Latency Synchronization for Social VR with Mobile Edge Computing

Ta-Che Hsiao[†], De-Nian Yang[♯], and Wanjiun Liao[†]
[†]Department of Electrical Engineering, National Taiwan University, Taiwan
[♯]Institute of Information Science, Academia Sinica, Taiwan
E-mail: d09921016@ntu.edu.tw, dnyang@iis.sinica.edu.tw, and wjliao@ntu.edu.tw

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838893

***Abstract*—While mobile edge computing (MEC) potentially supports the stringent latency requirements for Virtual Reality (VR), previous research only considers minimizing the latency of transmitting required data and ignores the latency consistency of a group of friends in social VR. In this paper, we leverage the human motion prediction to eliminate the inconsistency among users in a group while ensuring the low latency interaction between friends with the aid of MEC. Moreover, to capture the accurate users' interaction behaviors, it is critical to jointly predict the motions of users with spatially close proximity in VR. Accordingly, we formulate a new problem, named Group Motion Prediction for Social VR problem (GMSV), with group consistency requirement and the objective of prediction costs minimization on the MEC server. Then, we prove that GMSV is NP-Hard and design a new algorithm, named Social-Aware Latency Synchronization for Remote Users (SLSR), to select an appropriate set of remote users in social VR for motion prediction, with the ideas of generating a *partial order set* of *remote subgroups* and extracting the remote subgroups with high *social interaction utilities*. Simulation results show that SLSR can effectively increase latency consistency by more than 50% compared with the baseline schemes.**

I. Introduction

Virtual Reality (VR) has recently fostered various new multimedia applications to enable immersive experiences in the virtual world. In 2020, HTC launched a new app, Vive Sync,[1] that allows users to collaborate in a virtual meeting space without contact in the real world. VR applications require extremely low latency to prevent motion sickness [1] and support real-time social interaction. Mobile edge computing (MEC) is a promising way to meet the ultra-low latency requirement of VR. Typically, a VR sensor uploads motion tracking information of a remote user (i.e., body movement) via network transmission to a MEC server. The MEC server updates each user's state in the virtual environment and then renders the images for local VR users [2, 3] (see Fig. 1). However, long-distance and extensive user's state information transmitted from one user to another usually incurs high end-to-end latency, undermining the user's mutual understanding and task performances in the VR world [4].

Recently, VR rendering in MEC attracted increasing attention [1, 5], which improves the quality of services for each user by offloading computationally intensive rendering on MEC to reduce the motion-to-photon (MTP) latency, i.e., the time between the movement of a user and the view's update on head-mounting displays (HMD) of another user. It also reduces interaction latency, which is the end-to-end latency between two interacting VR users [2, 3]. However, previous works ignore the latency consistency for the members in a VR group with close interactions. Since each remote user is usually located in different physical locations, heterogeneous end-to-end latency to local VR users leads to motion inconsistency displayed in the HMD of local users, impairing the immersive experience of VR with vivid social interactions [6]. Moreover, previous research did not leverage the spatial and interaction information in VR to predict the motion of remote users with high end-to-end latency.

Research in Computer Graphics has proposed effective deep learning approaches to predict human motion by leveraging previous motion and interaction information [7, 8]. However, human motion prediction is computationally intensive due to complicated model structures and thereby not feasible to support all remote users in a MEC server with limited resources [5, 7]. On the other hand, since a local user may interact with only a part of remote users, e.g., people team up with friends in Facebook Horizon[2] to explore the virtual world, the motions of other irrelevant remote users are not critical to the local user. Therefore, it is essential for a MEC server to select an appropriate set of remote users with close social interactions for motion prediction according to its available resources.

Finding the set of remote VR users for motion prediction in MEC is challenging due to the following reasons.

1) Due to limited resources in a MEC server, motion prediction for a popular remote user is desired since it contributes to more local users with close interactions, i.e., effectively minimizing the total *prediction costs* (detailed later). It is also important to ensure the *average latency* and *social cohesion requirements* for each local VR user.
2) Predicting the motion of each user independently will fail to capture influences from the motions of nearby friends in a crowded environment [9]. To accurately forecast the motion of a user, the motion predictions for nearby friends are sometimes required due to their interactions[3] (e.g., shaking hands or hugging) [8, 9, 10]. Therefore, the *motion influence requirement* is crucial to identify nearby remote

[1]https://enterprise.vive.com/us/solutions/vive-sync/

[2]https://www.oculus.com/facebook-horizon/

[3]Interaction behaviors can not be captured if the nearby user's motion is not evaluated.

users who influence the motion of a selected remote user[4]. However, it may require additional motion predictions and drive up the total prediction costs.

3) Synchronizing latency for a group of friends that interact closely is essential for social VR since the latency inconsistency between closely interacting users will undermine the social VR experience [6]. VR users in different locations transmit motion states via the network, inducing the latency inconsistency naturally. The inconsistency degrades a local user's sense of presence in an interaction event[5]. Therefore, the *latency gap requirement* is necessary for a MEC server to shorten the latency differences among interacting users.

In this paper, therefore, we formulate a new optimization problem, called Group Motion Prediction for Social VR (GMSV), to minimize the total prediction costs while ensuring the average end-to-end latency requirement, the motion influence requirement, and the latency gap requirement. We prove that GMSV is NP-Hard and design a new algorithm, named Social-Aware Latency Synchronization for Remote Users (SLSR), to select a set of remote users for motion prediction. It first extracts *core remote users* in each group for motion prediction to ensure latency gap requirement. Next, a proposed *Partial Order Set* (POS) of *remote subgroups* summarizes the dependencies among remote users from the motion influence requirement. The *shared interaction utility* evaluates the potential average latency that can be reduced by selecting a remote subgroup. Afterward, SLSR iteratively extracts a set of additional remote subgroups with a low ratio of cost to shared interaction utility. Simulation results demonstrate that SLSR increases latency consistency among remote VR users by more than 50% compared with existing MEC solutions.

II. Problem Formulation

A. System model and problem formulation

In the following, we describe the system model, where the notation table is presented in [11]. Let L denote the set of local wireless VR users. Each local user in L is associated with a base station (BS) equipped with a MEC server. Let R denote a set of remote VR users. Each remote user uploads the motion state to the MEC server via the backhaul transmission for VR image rendering and human motion prediction. Motion prediction infers the VR user's motion according to recent motion data and user interactions with virtual objects (and other users) [8, 9, 12]. Let p_r denote the *computation cost* to a remote user $r \in R$ (i.e., the required computational cycles to predict the motion of r). Let x_r denote the decision variable on motion prediction for remote user r, i.e., $x_r = 1$ if the MEC server selects r for motion prediction. According to [13], a motion assessment model measures the quality of predicted human motion by a pre-trained deep learning model that outputs the motion inaccuracy (i.e., the evaluated probability that the given human motion is artificially generated). Let $\mathrm{m}_r \in [0, 1]$ denote the motion inaccuracy of r obtained by the motion quality assessment model[6]. Following [15], the *prediction cost* of selecting r for motion prediction is $\mathrm{p}_r + \mathrm{m}_r$.

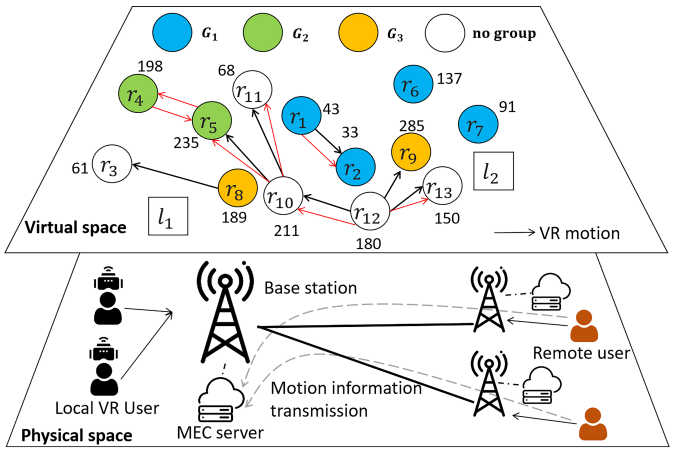

Fig. 1: Illustrative example of CPOSG.

According to [2, 3], the VR end-to-end latency between a local user ℓ and a remote user r contains three parts:

1) *Motion update latency* $d_r^{\mathrm{up}} = (1 - x_r) \cdot d_r^{\mathrm{ul}}$ is the time to update the motion of a remote user r depending on whether MEC predicts the motion. For remote users without motion prediction (i.e., $x_r = 0$), *upload latency* $d_r^{\mathrm{ul}} > 0$ includes the backhaul and wireless uplink latency to upload the motion state of r to the MEC server. For remote users with motion prediction (i.e., $x_r = 1$), following [16], the MEC server predicts the current motion of r and thereby does not require r to update and upload the motion state, i.e., $d_r^{\mathrm{up}} = 0$.
2) *Computation latency* $d_\ell^{\mathrm{c}} > 0$ includes the MEC processing time to render the VR scene for a local user ℓ and the prediction inference latency[7] d^{p} to generate the motions of remote users r with $x_r = 1$.
3) *Download latency* $d_\ell^{\mathrm{dn}} > 0$ is the time that BS transmits the VR image to ℓ.

Let $f_{r,\ell} \in [0, 1]$ denote *normalized interaction rate*[8] between r and ℓ, where $\sum_{r \in R} f_{r,\ell} = 1$. To guarantee low end-to-end latency, according to [4], the *average latency* for ℓ cannot exceed a latency requirement D_ℓ as follows,

$$\sum_{r \in R} f_{r,\ell} \cdot d_r^{\mathrm{ul}} \cdot (1 - x_r) + d_\ell^{\mathrm{c}} + d_\ell^{\mathrm{dn}} \leq D_\ell, \quad \forall \ell \in L. \tag{1}$$

[4]The motion influence can be inferred from spatial information, e.g., moving direction, speed, etc. [10].

[5]For users engage in a running race in Facebook Horizon, a local user viewing the race may misunderstand the result when a leading remote user with larger end-to-end latency is displayed as a follower in the race.

[6]The expected motion inaccuracy can be derived by infinite-armed bandit [14] that exploits the upper confidence bound to estimate the value of the most likely motion quality. If the motion of a remote user has never been predicted, then $\mathrm{m}_r = 0$.

[7]Following [16], d^{p} is negligible small, since the MEC server advances the prediction tasks to generate the motions of remote users in the near future, which hides the transmission and the inference latency measured in [17]. A Multi-core CPU server can execute different predictions in parallel [18].

[8]Following [19], the interaction rate can be derived from user's gaze behavior detected by eye tracker. It is set to 0 if two users have no interaction.

Let $\mathcal{G} = \{G | G \subseteq R\}$ denote the collection of *social interaction groups* G. An interaction group consists of users participating in the same event in VR[9], e.g., sports team, which can be formed by the service provider or detected by VR users' behaviors according to [20]. We define *latency gap* as the difference of the motion update latency between two group members. To ensure consistency for the users in an interaction group, following [21], latency gap cannot exceed a latency gap threshold γ. The *latency gap constraint* is as follows,

$$|d_u^{\text{up}} - d_v^{\text{up}}| \leq \gamma, \quad \forall u, v \in G \in \mathcal{G}. \tag{2}$$

Human motion is largely influenced by spatially proximal neighbors in crowded spaces due to social norms, e.g., avoiding possible collisions. Therefore, according to [8, 9], it is also necessary to evaluate and predict the possible motions of surrounding people during the prediction for a remote user in VR. The *motion influence weight* from a user to another user can be derived according to the distance and moving direction [10]. Let $\mathrm{N}_r \subseteq R$ denote the *motion influence set*, which is the set of neighbor users with motion influence weights to remote user r exceeding a threshold μ set according to [10]. Therefore, the *motion influence constraint* enforces that if a remote user r is selected (i.e., $x_r = 1$), the remote users in N_r also need to be chosen as follows,

$$x_v \geq x_r, \quad \forall r \in R, v \in \mathrm{N}_r. \tag{3}$$

Since friend relations play a crucial role in various social VR applications, establishing social cohesion facilitates long-time engagement for users in social VR [22]. Let $\mathtt{R}_\ell \subseteq R$ denote the set of remote users who are friends with ℓ, and $\mathtt{R}_\ell^{\mathtt{l}} \subseteq \mathtt{R}_\ell$ denotes the subset of the remote friends with large upload latency that deteriorates the quality of social interactions for ℓ. $\mathtt{R}_\ell^{\mathtt{s}} = \mathtt{R}_\ell \setminus \mathtt{R}_\ell^{\mathtt{l}}$ denotes the subset of remote friends with small upload latency. According to social community research [23, 24], finding a minimum proportion of friends to interact with users is vital in social events to foster warm atmosphere. Thus, it is important to ensure small upload latency for a proportion of friends [23]. Specifically, the social cohesion constraint requires the total number of 1) remote friends with low upload latency (i.e., $\|\mathtt{R}_\ell^{\mathtt{s}}\|$) and 2) remote users with high upload latency selected for motion prediction (i.e., $\sum_{r \in \mathtt{R}_\ell^{\mathtt{l}}} x_r$) to exceed a minimum proportion of remote friends $\sigma \cdot \|\mathtt{R}_\ell\|$ as follows,

$$\|\mathtt{R}_\ell^{\mathtt{s}}\| + \sum_{r \in \mathtt{R}_\ell^{\mathtt{l}}} x_r \geq \sigma \cdot \|\mathtt{R}_\ell\|, \forall \ell \in L. \tag{4}$$

From the above observations, we formulate GMSV as follows.

Problem: Group Motion Prediction for Social VR.

Given: Local user set L, remote user set R, collection of interaction groups $\mathcal{G}$, minimum ratio of friends with small upload latency σ, latency gap threshold γ, upload latency d_ℓ^{ul}, computation latency d_ℓ^{c}, download latency d_ℓ^{dn}, latency requirement D_ℓ, set of remote friends $\|\mathtt{R}_\ell\|$, set of remote friends with large upload latency $\mathtt{R}_\ell^{\mathtt{l}}$ for each local user $\ell \in L$, upload latency d_r^{ul}, computation cost p_r, and the motion influence set N_r for each remote user $r \in R$; normalized interaction rate $f_{r,\ell}$ for each pair of users (r, ℓ).

Find: A subset of remote users $S \subseteq R$ for motion prediction such that the total prediction costs $\sum_{r \in S} (\mathrm{p}_r + \mathrm{m}_r) \cdot x_r$ are minimized, while the average latency, latency gap, social cohesion, and motion influence constraints are satisfied.

Theorem 1. GMSV is NP-hard.

Proof. We prove the theorem in [11] with a reduction from the minimum precedence-constraint knapsack problem (PCKP), which is NP-Hard [25]. □

III. ALGORITHM DESIGN

We design a new algorithm, named Social-Aware Latency Synchronization for Remote Users (SLSR), to carefully find a subset of remote users for motion prediction, and the pseudocode is presented in [11]. SLSR includes two phases. The first phase, named *Consistency Partial Order Set Generation* (CPOSG), carefully examines the latency gap and VR spatial proximity among interacting users to find *core remote users* and relations among other remote users. It then generates a directed graph represented by Partial Order Set (POS) and identifies *remote subgroups* with remote users required to be predicted jointly, in order to ensure the latency consistency while selecting them. The second phase, named *Remote Subgroup Selection* (RSS), chooses more remote subgroups for motion prediction to foster social cohesion with low average latency. RSS evaluates the *social interaction utility* of each remote subgroup that includes 1) the total number of remote friends with large upload latency and 2) the total amount of average latency contributed to each local user. Following POS, RSS iteratively extracts a *smallest order subgroup* with a low cost and a high social interaction utility to satisfy the social cohesion constraint and the average latency constraint.

A. Consistency Partial Order Set Generation (CPOSG)

For an interaction group, CPOSG first finds the core remote users to ensure the latency gap constraint. Specifically, for any pair of users in the same interaction group $u, v \in G \in \mathcal{G}$ with a large latency gap, i.e., $\left|d_u^{\text{ul}} - d_v^{\text{ul}}\right| > \gamma$, CPOSG selects the remote user u with $d_u^{\text{ul}} > d_v^{\text{ul}}$ as a core remote user to predict the motion of u (i.e., $x_u = 1$ such that $d_u^{\text{up}} = 0$). If v's upload latency also exceeds γ (i.e., $\left|d_v^{\text{ul}} - 0\right| > \gamma$), v also acts as a core remote user (i.e., $x_v = 1$) to ensure that the MEC server predicts both remote users' motion to eliminate the latency gap. For other pairs of users $u, v \in G \in \mathcal{G}$ with a small latency gap, i.e., $\left|d_u^{\text{ul}} - d_v^{\text{ul}}\right| \leq \gamma$, CPOSG carefully finds the relations (two possible cases) for u and v to maintain a small latency gap. Case 1: When only d_v^{ul} is larger than γ, selecting u alone for motion prediction will incur latency inconsistency (i.e., $\left|d_v^{\text{up}} - d_u^{\text{up}}\right| = \left|d_v^{\text{ul}} - 0\right| > \gamma$). Therefore, the relation for u and v is $x_u \leq x_v$, representing that if u is selected, then v is also required to be chosen. Case 2: When both d_u^{ul} and d_v^{ul} are larger than γ, selecting only u or only v for motion prediction will also lead to latency inconsistency. Therefore, the relation

[9]A user can participate in multiple interaction groups.

of u and v is $x_v = x_u$, representing that u and v need to be determined jointly.

Fig. 1 presents an illustrative example, where each square in the virtual space is a local user, and each circle is a remote user colored by its interaction group. Each black directed arrow from r_u to r_v indicates that $r_v \in \mathrm{N}_u$. The upload latency of a remote user is presented beside the circle with the latency gap threshold γ assumed as 60. For $r_1, r_6 \in G_1$, CPOSG selects r_6 as a core remote user due to the large upload latency of r_6 and the small upload latency of r_1. For $r_8, r_9 \in G_3$, CPOSG chooses r_8 and r_9 as core remote users due to the large latency gap and the large upload latency for both of them. For $r_4, r_5 \in G_2$, $x_4 = x_5$ since the latency gap constraint is not satisfied when only one of them is selected. For $r_1, r_7 \in G_1$, $x_1 \leq x_7$ because the latency gap constraint is not satisfied when r_1 is selected but r_7 is not chosen. Accordingly, the set of core remote users is $\{x_6, x_8, x_9\}$ with the relations as $x_1 \leq x_7, x_2 \leq x_7, x_6 = x_7, x_4 = x_5$.

CPOSG then incorporates more core remote users according to the relations derived above and the motion influence requirement. Let S^c denote the set of core remote users obtained so far. For each core remote user u in S^c, CPOSG adds 1) each additional user v with relation $x_u \leq x_v$ or $x_u = x_v$ derived from the latency gap constraint, e.g., adding r_7 since $x_7 = x_6$, and 2) every user in the motion influence set N_u, e.g., adding r_3 since r_3 influences the motion of r_8 as in Fig. 1. CPOSG iteratively performs the above step until no remote user can be added to S^c, and $S^c = \{r_3, r_6, r_7, r_8, r_9\}$ with the relation $x_4 = x_5$ for remote users not in S^c.

To choose more remote users, CPOSG constructs a directed graph $\mathtt{G} = \{\mathtt{V}, \mathtt{E}\}$ with the vertex set $\mathtt{V} = R \setminus S^c$. A directed edge $\overrightarrow{uv}$ is included in $\mathtt{E}$ if $x_u \leq x_v$ or $v \in \mathrm{N}_u$, and both $\overrightarrow{uv}$ and $\overrightarrow{vu}$ appear in $\mathtt{E}$ when $x_u = x_v$. A cycle in $\mathtt{G}$ indicates that selecting a remote user with a vertex in the cycle needs to include all the other remote users in the cycle. Therefore, CPOSG iteratively finds a cycle by Depth-First Search and contracts (i.e., merges) it into a vertex because the MEC server needs to predict them as a whole. The resulting graph becomes a directed acyclic graph (DAG) because CPOSG merges each cycle into a vertex. Each vertex in DAG represents a *remote subgroup*. Following the example in Fig. 1, $\mathtt{V} = \{r_1, r_2, r_4, r_5, r_{10}, r_{11}, r_{12}, r_{13}\}$ since they are not in S^c. Each directed edge in $\mathtt{E}$ is presented as a red directed arrow. Two directed edges $\overrightarrow{r_4 r_5}$ and $\overrightarrow{r_5 r_4}$ appear in $\mathtt{E}$ since $r_5 = r_4$, and other directed edges $\overrightarrow{uv}$ belong $\mathtt{E}$ when $v \in \mathrm{N}_u$. Next, since $\overrightarrow{r_4 r_5}$ and $\overrightarrow{r_5 r_4}$ form a directed cycle, CPOSG merges r_4 and r_5 into a vertex, creating a remote subgroup $\mathtt{R}_4 = \{r_4, r_5\}$.

Afterward, we transfer DAG into a partial order set P to act as a cornerstone in the next phase. $P = (\mathbf{R}, \preceq)$ includes 1) $\mathbf{R}$ as the set of remote subgroups, i.e, the vertices in DAG, and 2) the order relation $\mathtt{R}_v \preceq \mathtt{R}_u$ corresponding to a directed edge $\overrightarrow{uv}$ in DAG. For the example in Fig. 2, the partial order set P includes remote subgroups $\mathtt{R}_i = \{r_i\}$ for $i = 1, 2, 10, 11, 12, 13$ and $\mathtt{R}_4 = \{r_4, r_5\}$. Moreover, a remote subgroup $\mathtt{R}_u \in \mathbf{R}$ is a *smallest order subgroup* if there is no remote subgroup $\mathtt{R}_v \in \mathbf{R}$ with $\mathtt{R}_v \preceq \mathtt{R}_u$. On the other hand, a remote subgroup $\mathtt{R}_u$ is a *large order subgroup*, if it is not a smallest order subgroup. For the example in Fig. 2, $\mathtt{R}_4, \mathtt{R}_{11}, \mathtt{R}_{13}, \mathtt{R}_2$ are smallest order subgroups, which are presented as red circles. $\mathtt{R}_1, \mathtt{R}_{10}, \mathtt{R}_{12}$ are large order subgroups.

TABLE I: Prediction costs of each subgroup in Fig. 2.

j	1	2	4	10	11	12	13
c_j^T	10	3	20	10	60	20	40

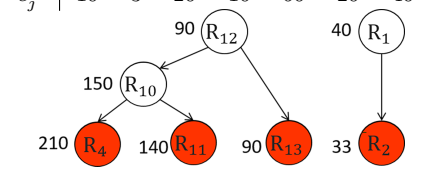

Fig. 2: Illustrative example of partial order set.

B. Remote Subgroup Selection (RSS)

This phase selects more remote subgroups to satisfy the average latency constraint and the social cohesion constraint. According to the partial order relations in P obtained in CPOSG, RSS iteratively adds a remote subgroup into the solution set $\mathbf{S}^r$ from the collection of all smallest order subgroups. To further evaluate the contribution of a selected remote subgroup for social cohesion and average latency, we define the *social interaction utility* of a remote subgroup $\mathtt{R}_u$ as $w_u = \sum_{\ell \in L} \|\mathtt{R}_u \cap \mathtt{R}_\ell^{\mathrm{l}}\| + \sum_{\ell \in L} \sum_{r \in \mathtt{R}_u} f_{r,\ell} \cdot d_r^{\mathrm{ul}}$, which includes 1) the total number of remote friends of all local users in $\mathtt{R}_u$ with large upload latency, and 2) the total amount of average latency that can be reduced for all local users by selecting all remote users in $\mathtt{R}_u$.

Since the partial order relations in P follow the latency gap constraint and the motion influence constraint, RSS ensures that a large order subgroup $\mathtt{R}_u$ can be selected only if every remote subgroup $\mathtt{R}_v$ with $\mathtt{R}_v \preceq \mathtt{R}_u$ is also chosen in $\mathbf{S}^r$. In other words, selecting a smallest order subgroup $\mathtt{R}_v$ not only contributes to ensuring the constraints (i.e., average latency and social cohesion), but also provides a chance to select large order subgroups $\mathtt{R}_u$ with $\mathtt{R}_v \preceq \mathtt{R}_u$ in P. Let $\mathcal{E}(\mathbf{S}^r)$ denote the collection of all smallest order subgroups in the candidate set $\mathbf{R} \setminus \mathbf{S}^r$. For the example in Fig. 2, given $\mathbf{S}^r = \{\mathtt{R}_2\}$, $\mathcal{E}(\mathbf{S}^r)$ is $\{\mathtt{R}_1, \mathtt{R}_4, \mathtt{R}_{11}, \mathtt{R}_{13}\}$. In order to prioritize large order subgroups with a high social interaction utility, RSS also evaluates the potential social interaction utilities of the large order subgroups if selecting the smallest order subgroups. More specifically, we define *shared interaction utility* $w_v^{\mathrm{s}}(\mathbf{S}^r) = w_v + \sum_{\mathtt{R}_v \preceq \mathtt{R}_u} \frac{w_u}{x_u(\mathbf{S}^r)}$ of a smallest order subgroup $\mathtt{R}_v$ to further evaluate the social interaction utilities of every large order subgroup $\mathtt{R}_u$, where $x_u(\mathbf{S}^r)$ is the number of smallest order subgroups $\mathtt{R}_k \in \mathcal{E}(\mathbf{S}^r)$ with $\mathtt{R}_k \preceq \mathtt{R}_u$ [10]. For the example in Fig. 2, the social interaction utility of each remote subgroup is presented beside the circle. $\mathtt{R}_{12}$ shares its social interaction utility to $\{\mathtt{R}_4, \mathtt{R}_{11}, \mathtt{R}_{13}\}$, and $\mathtt{R}_{10}$ shares its social interaction utility to $\{\mathtt{R}_4, \mathtt{R}_{11}\}$. Accordingly, for a smallest order subgroup $\mathtt{R}_4$, $w_4^{\mathrm{s}}(\emptyset)$ is $210 + \frac{90}{3} + \frac{150}{2}$.

[10] Since selecting a large order subgroup $\mathtt{R}_u$ may require multiple smallest order subgroups in $\mathcal{E}(\mathbf{S}^r)$ to be chosen, each large order subgroup divides its social interaction utility evenly onto them, i.e., $\frac{w_u}{x_u(\mathbf{S}^r)}$.

RSS iteratively extracts a smallest order subgroup $\mathtt{R}_v \in \mathcal{E}(\mathbf{S}^r)$ with a low ratio of the prediction costs $c_v^T = \sum_{r \in \mathtt{R}_v} (\mathrm{p}_r + \mathrm{m}_r)$ to the shared interaction utility to strikes a good balance between them. Specifically, given the empty solution set $\mathbf{S}^r = \emptyset$, RSS finds a smallest subgroup $\mathtt{R}_v$ in $\mathcal{E}(\mathbf{S}^r)$, which generates the minimum ratio of prediction costs to shared interaction utility (i.e., $\arg\min_{\mathtt{R}_v \in \mathcal{E}(\mathbf{S}^r)} c_v^T / w_v^{\mathrm{s}}(\mathbf{S}^r)$). For every other smallest order subgroup $\mathtt{R}_k$, RSS reduces the prediction costs of $\mathtt{R}_k$ to increase the chance for selecting it later, in order to allow more large order subgroups $\mathtt{R}_u$ with $\mathtt{R}_k \preceq \mathtt{R}_u$ to be chosen. Thus, RSS decreases the prediction costs of a smallest subgroup $\mathtt{R}_k$ to $c_k^T - w_k^{\mathrm{s}}(\mathbf{S}^r) \cdot c_v^T / w_v^{\mathrm{s}}(\mathbf{S}^r)$, which renders a lower prediction costs when the ratio $c_k^T / w_k^{\mathrm{s}}(\mathbf{S}^r)$ is closer to $c_v^T / w_v^{\mathrm{s}}(\mathbf{S}^r)$. Next, RSS prioritizes $\mathtt{R}_v$ for solution set (i.e. $\mathbf{S}^r$ adds $\mathtt{R}_v$).

Following the example in Fig. 2, $\mathbf{S}^r = \emptyset$, RSS finds the shared interaction utilities of every smallest order subgroups $\{\mathtt{R}_4, \mathtt{R}_{11}, \mathtt{R}_{13}, \mathtt{R}_2\}$ as $w_4^{\mathrm{s}}(\emptyset) = 210 + \frac{90}{3} + \frac{150}{2} = 315, w_{11}^{\mathrm{s}}(\emptyset) = 140 + \frac{90}{3} + \frac{150}{2} = 245, w_{13}^{\mathrm{s}}(\emptyset) = 90 + \frac{90}{3} = 120$, and $w_2^{\mathrm{s}}(\emptyset) = 33 + \frac{40}{1} = 73$. According to the prediction costs listed in Table I, the ratios of these subgroups are $\frac{20}{315}, \frac{60}{245}, \frac{40}{120}, \frac{3}{73}$ respectively. Hence, RSS selects the subgroup $\mathtt{R}_2$ for motion prediction. RSS reduces the prediction costs of $\mathtt{R}_4, \mathtt{R}_{11}, \mathtt{R}_{13}$ to $c_4^T = 20 - 315 \cdot \frac{3}{73}, c_{11}^T = 60 - 245 \cdot \frac{3}{73}, c_{13}^T = 40 - 120 \cdot \frac{3}{73}$. Finally, RSS updates the solution set $\mathbf{S}^r = \{\mathtt{R}_2\}$.

Accordingly, in each iteration, RSS includes the following steps: 1) update the set of smallest order subgroups $\mathcal{E}(\mathbf{S}^r)$, 2) update the shared interaction utility of each smallest order subgroup, 3) extract a smallest order subgroup with the minimum ratio to the solution set $\mathbf{S}^r$, and 4) reduce the prediction costs of other smallest subgroups. RSS repeats the above procedure until the average latency constraint and the social cohesion constraint are satisfied for every local user. RSS outputs the final solution set[11] $S = S^c \cup S^r$ for motion prediction. More algorithm details can be found in [11].

IV. Simulation

A. Simulation setup

We evaluate the performance of SLSR in a base station with 1GHz bandwidth and MEC CPU resource as 4GHz [26]. Following [3], we choose 20 coffee shops in Los Angeles as the locations of BSs. One is selected as the local BS, and the others are the locations of randomly distributed remote users with the default number of 60 (following [2]). The transmission power of each BS is set to 24 dBm. The path loss model is typical urban, while the shadowing model is log-normal with zero mean and σ^2 variance, and the standard deviation $\sigma = 8$. The backhaul latency follows Gamma distribution with an average value proportional to the distance [2]. We randomly distribute 20 local users (following [27]) in the transmission range of the local BS. The resolution of VR images is set to 1920×1080. Latency requirement for each local user is set to 100ms with γ as 60ms, σ as 0.3, and μ as 0.7 [4]. We test two VR scenarios, concentration and average, in a $50\mathrm{m} \times 50\mathrm{m}$ area in VR. For the concentration scenario, we randomly deploy 70% of users in a $20m \times 20m$ area, and the remaining users are randomly deployed over the remaining area. For the average scenario, we randomly distribute users over the whole area. We randomly assign the view direction and the walking direction for each user, and the interaction rate is randomly set from 0 to 1 [28]. The friend relations are randomly set between the local users and the remote users (following [2]). The motion influence weights are derived according to [10]. The social interaction groups are formed by F-formation detection [20]. For each remote user, the computation cost is randomly assigned from 10MHz to 40MHz [26], and the motion quality is randomly set from 0 to 1.

Since there is no related work exploring GMSV, we compare SLSR with a motion selection scheme and two resource allocation schemes in MEC: Motion Attention scheme (AT) [10], Minimize end-to-end latency Bandwidth Allocation scheme (BA) [2], and Earliest-Finish time Offloading algorithm (EFO) [27]. To evaluate SLSR, we change the following parameters: 1) number of remote users, 2) latency gap threshold γ, 3) MEC computation capacity, and 4) social cohesion ratio σ. We measure the following performance metrics: 1) computation costs, 2) consistent pairs[12], and 3) maximum average latency of local users. Each result is averaged over 100 samples. In addition, we find the optimal solution solved by Gurobi. More simulation results are presented in [11].

B. Simulation results

Figs. 3 presents the simulation results in the *average* and *concentration* scenarios. The computation costs gradually increase when more remote users interact with the local users in VR. However, SLSR induces much smaller computation costs (nearly optimal) since it correctly finds core remote users and other remote users with a low ratio of prediction costs to shared interaction utility. In Figs. 3(a) and 3(b), more computation costs are required in the concentration scenario, because the remote users clustered in a small area largely influence each other's motion. The computation costs incurred by the baselines remain high since they select each remote user independently without considering the relations of latency consistency among them. When both numbers of remote users increase in Figs. 3(c) and 3(d), the maximum average latency grows since local users interact with more remote users, leading to a less normalized interaction rate to each remote user and then causing a smaller amount of average latency reduced by motion predictions. Therefore, SLSR exploits the social interaction utility to find the remote users that substantially contribute to the average latency requirement. On the other hand, the number of consistent pairs increases when more MEC resources are equipped for motion prediction as shown in Figs. 3(e) and 3(f), SLSR carefully follows the relations derived from the latency gap constraint to ensure latency consistency when extracting low-cost remote users. In summary, SLSR is much closer to optimal. It not only

[11] S^c is obtained in the first phase CPOSG, and S^r refers to the set of selected remote users in RSS, i.e., $\{r | r \in \mathtt{R}_i \in \mathbf{S}^r\}$.

[12] The number of the user pairs with small latency gaps.

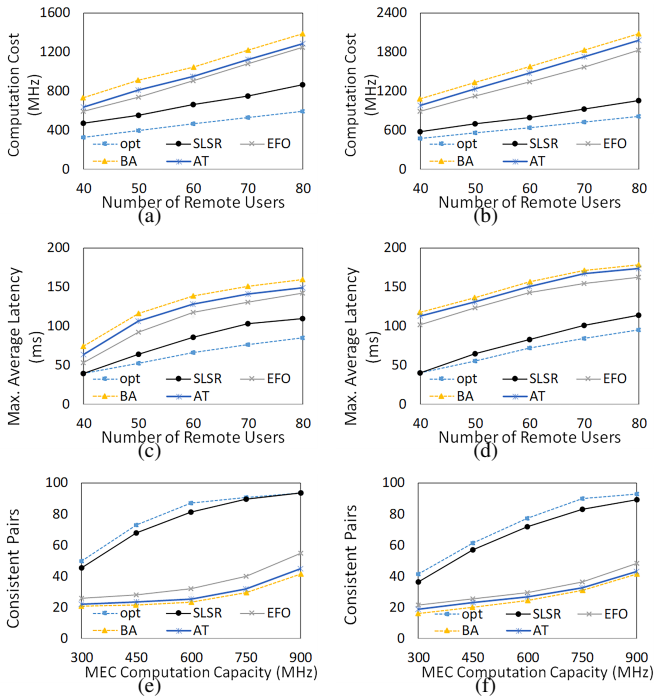

Fig. 3: Simulations in average (right) and concentration (left) scenarios.

effectively reduces the computation costs by more than 40% but also increases the number of consistent pairs of VR users by more than 50% compared with the baselines.

V. Conclusion

For social VR applications with MEC, we carefully analyze the social relations to achieve latency consistency while minimizing the motion prediction costs. We first formulate a new optimization problem GMSV under the average latency, latency gap, social cohesion, and motion influence constraints. We then design a new algorithm SLSR to select a subset of remote users for motion prediction with the ideas of the partial order set and the social interaction utility of each remote subgroup. Simulation results show that SLSR effectively increases latency consistency by more than 50% compared with the baseline schemes.

References

[1] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Network*, vol. 32, no. 2, pp. 78–84, 2018.

[2] J. Park, P. Popovski, and O. Simeone, "Minimizing latency to support vr social interactions over wireless cellular systems via bandwidth allocation," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 776–779, 2018.

[3] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM*. IEEE, 2018, pp. 468–476.

[4] A. Becher, J. Angerer, and T. Grauschopf, "Negative effects of network latencies in immersive collaborative virtual environments," *Virtual Reality*, vol. 24, no. 3, pp. 369–383, 2020.

[5] X. Hou, J. Zhang, M. Budagavi, and S. Dey, "Head and body motion prediction to enable mobile vr experiences with low latency," in *2019 IEEE GLOBECOM*. IEEE, 2019, pp. 1–7.

[6] W. Wu, A. Arefin *et al.*, "" i'm the jedi!"-a case study of user experience in 3d tele-immersive gaming," in *2010 IEEE International Symposium on Multimedia*. IEEE, 2010, pp. 220–227.

[7] J. Martinez, M. J. Black, and J. Romero, "On human motion prediction using recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2891–2900.

[8] T. Yao, M. Wang, B. Ni, H. Wei, and X. Yang, "Multiple granularity group interaction prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2246–2254.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *IEEE CVPR*, 2016, pp. 961–971.

[10] F. Yang, H. Saikia, and C. Peters, "Who are my neighbors? a perception model for selecting neighbors of pedestrians in crowds," in *IVA*, 2018, pp. 269–274.

[11] T.-C. Hsiao, D.-N. Yang, and W. Liao, "Latency synchronization for social vr with mobile edge computing (full version)," *NTU Technical Report*, Oct 2021. [Online]. Available: https://kiki.ee.ntu.edu.tw/%7Ed09921016/VRPrediction.pdf

[12] L. Bruckschen, N. Dengler, and M. Bennewitz, "Human motion prediction based on object interactions," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–6.

[13] J. N. Kundu, M. Gor, and R. V. Babu, "Bihmp-gan: Bidirectional 3d human motion prediction gan," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 8553–8560.

[14] G. Burtini, J. Loeppky, and R. Lawrence, "A survey of online experiment design with the stochastic multi-armed bandit," *arXiv preprint arXiv:1510.00757*, 2015.

[15] S. Wang, Y. Ruan, Y. Tu, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," *IEEE/ACM Transactions on Networking*, 2021.

[16] Y. Horiuchi, Y. Makino, and H. Shinoda, "Computational foresight: Forecasting human body motion in real-time for reducing delays in interactive system," in *2017 ACM ISS*, 2017, pp. 312–317.

[17] L. L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, and Y. Liu, "nn-meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices," in *ACM MobiSys*, 2021, pp. 81–93.

[18] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *2017 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2017, pp. 1–7.

[19] L. A. Meerhoff *et al.*, "Guided by gaze: Prioritization strategy when navigating through a virtual crowd can be assessed through gaze activity," *Acta psychologica*, vol. 190, pp. 248–257, 2018.

[20] M. Cristani, L. Bazzani, G. Paggetti, A. Fossati, D. Tosato, A. Del Bue, G. Menegaz, and V. Murino, "Social interaction discovery by statistical analysis of f-formations." in *BMVC*, vol. 2, 2011, p. 4.

[21] M. Yu, M. Thottan, and L. Li, "Latency equalization as a new network service primitive," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 125–138, 2011.

[22] M. H. Schiller, G. Wallner *et al.*, "Inside the group: Investigating social structures in player groups and their influence on activity," *IEEE Transactions on Games*, vol. 11, no. 4, pp. 416–425, 2018.

[23] P. Lee and L. V. Lakshmanan, "Query-driven maximum quasi-clique search," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 522–530.

[24] A. I. Matin, S. Jahan, and M. R. Huq, "Community recommendation in social network using strong friends and quasi-clique approach," in *8th ICCEE*. IEEE, 2014, pp. 453–456.

[25] S. T. McCormick, B. Peis, J. Verschae, and A. Wierz, "Primal–dual algorithms for precedence constrained covering problems," *Algorithmica*, vol. 78, no. 3, pp. 771–787, 2017.

[26] J. Zhang, X. Hu *et al.*, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, 2017.

[27] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1678–1689, 2019.

[28] Y. Zhang, L. Jiao, J. Yan, and X. Lin, "Dynamic service placement for virtual reality group gaming on mobile edge cloudlets," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 8, pp. 1881–1897, 2019.