Supplementary Material: Meta-Reinforcement Learning via Buffering Graph Signatures for Live Video Streaming Events

Stefanos Antaris
KTH Royal Institute of Technology
Hive Streaming AB
Sweden
antaris@kth.se

Dimitrios Rafailidis
University of Thessaly
Greece
draf@uth.gr

Sarunas Girdzijauskas
KTH Royal Institute of Technology
Sweden
sarunasg@kth.se

I. Live Video Streaming Analysis

TABLE I
Summary statistics of the three datasets.

Datasets	LiveStream-1	LiveStream-2	LiveStream-3
#Events	30	30	30
#Offices	62	12	56
#Enterprises	1	1	15
#Viewers (K)	24.752	28.879	148.972
#Connections (M)	1.807	0.590	4.139
Avg. #Events Per Viewer	2.722 ± 2.061	1.349 ± 0.883	1.141 ± 0.348

Our study focuses on how the tracker/agent can learn the viewers' connection selection policy based on past live video streaming events and quickly generalize to new ones. We collected three real-world datasets, LiveStream-1, LiveStream-2, and LiveStream-3 which were anonymized and made publicly available. In Table I, we summarize the statistics of the three datasets. Each dataset consists of 30 temporal interaction networks, corresponding to 30 different live video streaming events in Fortune-500 companies. To generate each temporal interaction network, we monitored the viewers' interactions via the tracker during the live video streaming events, as well as the viewers' communication. We considered the tracker's reported interactions during the first 45 minutes of each event, as afterwards viewers started to abandon the event. Both LiveStream-1 and LiveStream-2 contain live video streaming events occurred at a single enterprise in different time periods. On inspection of Table I we can observe that the viewers in LiveStream-1 are distributed to more offices than in LiveStream-2. This happens because the selected enterprise in the LiveStream-1 dataset adapted its network between different live video streaming events. LiveStream-3 consists of 30 temporal interaction networks, that correspond to two live video streaming events occurred at 15 different enterprises. Provided that viewers cannot attend events of different enterprises, the viewers are distributed to 56 offices and the majority of the viewers participate in a single event.

In Figure 1, we present the average number of viewers in the collected events. The live video streaming events in LiveStream-3 attracted more viewers than LiveStream-1 and LiveStream-2. Provided the limited number of edges/connections per viewer (Table I), this means that the viewers' interaction data in the LiveStream-3 events are more sparse than the other datasets. Moreover, we observe that in the LiveStream-3 events, more than $1.5K$ viewers emerge at the first $0-10$ minutes, whereas in LiveStream-1 and LiveStream-2, $0.8K$ and $0.4K$ viewers emerged, respectively. This indicates that the temporal interaction graphs of the events in LiveStream-3 change significantly, when compared with the events of the other two evaluation datasets.

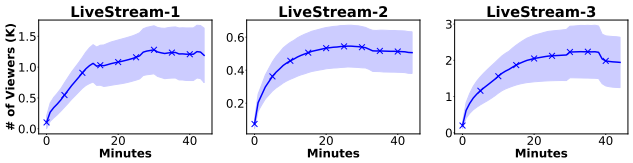

Fig. 1. Average #viewers during the live video events.

The tracker adapts the viewers' connections to distribute the video content through connections with high network capacity. Given that viewers emerge and leave at unexpected pace, the weight/throughput distribution of the viewers interactions, changes every minute $t = 1, \ldots, T$ of a live video streaming event. To measure the difference between the weight/throughput distributions $\mathcal{B}^t$ and $\mathcal{B}^{t-1}$ of two consecutive minutes, we adopt the Kullback-Leibler (KL) divergence [1], as follows:

$$D_{KL}(\mathcal{B}^t||\mathcal{B}^{t-1}) = \sum_{x \in \mathcal{X}} \mathcal{B}^t(x) log\left(\frac{\mathcal{B}^t(x)}{\mathcal{B}^{t-1}(x)}\right)$$

where $\mathcal{X}$ is the equally partitioned probability space of the interactions weight/throughput. A high KL-divergence value corresponds to significant changes on the weight/throughput distributions between consecutive live video streaming minutes, whereas a low value indicates that viewers converge to their optimal connections.

As illustrated in Figure 2, the weight/throughput distribution in LiveStream-1 has a higher KL-divergence value than

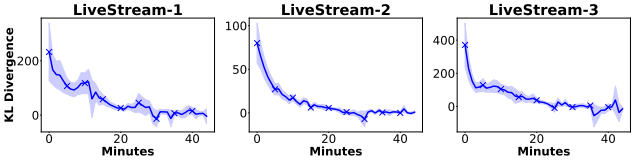

Fig. 2. Average distribution of KL-divergence of the connections' weights during the live video streaming events.

LiveStream-2. This means that the tracker in LiveStream-1 adapts the viewers' connections more frequently than in the LiveStream-2 dataset. In LiveStream-2, the connections converge in less minutes than the other two evaluation datasets. This occurs because the viewers in the LiveStream-2 events are distributed to the lowest number of offices, thus, the viewers require less changes in their connections than the viewers in LiveStream-1 and LiveStream-3. Moreover, we observe that the weight/throughput distribution in the LiveStream-1 dataset vary among different events, reflected by the high standard deviation of the KL-divergence value over the streaming minutes. This indicates that each event in LiveStream-1 occurred at different offices of the enterprise network. Finally, the network's capacity distribution in LiveStream-3 significantly changes at the first minutes of the events, as the KL-divergence values drop. As we demonstrated in Section IV, the performances of the proposed meta-learning strategy and the examined baseline approaches not only depend on the different patterns that viewers emerge, but also on the differences of the networks' capacities between the events.

II. Melanie Algorithm

Algorithm 1 MELANIE

Input: $p(\mathcal{T})$
Output: π_θ, f_w
Random Initialisation : θ, w
1: **for** $\mathcal{T}_i \in p(\mathcal{T})$ **do**
2: $\mathcal{E}_i = \mathcal{E}_i^s \cup \mathcal{E}_i^q$
3: **for** $e_{u,v}^t \in \mathcal{E}_i^s$ **do**
4: $a^t = \pi_\theta(s^t)$ // Equation 4
5: $Q_w(s^t, a^t) = f_w(s^t, a^t)$ // Equation 5
6: *Store transition* (s^t, a^t, s^{t+1}) *in the replay memory buffer* $\mathcal{D}$
7: *Retrieve K transitions from* $\mathcal{D}$ *based on the highest KL-divergence values*
8: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{actor}(\pi_\theta; \mathcal{E}_i^s)$ // Equation 6
9: $w \leftarrow w - \eta \nabla_w \mathcal{L}^{critic}(f_w; \mathcal{E}_i^s)$ // Equation 7
10: **end for**
11: *Compute the graph signature* $\mathbf{H}_i$ *of task* $\mathcal{T}_i$ // Equation 3
12: *Store* $\mathbf{H}_i$ *in the graph signature buffer* $\mathcal{J}$
13: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\theta^{meta}(\pi_\theta; \mathcal{E}_i^q)$
14: $w \leftarrow w - \eta \nabla_w \mathcal{L}_w^{meta}(f_w; \mathcal{E}_i^q)$ // Equation 8
15: **end for**
16: **return** π_θ, f_w

In Algorithm 1, we present the steps of the MELANIE model to learn a global policy π_θ of the agent/tracker. The input of MELANIE is a distribution of tasks $\mathcal{T}$, where each task corresponds to a streaming event, and the outputs are the global policy π_θ of the Actor network and the global parameters f_w of the Critic network. We randomly initialize the parameters θ and w. In lines 3-10, we train our model to adapt to a new task $\mathcal{T}_i \sim \mathcal{T}$, given a set of support interactions $\mathcal{E}_i^s$. In line 4, we compute the action a^t based on the policy π_θ, followed by the Actor network and the state s^t of the user u in the interaction $e_{u,v}^t$. In line 5, the Critic network evaluates the selected action a^t of the Actor network. In Lines 6-9, the state-action transition is stored in the replay memory buffer $\mathcal{D}$ and the K most dissimilar state-actions are retrieved to update the parameters θ and w based on Equations 6 and 7, respectively. At the end of the task adaptation component, in Lines 11-12 we compute the graph signature $\mathbf{H}_i$ of the task $\mathcal{T}_i$ and store it in the graph signature buffer $\mathcal{J}$. In lines 13-14, we update the global parameters θ and w based on the query set $\mathcal{E}_i^q$ and Equation 9. We repeat the process in lines 1-15 until we have experienced all the input tasks in $\mathcal{T}$.

III. Experimental Environment

All experiments were run on a single server with a CPU Intel Xeon Bronze 3106, 1.70GHz and a GPU Geforce RTX 2080 Ti. The operating system of the server was Ubuntu 18.04.5 LTS. We implemented MELANIE using PyTorch 1.7.1 and generated the reinforcement learning environment with OpenAI Gym 0.17.3. During the agent's training, we initialized the reinforcement learning environment with the viewers' interactions $\mathcal{E}^s$ of the support set. At each time step, the agent/tracker took an action to establish a connection among viewers $u \in \mathcal{V}$ and $v \in \mathcal{V}$, where $e_{u,v} \in \mathcal{E}^s$. Therefore, the agent exploited only the interactions that appeared in the support set $\mathcal{E}^s$, rather than the whole set of connections that the viewer established during the event. At each time step t, the agent took an action a^t based on a policy π_θ for a viewer interaction $e_{u,v} \in \mathcal{E}^s$. Afterwards, the agent observed the reward $b_{u,v} = R(s, a)$, where $b_{u,v}$ was the throughput measurement of viewer u in the interaction $e_{u,v}$. Then, the viewer's state s^t was updated to s^{t+1} and the interaction $e_{u,v}$ was removed from the support set $\mathcal{E}^s$. To evaluate the learned policy π_θ, we initialized the test environment with the interactions that appeared in the query set $\mathcal{E}^q$. Similar to the training process, given an interaction $e_{u,v} \in \mathcal{E}^q$ the agent took an action a^t to connect viewers $u \in \mathcal{V}$ and $v \in \mathcal{V}$. Provided that different live video streaming events had different throughput distributions and the throughput corresponded to the received rewards, in our implementation we normalized the reward based on the maximum throughput.

IV. Parameter Settings

For each examined model, we tuned the hyperparameters based on a grid selection strategy. In Jodie we set the node embedding size to 128 in all datasets. In EvolveGCN and DySAT, the node embedding size was fixed to 256 and we learned the node embedding based on 2 consecutive graph snapshots in all datasets. In DySAT we employed 4 attention heads in the LiveStream-1 dataset, and 6 attention heads in LiveStream-2 and LiveStream-3. VStreamDRLS used 64-dimensional node embeddings and learned node representations based on 3 consecutive graph snapshots in all datasets. In PolicyGNN, we used 128-dimensional node embeddings.

The discount factor γ was set to 0.95 and in the ϵ-greedy exploration-exploitation strategy ϵ was fixed to 0.2. MetaHIN used 32-dimensional node embeddings LiveStream-1, while we set the node embeddings to 256 for LiveStream-2 and LiveStream-3. In Meta-Graph, we generated 128-dimensional node embeddings for the LiveStream-1 and LiveStream-2 datasets and 32-dimensional node embeddings for LiveStream-3. In MELANIE and its variants, we generated 64-dimensional representations in all datasets. The replay memory buffer size was set to 128, ϵ was fixed to 0.2 and the discount factor γ to 0.65. We optimized the examined models via the Adam optimizer with a learning rate 0.01 and weight decay rate 0.0005.

In Figure 7, we evaluate the impact of the dimensionality d of the node embeddings on the performance of MELANIE. We vary d in $\{16, 32, 64, 128, 256\}$ and report the average RMSE over all the streaming minutes. In all datasets, MELANIE achieves the lowest RMSE when setting the node embedding dimensions to $d = 64$. By increasing the node embedding dimensions, MELANIE has no performance gain in the LiveStream-1 dataset, while the prediction accuracy degrades significantly in LiveStream-2 and LiveStream-3.

In Figures 3 and 4, we evaluate the influence of the replay memory buffer size K on the link prediction accuracy and the training time of the MELANIE model, respectively. We vary the number of stored state-action transitions in the replay memory buffer in $K = \{16, 32, 64, 128, 256\}$. We observe that in all datasets the MELANIE model achieves the best link prediction accuracy when fixing $K = 128$. Increasing the size of the replay memory buffer negatively impacts the prediction performance of MELANIE. This occurs because after selecting more than 128 state-action transitions, the less divergent experiences are considered when training the model, provided that the stored state-action transitions in the replay memory buffer are based on the KL-divergence of the viewers' throughputs. Figure 4 shows that large K values of the replay memory buffer size significantly increase the training time.

In Figures 5 and 6, we examine the influence of the graph signature buffer size C on the link prediction accuracy and the inference/testing time in the query set $\mathcal{E}^q$. In LiveStream-1, MELANIE converges to 15 graph signatures/events. Instead, MELANIE converges when trained over 20 graph signatures/events in LiveStream-2 and LiveStream-3. This means that LiveStream-1 events have high structural similarities, when compared with LiveStream-2 and LiveStream-3 events. Moreover, the inference time in LiveStream-3 is significantly higher than LiveStream-1 and LiveStream-2. This occurs because the LiveStream-3 events attracted more viewers than the other two datasets. Therefore, in LiveStream-3 the MELANIE model maintains many interactions which negatively impacts the inference time to compute the global model.

REFERENCES

[1] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, pp. 79–86, 1951.

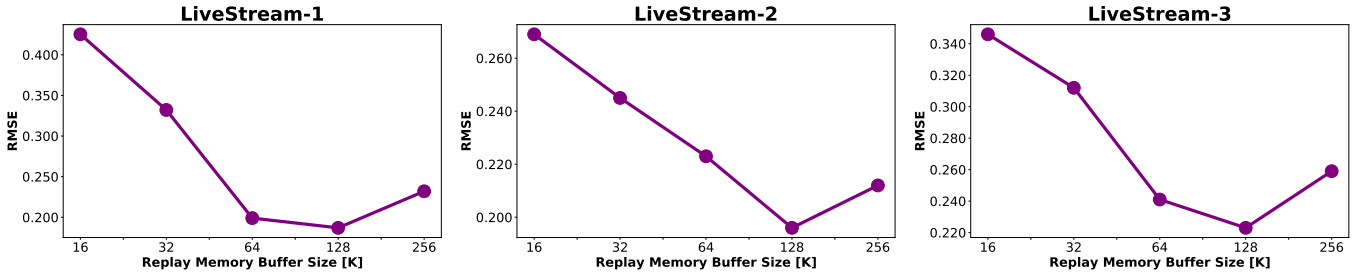

Fig. 3. Impact of the replay memory buffer size D on the link prediction task in terms of RMSE

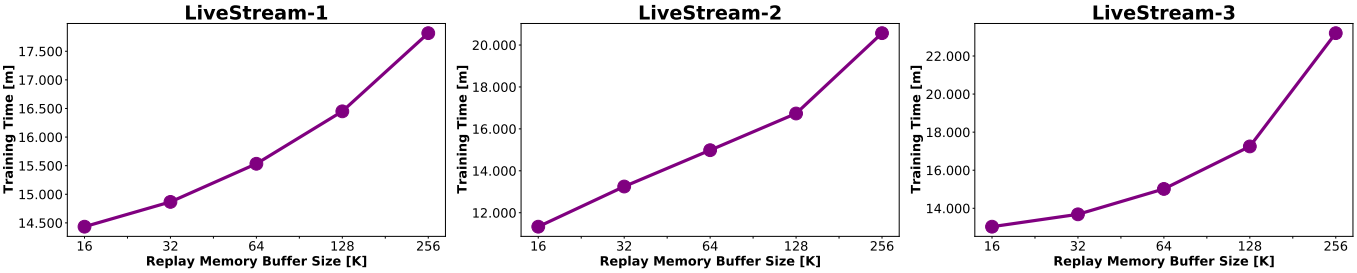

Fig. 4. Impact of the replay memory buffer size D on the training time (minutes)

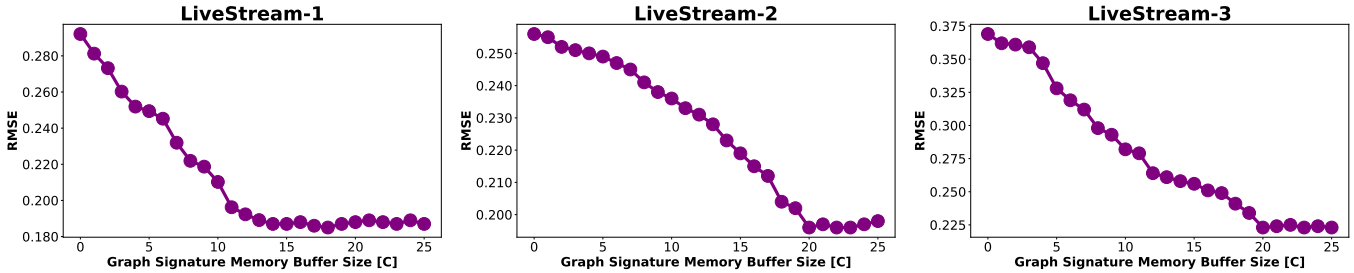

Fig. 5. Impact of the graph signature buffer size C on the link prediction task in terms of RMSE

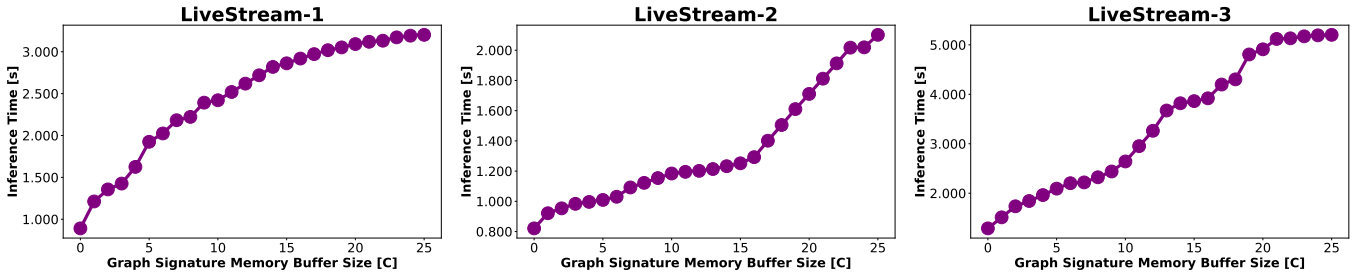

Fig. 6. Impact of the graph signature buffer size C on the inference/testing time (seconds)

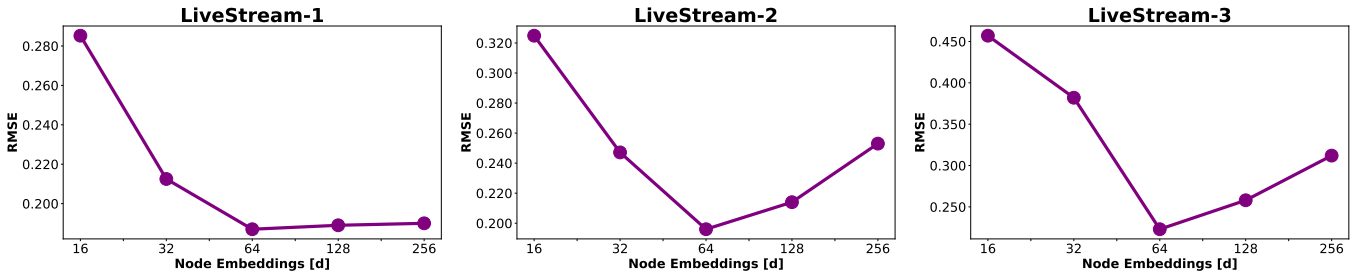

Fig. 7. Impact of the node embedding dimensionality d on the link prediction task in terms of RMSE