active in step $t-1$ remain active, and nodes that were abandoned in step $t-1$ remain abandoned; furthermore, each candidate node is accessed by a client. These clients coordinate with each other to decide the candidate nodes that should receive sub-events and select a candidate edge for each decided candidate node to propagate sub-events. All the selected edges collectively form a request to the recommender. The recommender responds to the request by generating the order of sub-event propagation for each selected edge. The sub-events are continuously propagated through these selected edges in the corresponding order until the decided nodes are activated or reach the message limit. This process ends after there are no candidate nodes in the graph.

### D. Deep Reinforcement Diffusion Optimization

This subsection discusses deep reinforcement learning techniques to optimize the diffusion forest. We first discuss the optimal diffusion forest could be approximated by retrieving the optimal node order. Then we propose a novel neural network model supporting the optimization of node coordination and edge selection. After that, we discuss how to train this model using hierarchical policy update algorithm.

*1) Node coordination and edge selection:* After selecting the seed nodes, the node set composing of all the nodes reachable by seed nodes under the message limit is settled. The final objective is to maximize the number of nodes within the set that are activated by their optimal neighbors. The optimal neighbor of a node is identified as the neighbor capable of consuming the fewest messages to activate the node. Due to the limitation of influence only flowing from active nodes to inactive nodes, it's not feasible to expect all candidate nodes to have the active optimal neighbors at the same diffusion step. Consequently, the activation of some candidates need to be delayed until their optimal neighbors are activated. Furthermore, situations may arise where none of the candidate nodes have active optimal neighbors. In such cases, it becomes necessary to activate some candidates through their sub-optimal neighbors. Although they are all sub-optimal activation chances for the candidates, they vary in their effects on future diffusion forest expansion. Specifically, prioritizing the activation of certain candidates can create more favorable conditions for the optimal activation of successor nodes within the diffusion forest. To effectively identify these candidates, incorporating the sub-tree rooted at each candidate node into the node coordination process is necessary.

In a word, the optimal diffusion forest is approximated by prioritizing the activation of candidates that either have active optimal neighbors or can create subsequent optimal activation chances. Prioritizing among the candidates at each diffusion step forms a node sequence. After retrieving the optimal node sequence, the optimal diffusion forest is obtained by activating nodes in the order of the node sequence and selecting the edge consuming the fewest messages for each activation.

Example 2. As depicted in Figure 1 (c), the optimal node sequence is $b, c, e, f$. Node b is firstly activated as its optimal neighbor, node d, is already active. After the activation of node b, both candidate nodes c and f could only be activated by their sub-optimal neighbors, nodes a and d, respectively. In this case, node c is secondly activated as the sub-tree rooted at c offers optimal activation chances for nodes e and f, while the sub-tree rooted at f just offers optimal activation chances for nodes e. Node e is thirdly activated since it has the active optimal neighbor, unlike node f. Lastly, node f is activated by its optimal neighbor, node e.

*Definition 5 (Constrained Optimal Neighbor):* Given an $\ell$-forest $F_\ell$, the candidate network $G_c$, for each node $v$ in $V_c$, there is a constrained optimal neighbor $\hat{u}_v$ in $V_F$ to form the constrained optimal input edge to $v$, which is represented as $\hat{u}_v = \arg\min_{u \in N^{V_F}(v)} M^{\mathbf{E}}(u, v)$.

*Definition 6 (K-hop Uninfluenced Network):* Given an $\ell$-forest $F_\ell$ and the original network G, the K-hop uninfluenced network $G^K_{ui}$ is a sub-network of G composing of all the candidate edges and all available nodes that are within a maximum distance of K hops from any of the forest nodes.

Definition 5 is used to illustrate the significance of node coordination when deciding a subset of candidate nodes to receive influence. If we were to activate all candidate nodes at the same diffusion step, the best outcome we could achieve is to activate each candidate through their constrained optimal neighbors. Such an approach would still leave a substantial quality gap between our achieved diffusion forest and the optimal diffusion forest, as each decision made would only be locally optimal. Definition 6 specifies the scope for future diffusion estimation is confined to the K-hop uninfluenced network. This scope ensures that the maximum depths of all candidate sub-trees are limited to K hops. Restricting the depths of sub-trees prevents the complex overlap of sub-trees across the entire uninfluenced network, which would otherwise obscure the distinct effects of different sub-trees on subsequent activations. This restricted sub-tree depth also helps maintain a balance between effectiveness and efficiency of our method. By limiting sub-tree analysis to the K-hop range, the computational overhead is significantly reduced compared with the unrestricted sub-tree depth.

**Lemma 1.** $\delta(\hat{u}_v, v) + \tilde{\delta}(\mathcal{E}^K_{ui} \setminus \mathcal{E}^{in}_v) - \tilde{\delta}(\mathcal{E}^K_{ui}) \leq 0$, where $\hat{u}_v \in V_F$ and $v \in V_c$.

**Proof.** $\hat{u}_v$ is the constrained optimal neighbor of $v$, although it is not necessarily the optimal neighbor of $v$ since that one might not have been activated yet. However, $\tilde{\delta}(\mathcal{E}^K_{ui})$ is estimated by assuming the maximum number of nodes are activated by its optimal neighbor in $V^K_{ui}$.

**Lemma 2.** Under the condition $\Delta\mathcal{E}^{K+1}_{\hat{u}_v} \subset \mathcal{E}^K_{ui}$,

$$\tilde{\delta}\left((\mathcal{E}^K_{ui} \setminus \mathcal{E}^{in}_v) \cup (\Delta\mathcal{E}^{K+1}_{ui} \cap \Delta\mathcal{E}^{K+1}_{\hat{u}_v})\right) = \tilde{\delta}(\mathcal{E}^K_{ui} \setminus \mathcal{E}^{in}_v).$$

**Proof.** $\Delta\mathcal{E}^{K+1}_{\hat{u}_v} = \mathcal{E}^{K+1}_{\hat{u}_v} \setminus \mathcal{E}^K_{\hat{u}_v}$. $\Delta\mathcal{E}^{K+1}_{\hat{u}_v} \subset \mathcal{E}^K_{ui}$ indicates no $(K+1)$-hop neighbours of $\hat{u}_v$ is newly included into $G^K_{ui}$, that is to say $\Delta\mathcal{E}^{K+1}_{ui} \cap \Delta\mathcal{E}^{K+1}_{\hat{u}_v} = \emptyset$.

These two lemmas are used to demonstrate our expected $K$-hop Influence function of a given diffusion forest doesn't have monotonicity or submodularity. Lemma 1 presents an inequality stating that after inserting the edge $(\hat{u}_v, v)$, and

before updating $G_{ui}^K$, the expected $K$-hop influence spread function is equal to or smaller than the expected $K$-hop influence spread function before the insertion of $(\hat{u}_v, v)$. As $\hat{u}_v$ is just the constrianed optimal neighbor of node $v$, while the expected influence spread function assumes the maximum number of nodes are activated by their optimal neighbors. Lemma 2 presents a strong condition about the simplification of changes in the expected K-hop Influence function. The activation of node $v$ by node $\hat{u}_v$ induces the update of the K-hop uninfluenced network $G_{ui}^K$. The update is represented as $(\mathcal{E}_{ui}^K \setminus \mathcal{E}_v^{in}) \cup (\Delta\mathcal{E}_{ui}^{K+1} \cap \Delta\mathcal{E}_{\hat{u}_v}^{K+1})$. Here, $\mathcal{E}_{ui}^K \setminus \mathcal{E}_v^{in}$ denotes the removal of all input edges to node $v$. $\Delta\mathcal{E}_{ui}^{K+1} \cap \Delta\mathcal{E}_{\hat{u}_v}^{K+1}$ indicates the addition of new edges that connect the newly reachable nodes. These nodes are at the intersection of the nodes $(K+1)$-hop away from the diffusion forest $F_\ell$ before the activation of node $v$ and the nodes $(K+1)$-hop away from node $\hat{u}_v$. The process of adding and removing edges complicates the analysis of changes in the expected K-hop influence. The condition specified in Lemma 2 eliminates the addition of new edges, simplifying the analysis by focusing solely on the effects of removing existing connections.

**Theorem 2.** $\tilde{\delta}_{F_l}^K$ is not monotonous.

**Proof.** Under the condition $\Delta\mathcal{E}_{\hat{u}_v}^{K+1} \subset \mathcal{E}_{ui}^K$, $\tilde{\delta}^K(F_l \cup (\hat{u}_v, v)) - \tilde{\delta}^K(F_l) = \delta(\hat{u}_v, v) + \tilde{\delta}(\mathcal{E}_{ui}^K \setminus \mathcal{E}_v^{in}) - \tilde{\delta}(\mathcal{E}_{ui}^K) \leq 0$.

**Theorem 3.** $\tilde{\delta}_{F_l}^K$ is not sub-modular.

**Proof.** Under the condition $(\Delta\mathcal{E}_{\hat{u}_v}^{K+1} \cup \Delta\mathcal{E}_{\hat{u}_w}^{K+1}) \subset \mathcal{E}_{ui}^K$, $\tilde{\delta}^K(F_l \cup (\hat{u}_w, w)) - \tilde{\delta}^K(F_l) - \tilde{\delta}^K(F_l \cup (\hat{u}_v, v) \cup (\hat{u}_w, w)) + \tilde{\delta}^K(F_l \cup (\hat{u}_v, v)) =\leq 0$.

These two theorems reveal the properties of the objective function we aim to maximize. In a word, this function lacks both monotonicity and submodularity. This indicates that traditional greedy-based methods are not guaranteed to be effective for addressing our problem. Consequently, we propose a heuristic method to pursue an effective solution.

*2) Markov Decision Process Formulation:* Traditional influence maximization approaches usually adopt the greedy framework. However, the FEIM problem can't be well solved if we follow the greedy framework. The FEIM problem involves a more complex decision-making process which the greedy framework can't well handle. The greedy framework excels in scenarios where sequential decisions maximize or minimize a specific objective function. In traditional IM, the objective function is the influence spread, and the node set or edge set maximizing the influence spread is selected. However, our concern extends to minimizing the number of messages during the influence spread. Consequently, determining the optimal edge set becomes challenging because it's not reliable to identify which edge is helpful to minimize the total message consumption based on the marginal influence.

Example 3. As depicted in Figure 1 (c), two candidate edges $(a, c)$ and $(d, f)$ have the same marginal influence. But prioritizing the selection of $(a, c)$ over $(d, f)$ can save at most 2 messages compared with selecting these two edges at the same diffusion step.

We opt to propose a deep reinforcement learning (DRL) method to solve our problem. DRL supports decision-making by considering both the immediate rewards and the expected future rewards which aligns better with the trade-off between our dual objectives. Apart from that, DRL is also well-suited to the characteristics of our environment. Firstly, the edge weights in our graph are partially observable and DRL has demonstrated its effectiveness in training agents with incomplete environmental perceptions to conduct specific tasks. Secondly, our graph is dynamic due to the detection of new sub-events. DRL is designed to continually update the action policy based on observed rewards and states, allowing agents to quickly adapt to such dynamics.

We start by formulating the diffusion process as MDP.

- State: At each time step $t$, the environment has a state $s_t$, which contains the reachable range $rr$ and the nodes' status matrix $X_t$, denoted as $s_t = \{rr, X_t\}$. $X_t$ lists every node's status at time step t.
- Observation: At each time step $t$, the observation of the environment $o_t$ is an embedding matrix of candidate edges, where the first dimension corresponds to each candidate edge and the second dimension represents a combination of the edge's impact on influence and fairness.
- Action: At each time step $t$, according to $X_t$, the candidate edge set $\mathcal{E}_t^c$ is retrieved from $rr$. The agent takes an action $a_t$ which is defined as a one-hot vector indicating which edge is selected.
- Transition: Every node $v$ has an initial status $x_v = 0$, indicating the node hasn't received any messages. $x_v$ transitions to 1 if $v$ is selected as a seed or activated before reaching the message limit. Conversely, $x_v$ transitions to -1 if $v$ reaches the message limit but hasn't been activated.

*3) Edge Embedding:* In order to balance influence and fairness, the edge embedding includes two categories of features:

- **Expected user-engagement** is the sum of propagation probabilities of an edge. It indicates whether the edge's target node receives messages supportive to stimulate its potential to engage with the event.
- **Neighborhood-based influence** is an upper-bound estimation of an edge's marginal influence on the target node's neighborhood. Formally, let $\mathcal{N}^K(v)$ denote the $K$-hop out-neighborhood set of edge $(u, v)$. Given a query sub-event sequence $\mathbf{E}$ and a diffusion sub-network $G_\ell$, the marginal influence of candidate edge $(u, v)$ is estimated as

$$\triangle\delta^K((u, v) \mid G_\ell, \mathbf{E}) = \sum_{w \in \mathcal{N}^K(v)} P_{\mathbf{E},v}(w).$$

*4) Reward Formulation:* The reward of a diffusion sub-network is derived from the results of influence spread. We use the influence ratio ($IR$) and message precision ($Prec$) to assess the effectiveness of the influence spread, while the disparity in message precision ($|\Delta Prec|$) is used to evaluate fairness. In this paper, we do not explore the optimal way to

**Algorithm 1** Hierarchical Policy Update

Initialize training epoch counter $T = 0$.
Initialize training step counter $t \leftarrow 1$.
Initialize GFN with random weights $\theta$
**repeat**
  Reset gradients: $d\pi \leftarrow 0$ and $d\pi^M \leftarrow 0$
  $t_{\text{start}} = t$
  Get state $s_t$
  **repeat**
    Perform $a_t$ according to policy $\pi(a_t \mid s_t; \theta)$
    Receive reward $r_t$ and new state $s_{t+1}$
    $t \leftarrow t + 1$
  **until** terminal $s_t$
  $R = 0$
  $T \leftarrow T + 1$
  **for** $i = t - 1, \ldots, t_{\text{start}}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients w.r.t $\pi$: $d\pi \leftarrow d\pi + \nabla_\theta \log \pi(a_i \mid s_i; \theta)(R - V^W(s_i; \theta))$
    Accumulate gradients w.r.t $\pi^M$: $d\pi^M \leftarrow d\pi^M + \nabla_\theta \log \pi^M(s_i)(R - V^M(s_i; \theta))$
  Perform update of $\theta$ using $d\theta = d\pi + d\pi^M$
**until** $T > T_{\text{max}}$

combine these three reward signals.

$$R_{episode} = IR_{epiosde} + Prec_{episode} + \max_{\varepsilon_1, \varepsilon_2 \in \mathcal{E}_{G_\ell}} \left| B_\ell^{\mathbf{E}}(\varepsilon_1) - B_\ell^{\mathbf{E}}(\varepsilon_2) \right|$$

*5) Multi-objective Optimization:*

*6) Policy Update:* We train the Manager to predict the marginal influence of a candidate edge, which is one of three embeddings exploited to select an edge. This indicates that the Manager plays a partial role in decision-making. Given this partial involvement, it is challenging to effectively train the Manager solely on the gradient signals derived from the edge selection task. We introduce two loss functions aligning with the nature of node assignment task to train the Manager. The first one is the node assignment loss function from the perspective of making it easier for two connected nodes to be mapped to the same cluster, which is formulated as

$$L_n = \left\| Adj_t, S_t {S_t}^T \right\|_F, \tag{3}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. $Adj_t$ is the adjacency matrix of the uninfluenced network at time step t, and each element in $S_t {S_t}^T$ represents the probability that two nodes are divided to the same cluster.

The second one is the cluster affiliation loss function from the perspective of making it easier to clearly define the affiliation of each cluster, which is formulated as

$$L_c = \frac{1}{n} \sum_{i=1}^{n} H\left(S_i\right), \tag{4}$$

where $H(\cdot)$ is the entropy function that can reduce the uncertainty of the mapping distribution. The optimal situation is that the $i$-th node is only mapped to one cluster, and the entropy $H\left(S_{i:}\right)$ is 0 at this time. Based on these two loss functions, we formalise the update rule for the Manager as:

$$\nabla z_t = A_t^M \nabla_\theta \left( L_M(\theta) + L_A(\theta) \right) \tag{5}$$

where $A_t^M = R_t - V_t^M(o_t, \theta)$ is the Manager's advantage function, computed using a value function estimate $V_t^M(o_t, \theta)$ from the internal critic.

The Worker is trained to maximise the sum of environmental rewards $R_t$. The Workers policy $\pi$ is trained by an advantage actor critic formulated as:

$$\nabla \pi_t = A_t^W \nabla_\theta \log \pi\left(a_t \mid o_t; \theta\right) \tag{6}$$

where the advantage function estimator $A_t^W = R_t - V_t^W(o_t; \theta)$ is calculated using an internal critic, which estimates the value function for rewards.

## IV. EXPERIMENT

This section evaluates the effectiveness and robustness of our methods.

### A. Experimental Setting

*1) Dataset:* We conduct the experiments on data collected from Twitter during two natural disasters, Nepal Earthquake 2015 and Texas Flood 2015. The Nepal Earthquake 2015 data were collected from 15 April to 24 May 2015 and the Texas Flood 2015 data were collected from 12 May to 5 June 2015. We use the subset of Nepal Earthquake data in 15-24 April and those for Texas Flood in 12-21 May to construct the social graph. The social graph from Nepal Earthquake dataset is consisting of 510776 edges and 527153 nodes and that from Texas Flood dataset is consisting of 579515 edges and 550532 nodes. We use the Nepal earthquake subset after 25 April and the Texas flood subset after 22 May to detect sub-events. We decide whether a sub-event is relevant to the investigated event by adopting the labels from [14]. In total, we detect 1059 sub-events for the Nepal Earthquake and 1071 sub-events for the Texas Flood. For each sub-event, we search an interest region to spread the influence. These detected sub-events and searched sub-graphs are arranged in chronological order and we divide them into training set $(70\%)$ and test set $(30\%)$. The $50\%$ of test set is used for validation. For both training and test, we spread all the sub-events included in that stage for each interest region. We identify the matching sub-event of a node as the sub-event with the highest cosine similarity to one of the node's linked edges. Then, for each node, there is at least an edge consuming 1 message to activate the node.

*2) Baselines:* We compare three baselines for random k seeds and top-k seeds. All the baselines are implemented under our fairness constraint.

- Random based method where an edge is randomly selected for a candidate with multiple active neighbors. Its performance is reported as the average over five independent runs.
- Greedy based method where the edge which brings the maximum marginal gain to the sum of the influence increment of all seed nodes is selected iteratively. The