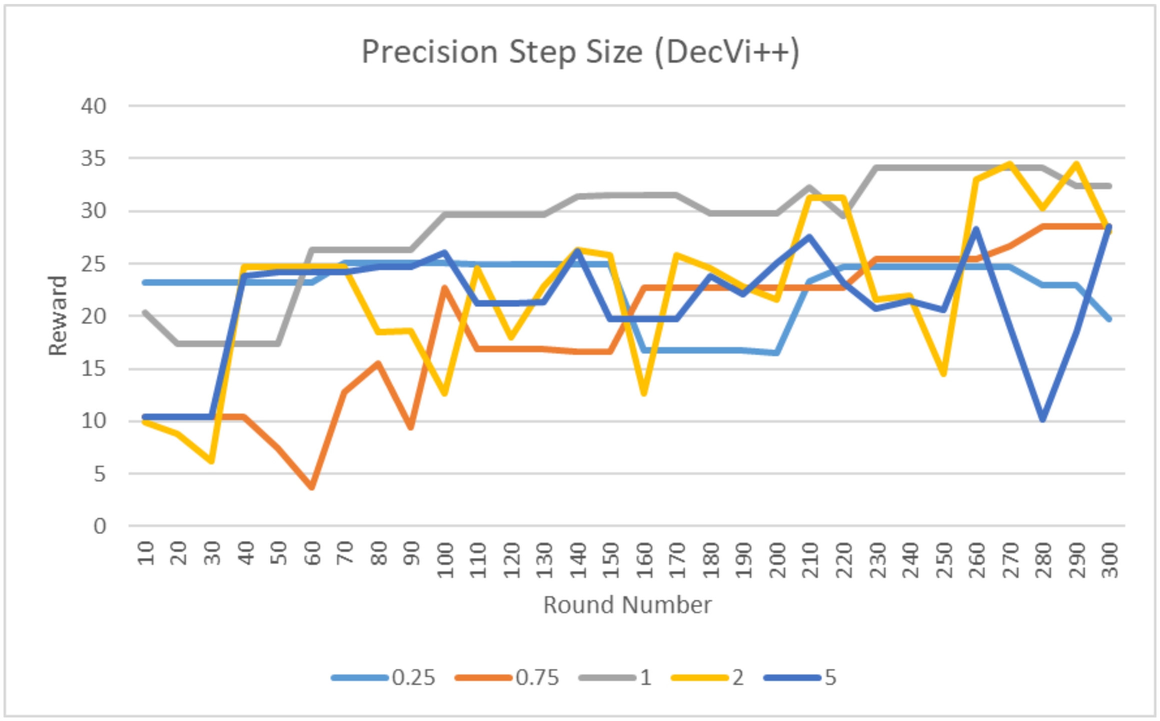

Figure 3.1: Comparison of Different Values of τ for DecVi++ for a small network.

3.3.2 Experiment Setup

Trials of the DecVi algorithm showed that for smaller numbers of clients and servers, the algorithm converged in approximately 200 rounds. The DecVi++ algorithm was compared to the DecVi algorithm in the same configuration used to compare the algorithm hyperparameters. The algorithms were run and rewards were collected in increments of 10 rounds from 10 to 300. 3 trials of each algorithm were run and the rewards were averaged over the trials.

The algorithms were compared with different levels of prior knowledge incorporated into the system. First, no prior knowledge was incorporated into the system.

The algorithms were then run with half of the edge weights for latency and bandwidth initialized to their true values plus Gaussian noise. Finally, the algorithms were compared with all of the edge weights for latency and bandwidth initialized to their true values plus Gaussian noise.

For both algorithms, the final multicast trees generated after 300 rounds were compared to a global optimum baseline tree generated using the Gurobi Optimization Toolbox that DecVi was compared to [7]. Cumulative distribution functions for the bandwidth ratios and latencies experienced by the SFUs in the networks were also compared to this global optimum for both algorithms.

DecVi and DecVi++ were compared in two network settings. First, the algorithms were compared in a small network setting with 4 SFUs and 5 clients including the source client. The algorithms were then compared in a medium network setting with 7 SFUs and 11 clients including the source client.

Chapter 4: Results

For the small network comparison with 4 SFUs and 5 clients, varying results were seen among the varying comparisons, as shown in Figure 4.1. Like in other experiments, DecVi++ tended to outperform DecVi early on in the number of rounds, whereas DecVi surpassed DecVi++ given a longer simulation duration. When both fully initialized with noisy edge weights close to true values, DecVi++ achieved the optimal topology, shown in Figure 4.2, at every round number, whereas DecVi had more variance and only achieved the optimal topology at certain rounds.

For the medium network comparison with 7 SFUs and 11 clients, both DecVi and DecVi++ performed poorly overall, yielding negative rewards as seen in Figure 4.3. DecVi++ with no initializations initially struggled against DecVi with no initializations but yielded a higher final reward after 300 rounds. When both were initialized with half of the true edge weights plus Gaussian noise, DecVi outperformed DecVi++ at almost every round in the medium network setting.

Between the trials, DecVi++ yielded a worst-case reward of approximately -103 and a best-case reward of approximately -45 as seen in Figures 4.4 and 4.5.

Finally, the cumulative distribution functions for the bandwidth ratios and latencies experienced by the SFUs in the medium network setting were comparable to the global optimum, as shown in Figures 4.6 and 4.7.

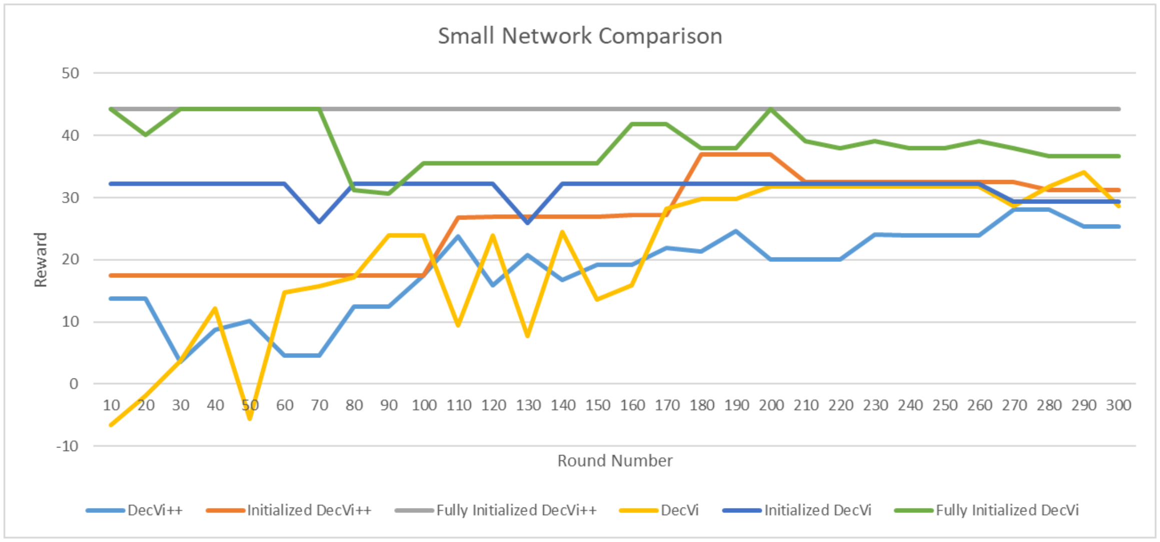

Figure 4.1: Comparison of DecVi and DecVi++ for a small network under different initialization conditions.

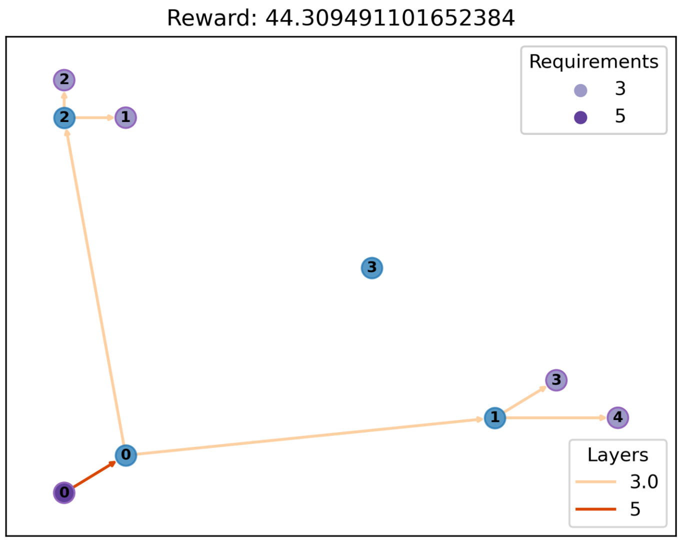

Figure 4.2: Multicast Tree generated by DecVi++ with fully initialized noisy edge weights.

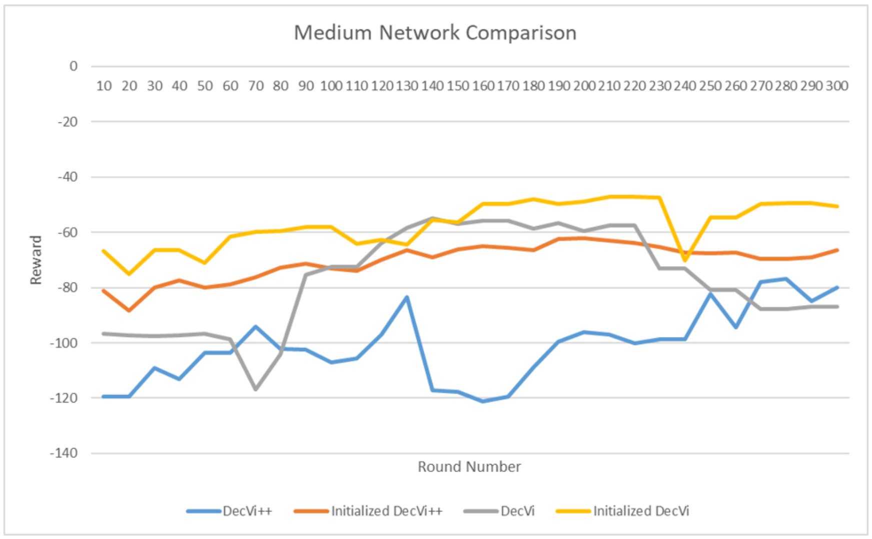

Figure 4.3: Comparison of DecVi and DecVi++ for a medium network under different initialization conditions.

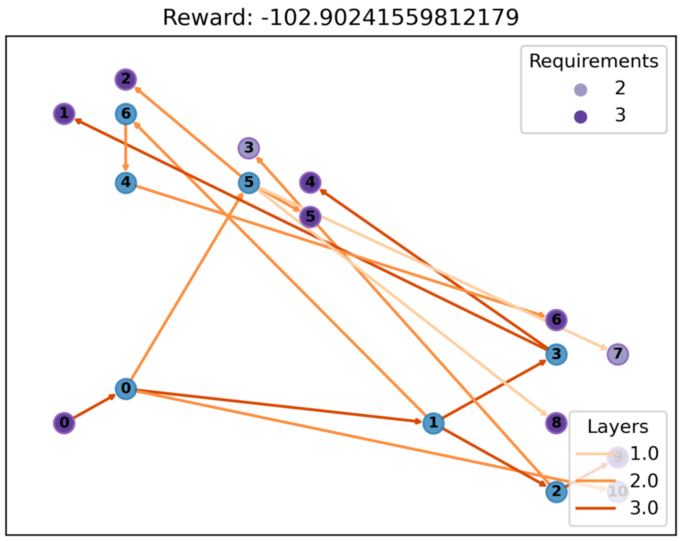

Figure 4.4: Worst-Case Multicast Tree generated by DecVi++ in a Medium Network Setting.

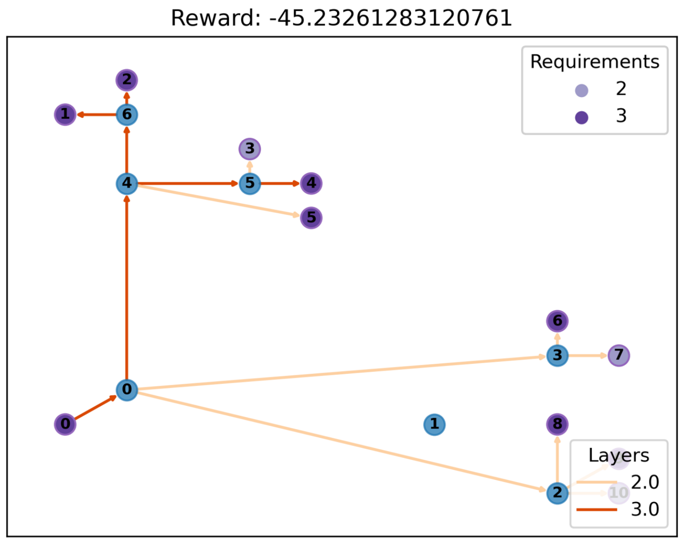

Figure 4.5: Best-Case Multicast Tree generated by DecVi++ in a Medium Network Setting.

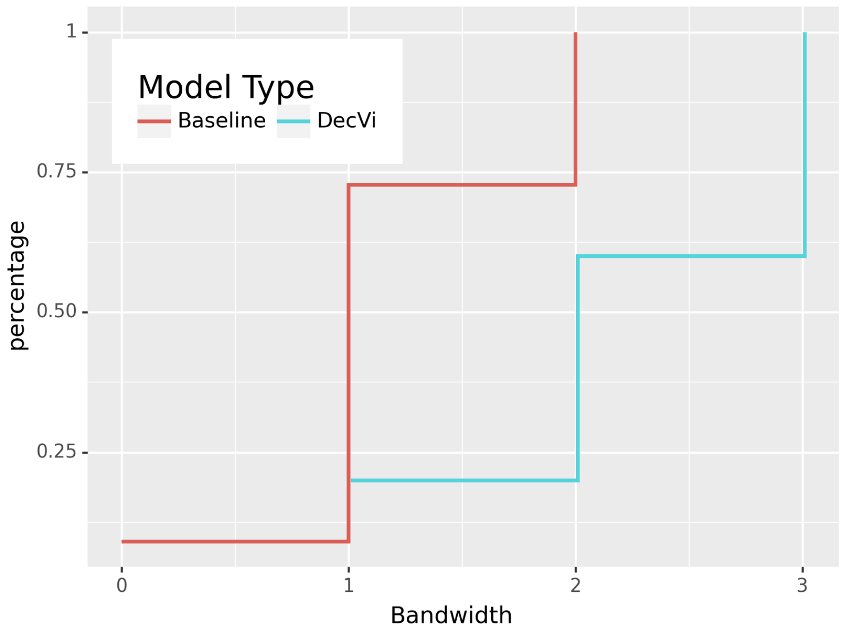

Figure 4.6: CDF of Bandwidth for Medium Network.

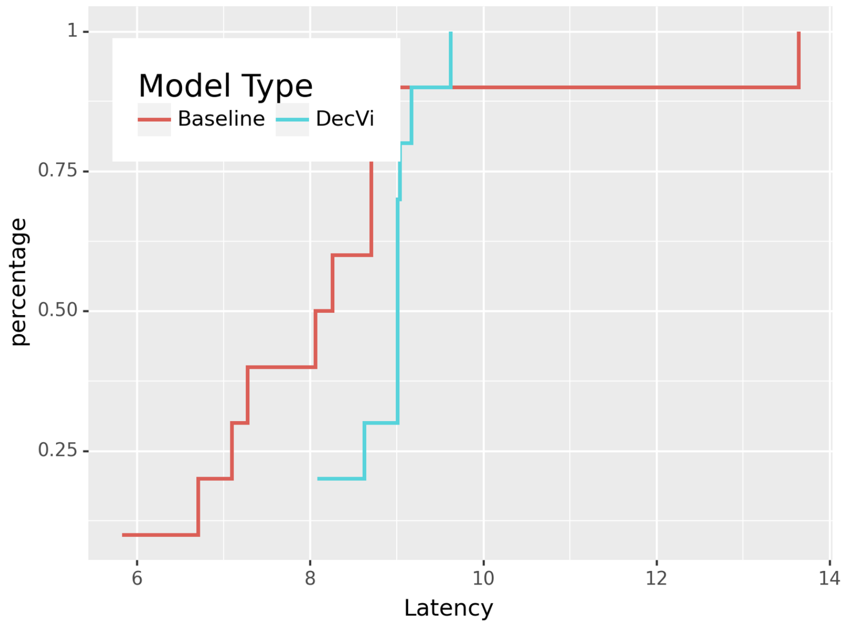

Figure 4.7: CDF of Latency for Medium Network.

Chapter 5: Conclusions

The DecVi++ algorithm tended to yield higher initial performance rewards than the DecVi algorithm but was sensitive to various hyperparameter initializations and experiment setups. Overall, the results of this research suggest that DecVi++ offers an encouraging approach to decentralized video conferencing.

An analysis of the data and edge weight updates suggests that when DecVi++ outperformed DecVi, it did so because the exploration aspect of DecVi++ was better informed than that of DecVi. In DecVi++, even if the superarm with the highest estimated reward is not selected based on sampling the distributions for its arm's estimated bandwidth ratios and latencies, it is likely that another superarm with a high estimated reward is selected. In the DecVi algorithm, however, the exploration aspect is completely uninformed – a random superarm is selected if the random value parameter is true.

Furthermore, in the cases of no prior knowledge, DecVi++ likely tended to outperform DecVi initially because a sample of the edge weights from a Gaussian distribution is a better initial estimate of the weights than a static value. Sampling edge weights for different mappings reflect the true diversity of rewards from different mappings, whereas static initializations are almost guaranteed to be a poor representation of true rewards.

In the future, more detailed comparisons of the algorithm are suggested to gain a more robust mathematical explanation of the performance differences between different approaches to the combinatorial multi-armed bandit problem for the application of video conferencing. A small and simple network is recommended to compare the per-round updates of the edge weight estimates for the two approaches.

Moreover, further hyperparameter tuning and adjustments to the algorithm are recommended for continued analysis. Different distributions and conjugate priors can be used to sample the edge weights to understand which distributions best model video conferencing in the real world.

Finally, more configurations should be tested to fully grasp the difference between the DecVi and DecVi++ algorithms. The DecVi algorithm was previously compared to a global optimal baseline using random node locations and real-world locations. These comparisons are a good starting point for future comparisons with DecVi++.

Bibliography

[1] "Huddle". *Huddle01.com*.

[2] "Video Conferencing Market Size, Share and Covid-19 Impact Analysis". *Fortune Business Insights*.

[3] "Zoom Data Center Locations". *DataCenterLocations.com*.

[4] Shipra Agrawal and Navin Goyal. "Analysis of Thompson Sampling for the Multi-armed Bandit Problem". *Journal of Machine Learning Research*, 23, 2012.

[5] Branimir K. Hackenberger. "Bayes or not Bayes, is this the question?". *Croatian Medical Journal*, 2019.

[6] Aaron Mak. "What Is Web3 and Why Are All the Crypto People Suddenly Talking About It?". *Slate*, November 2021.

[7] Jingren Wei and Shaileshh Bojja Venkatakrishnan. "DecVi: Adaptive Video Conferencing on Open Peer-to-Peer Networks". 2022.