Steering Top-k Influencers in Dynamic Graphs via Local Updates

Vijaya Krishna Yalavarthi* †, Arijit Khan*
*Nanyang Technological University, Singapore
†Information Systems and Machine Learning Lab, University of Hildesheim, Germany

***Abstract*—We propose a *generalized* framework for influence maximization in large-scale, time evolving networks. Many real-life influence graphs such as social networks, telephone networks, and IP traffic data exhibit dynamic characteristics, e.g., the underlying structure and communication patterns evolve with time. Correspondingly, we develop a dynamic framework for the influence maximization problem, where we perform effective *local updates* to quickly adjust the top-k influencers, as the structure and communication patterns in the network change. We design a novel** N-Family **method (N=1, 2, 3, ...) based on the maximum influence arborescence (MIA) propagation model with approximation guarantee of $(1 - 1/e)$. We then develop heuristic algorithms by extending the** N-Family **approach to other information propagation models (e.g., independent cascade) and influence maximization algorithms (e.g., CELF, reverse reachable sketch). Based on a detailed empirical analysis over several real-world, dynamic, and large-scale networks, we find that our proposed solution,** N-Family **improves the updating time of the top-k influencers by $1 \sim 2$ orders of magnitude, compared to existing algorithms, while ensuring similar memory usage and influence spreads.**

I. Introduction

The problem of influence analysis [6] has been widely studied in the context of social networks, because of the tremendous number of applications of this problem in viral marketing and recommendations. The assumption in bulk of the literature on this problem is that a static network has already been provided, and the objective is to identify the top-k seed users in the network such that the expected number of influenced users, starting from those seed users and following an influence diffusion model, is maximized.

In recent years, however, people recognized the inherent usefulness in studying the dynamic network setting [2], and influence analysis is no exception to this general trend [10], [9], because many real-world social networks evolve over time. In an evolving graph, new edges (interactions) and nodes (users) are continuously added, while old edges and nodes get dormant, or deleted. In addition, the communication pattern and frequency may also change.

From an influence analysis perspective, even modest changes in the underlying network structure (e.g., addition/ deletion of nodes and edges) and communication patterns (e.g., update in influence probabilities over time) may lead to changes in the top-k influential nodes. As an example, let us consider the influence graph in Figure 1 with 12 nodes, out of which the top-2 seed nodes are A and I (marked in bold),

This work was done when the first author was employed at NTU

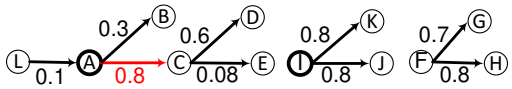


Fig. 1: Running example: an influence graph

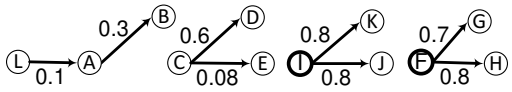


Fig. 2: Influence graph after update operation: edge delete AC

following the Maximum Influence Arborescence (MIA) model and $\theta = 0.07$ [5]. The influence spread obtained from this seed set, according to the MIA model, is: $2.58 + 2.6 = 5.18$. Now, assume an update operation in the form of an edge removal AC (marked in red). The new influence spread obtained from old seed nodes would be: 3.9, whereas if we recompute the top-2 seed nodes, they are I and F, as shown in Figure 2. The influence spread from these new seed nodes is: 5.1. Notice that there is a significant difference in the influence spread obtained with the old seeds vs. the new ones (even for such a small example graph), which motivates us to efficiently update the seed nodes when the influence graph evolves.

However, computing the seed set from ground, after every update, is prohibitively expensive [10], [9] — this inspires us to develop dynamic influence maximization algorithms. By carefully observing, we realize that among the initial two seed nodes, only one seed node, namely A is replaced by F, whereas I still continues to be a seed node. It is because A is in the *affected* region of the update operation, whereas I is not affected by it. Therefore, if we can identify that A can no longer continue as a seed node, then we can remove it from the seed set; and next, the aim would be to find one new seed node instead of two. Hence, we save almost $1/2$ of the computation in updating the seed set. To this end, the two following questions are critical for identifying the top-k seed nodes in a dynamic environment.

- What regions are affected when the graph evolves?
- How to efficiently update the seed nodes with respect to such affected regions?

Affected region. The foremost query that we address is identifying the affected region, i.e., the set of nodes potentially affected due to the update. They could be: (1) the nodes (including some old seed nodes) whose influence spreads are *significantly* changed due to the update operation, and also (2) those nodes whose *marginal gains* might change due to an affected seed node, discovered in the previous step(s). Given a seed set S, the marginal gain of a node $v \notin S$ is computed as

the additional influence that v can introduce when it is added to the seed set.

Given the influence graph and dynamic updates, we design an iterative algorithm to quickly identify the nodes in the affected region. We call our method N-Family, $N = 1, 2, 3, \ldots$, (until a base condition is satisfied), which we shall discuss in Section IV.

Updating the seed nodes. Once the affected region is identified, updating the top-k seed set with respect to that affected region is also challenging. We develop an approximate algorithm under the MIA model of information diffusion, with *theoretical performance guarantee* of $1 - 1/e$.

Moreover, it should be understood that our primary aim is to maximize the influence spread as much as possible with the new seed nodes, instead of searching for the exact seed nodes (in fact, finding the exact seed nodes is $\mathbf{NP}$-hard [6]). Therefore, we also show how to design more efficient heuristic algorithms, by carefully tuning the parameters (e.g., by limiting $N = 2$) of our N-Family approach.

Our proposed framework to update the top-k seed nodes is a *generic* one, and we develop heuristics (presented in the extended version [14]) by using it on top of other information propagation models (e.g., independent cascade [6]) and several influence maximization (IM) algorithms (e.g., Greedy [6], CELF [7], RR-sketch [3]). In particular, we first find the affected region, and then update the seed nodes *only by adding a few sub-routines to the existing static IM algorithms*, so that they can easily adapt to dynamic changes.

Our contributions.

- We propose an iterative technique, N-Family that systematically identifies affected nodes (including old seed nodes) due to dynamic updates, and develop an incremental method that replaces the affected seeds with new ones, so to maximize the influence spread in the updated graph. We derive theoretical performance guarantees of our algorithm under MIA model.
- We show how to develop efficient heuristics by extending proposed algorithm to other information propagation models and influence maximization algorithms for updating the seed nodes in an evolving network.
- We conduct a thorough experimental evaluation using several real-world, dynamic, and large graph datasets. The empirical results with our heuristics attest $1 \sim 2$ orders of efficiency improvement, compared to state-of-the-art approaches [10], [9].

II. Related Work

Kempe et al. [6] addressed the problem of influence maximization in a social network as a discrete optimization problem, and proposed a hill climbing greedy algorithm, with an accuracy guarantee of $(1-1/e)$. They used the MC simulation to compute the expected influence spread of a seed set. Since the introduction of the influence maximization problem, many algorithms [7], [5], [3] have been developed, both heuristic and approximated, to improve the efficiency of the original greedy method (see [4] for details).

In recent years, there has been interest in performing influence analysis in dynamic graphs [1], [10], [8], [13], [9], [12]. The work in [1] was the first to propose methods that maximize the influence over a specific time interval; however, it was not designed for online setting. The work in [13] probed a subset of the nodes for detecting the underlying changes. Liu et al. [8] considered an evolving network model (e.g., preferential attachment) for influence maximization. Subbian et al. [11] discussed the problem of finding influencers in social streams, although they employed frequent pattern mining techniques over the underlying social stream of content. This is a different modeling assumption than the dynamic graph setting considered in this work. Recently, Wang et al. [12] considered a *sliding window model* to find influencers based on the most recent interactions. Once again, their framework is philosophically different from the classical influence maximization setting [6], as they do not consider any edge probabilities; and hence, not directly comparable to ours.

In regards to problem formulation, recent works in [10], [9], [15] are closest to ours. UBI+ [10] was designed for MC-simulation based algorithms and IC model. It performs greedy exchange for multiple times — every time an old seed node is replaced with the best possible non-seed node. If one continues such exchanges until there is no improvement, the method guarantees 0.5 approximation. DIA [9] and [15] work on top of RR-Sketches. These methods generate all RR-sketches only once; and after every update, quickly modifies those existing sketches. After that, *DIA [9] identifies all seed nodes from ground using modified sketches*. This is the key difference with our framework, since *we generally need to identify only a limited number of new seed nodes*, based on affected regions due to updates. On the contrary, [15] reports the top-k nodes having maximum influence spreads individually with the modified sketches. Thus, the objective of [15] is different from that of classic influence maximization, which we study in this work.

Note that it is non-trivial to adapt UBI+ [10] and DIA [9] for other influence models and IM algorithms, than their respective ones. A drawback of this is as follows. Sketch based methods (e.g., DIA) consume higher memory for storing multiple sketches. In contrast, MC-simulation based methods (e.g., UBI+) are slower over large graphs. On the other hand, our proposed N-Family approach can be employed over many IM models and algorithms, and *due to the local updating principle, it significantly improves the efficiency under all scenarios*. Therefore, one can select the underlying IM models and algorithms for the N-Family approach based on system specifications and application requirements. This demonstrates the *generality* of our solution.

III. Preliminaries

An influence network can be modeled as an uncertain graph $\mathcal{G}(V, E, P)$, where V and $E \subseteq V \times V$ denote the sets of nodes (users) and directed edges (links between users) in the network, respectively. P is a function $P : E \rightarrow (0, 1)$ that assigns a probability to every edge $uv \in E$, such that P_{uv} is the strength at which an active user $u \in V$ influences

her neighbor $v \in V$. The edge probabilities can be learnt (from past propagation traces), or inferred (following various models), as discussed in [4]. In this work, we shall assume that $\mathcal{G}(V, E, P)$ is given as an input to our problem.

A. Influence Maximization in Static Graphs

Whenever a social network user (node) u buys a product, or endorses an action (e.g., re-tweets a post), she is viewed as being influenced or activated. When u is active, she automatically becomes eligible to influence her neighbors who are not active yet. All active users (nodes) at the end are considered as the users (nodes) influenced by S.

In this work we consider two different models for the influence propagation: maximum influence arborescence (MIA) [5] and independent cascade (IC) [6]. Due to space limitation, details of the models are provided in our extended version [14]. The exact estimation of influence spread is a $\#\mathbf{P}$-hard problem under the IC model [5]. However, influence spread can be computed in polynomial time for the MIA model. Hence, we consider MIA model to develop an approximate algorithm with theoretical guarantee and IC model for efficient heuristics.

Influence maximization (IM) problem. Influence maximization is the problem of identifying the seed set S^* of cardinality k that has the maximum expected influence spread ($\sigma(S^*)$) in the network. The influence maximization is an $\mathbf{NP}$-hard problem, under both MIA and IC models [5], [6].

In spite of the aforementioned computational challenges of influence estimation and maximization, we can develop a *Greedy* Algorithm (Algorithm 1) with approximation guarantee of $(1 - \frac{1}{e})$ [6]. The Greedy algorithm repeatedly selects the node with the maximum marginal influence gain ($MG(S, u) = \sigma(S \cup \{u\}) - \sigma(S)$) (line 2), and adds it to the current seed set (line 3) until k nodes are identified.

B. IM in Dynamic Graphs

Classical influence maximization techniques are developed for static graphs. The real-time influence graphs are seldom static and evolves over time as follows.

Graph update categories. We recognize two update categories, *additive updates*: an edge or node is added, or probability of an edge increases and *reductive updates*: an edge or node is deleted, or probability of an edge decreases. Hereafter, we use a general term *update* for any of the above operations, and we denote an update operation with o.

Dynamic influence maximization problem.

Problem 1. *Given an initial uncertain graph $\mathcal{G}(V, E, P)$, old set S^*_{old} of top-k seed nodes, and a series of consecutive graph updates $\{o_1, o_2, \ldots, o_t\}$, find the new set S^*_{new} of top-k seed nodes for this updated graph.*

The baseline method to solve the dynamic influence maximization problem will be to find the updated graph at every time, and then execute an IM algorithm on the updated graph, which returns the new top-k seed nodes. However, *computing all seed nodes from ground at every snapshot is prohibitively expensive*, even for moderate size graphs [10], [9]. Hence, our work aims at *incrementally updating the seed set*, without explicitly running the complete IM algorithm at every snapshot of the evolving graph.

Algorithm 1 $Greedy(\mathcal{G}, S, k)$: for IM in static networks

Require: Graph $\mathcal{G}(V, E, P)$, seed set S (initially empty), positive integer k
Ensure: Seed set S having the top-k seed nodes
1: **while** $|S| <= k$ **do**
2: $u^* = \arg\max_{u \in V \setminus S}\{\sigma(S \cup \{u\}) - \sigma(S)\}$
3: $S = S \cup u^*$
4: Output S

IV. Proposed Solution

We propose a novel N-Family framework for dynamic influence maximization, which can be adapted to many influence maximization algorithms and several influence diffusion models. We first introduce our framework under the MIA model that illustrates how an update affects the nodes in the graph (Section IV-A), and how to re-adjust the top-k seed nodes with a theoretical performance guarantee (Section IV-B). Initially, we explain our technique for a single update, and later we show how it can be extended to batch updates (Section IV-C). In Section IV-D and in the extended version [14], we show how to extend our algorithm to IC and LT models, with efficient heuristics.

A. Finding Affected Regions

Given an update, the influence spread of several nodes in the graph could be affected. However, the nearby nodes would be impacted heavily, compared to a distant node. We, therefore, design a threshold (θ)-based approach to find affected regions, and this is consistent with the notion of the MIA model. Note that in MIA model, the influence spread happens only via the maximum influence paths (defined below), and an influence threshold θ is used to eliminate maximum influence paths with propagation probabilities less than θ. Clearly, θ is an input parameter to trade off between efficiency and accuracy, and its optimal value is decided empirically.

In MIA model, the affected nodes could be computed exactly in polynomial time. We, however, consider a more efficient upper bounding technique as discussed next.

1) Definitions: We start with a few definitions.

Definition 1 (Maximum influence paths)**.** *Maximum influence path (MIP) is a path Pt from a source u to a destination node v which has the highest probability compared to all other paths between the same pair of nodes.*

$$MIP(u, v) = \underset{Pt \in \mathcal{P}(u,v)}{\arg\max} \{ \prod_{e \in Pt} P_e \}$$

Definition 2 (Maximum Influence In (Out)-Arborescence)**.** *Maximum Influence In (Out)-Arborescence [5] of a node $u \in V$ is the union of all the maximum influence paths to (from) u, where every node in that path reaches (by) u with a minimum propagation probability θ, and is denoted as $MIIA(u, \theta)$ ($MIOA(u, \theta)$).*

$$MIIA(u, \theta) = \bigcup_{v \in V} \{MIP(v, u) : \prod_{e \in MIP(v,u)} P_e \geq \theta\}$$

$$MIOA(u,\theta) = \bigcup_{w \in V} \{MIP(u,w) : \prod_{e \in MIP(u,w)} P_e \geq \theta\}$$

Definition 3 (1-Family)**.** *For every node $u \in V$, 1-Family of u, denoted as $F_1(u)$, is the set of nodes that influence u, or get influenced by u with minimum probability θ through the maximum influence paths, i.e.,*

$$F_1(u) = MIIA(u,\theta) \cup MIOA(u,\theta)$$

Definition 4 (2-Family)**.** *For every node $u \in V$, 2-Family of u, denoted as $F_2(u)$, is the union of the set of nodes present in 1-Family of every node in $F_1(u)$, i.e.,*

$$F_2(u) = \bigcup_{w \in F_1(u)} F_1(w)$$

Note that 2-Family is a superset of 1-Family of a node.

Example 1. *In Figure 1, let us consider $\theta = 0.07$. Then, $pp(\{C\}, C) = 1, pp(\{A\}, C) = 0.8$, and $pp(\{L\}, C) = 0.8 \times 0.1 = 0.08$. For any other node in the graph, its influence on C is 0. Hence, $MIIA(C, 0.07) = \{C, A, L\}$. Similarly, $MIOA(C, 0.07) = \{C, D, E\}$. $F_1(C)$ will contain $\{C, A, L, D, E\}$. Analogously $F_2(C)$ will contain $\{C, A, L, D, E, B\}$. Since the context is clear, for brevity we omit θ from the notation of family.*

We note that Dijkstra's shortest path algorithm, with time complexity $\mathcal{O}(|E| + |V| \log |V|)$ [5], can be used to identify the $MIIA$, $MIOA$, and 1-Family of a node. The time complexity for computing 2-Family is $\mathcal{O}(|E| + |V| \log |V|)^2$. For simplicity, we refer to 1-Family of a node as its family.

The 2-Family of a seed node satisfies an interesting property (given in Lemma 1) in terms of marginal gains.

Lemma 1. *Consider $s \in S$, then removing s from the seed set S does not change the marginal gain of any node that is not in $F_2(s)$. Formally, $MG(S, u) = MG(S \setminus \{s\}, u)$, for all $u \in V \setminus F_2(s)$, according to the MIA model.*

Formal proofs of all our lemma and theorems are given in the extended version [14]. Intuitively, Lemma 1 holds because the marginal gain of a node u depends on the influence of seed nodes over those nodes that u influences. For a node u that is outside $F_2(s)$, there is no node that can be influenced by both s and u. It follows from the fact that a node influences, or gets influenced by the nodes that are present only in its family, based on the MIA model.

Change in family after an update. During an additive update, the size of the family of a node nearby the update may increase. Analogously, during a reductive update, the size of family of a node surrounding the update may decrease. Thus, for soundness, in case of an additive update, we compute $MIIA$, $MIOA$, and family on the updated graph. On the contrary, for a reductive update, we compute them on the old graph. Next, we show in Lemma 2 that $MIIA(u, \theta)$ provides a safe bound on affected region for any update originating at node u, according to the MIA model.

Lemma 2. *In an influence graph $\mathcal{G}(V, E, P)$, adding a new edge uv does not change the influence spread of any node outside $MIIA(u, \theta)$ by more than θ, following MIA model.*

Lemma 2 holds because a node u cannot be influenced by any node that is not in $MIIA(u, \theta)$, according to the MIA model. Hence, adding an edge uv does not change the influence spread (at all) of any node outside $MIIA(u, \theta)$. This phenomenon can be extended to edge deletion, edge probability increase, and for edge probability decrease. Moreover, for a node update (both addition and deletion) u, $MIIA(u, \theta)$ gives a safe upper bound of the affected region. We omit the proof due to brevity. Therefore, $MIIA(u, \theta)$ is an efficient (computing time $\mathcal{O}(|E|+|V| \log |V|)$) and a safe upper bound for the affected region.

2) Infected Regions: Due to an update in the graph, we find that a node may get affected in two ways: (1) the nodes (including a few old seed nodes) whose influence spreads are significantly affected due to the update operation, and also (2) those nodes whose marginal gains might change due to an affected seed node, discovered in the previous step(s). This gives rise to a recursive definition, and multiple levels of *infected* regions, as introduced next.

First infected region (1-IR). When an update o takes place, the influence spread of the nodes surrounding it, will change. Hence, we consider the first infected region as the set of nodes, whose influence spreads change at least by θ.

Definition 5 (First infected region (1-IR))**.** *In an influence graph $\mathcal{G}(V, E, P)$, a given probability threshold θ, and an update operation o, 1-IR(o) is the set of nodes whose influence spread changes at least by θ. Formally,*

$$\textit{1-IR}(o) = \{v \in V : |\sigma_{\mathcal{G}}(v) - \sigma_{\mathcal{G},o}(v)| \geq \theta\}$$

In the above equation, $\sigma_{\mathcal{G}}(v)$ denotes the expected influence spread of v in $\mathcal{G}$, whereas $\sigma_{\mathcal{G},o}(v)$ is the expected influence spread of v in the updated graph. Following our discussion, we consider $MIIA(u, \theta)$ as a proxy for 1-IR(o), where u is the starting node for the update operation o.

Second infected region (2-IR). We next demonstrate how infection propagates from the first infected region to other parts of the graph through the *family* of affected seed nodes.

First, consider a seed node $s \in S$, a non-seed node $u \notin S$, and $s \in F_2(u)$. If the influence spread of u has increased due to an update, then to ensure that s continues as a seed node, we have to remove s from the seed set, and recompute the marginal gain of every node in $F_2(s)$. The node, which has the maximum gain, will be the new seed node. Second, if a seed node s gets removed from the seed set in this process, the marginal gains of all nodes present in $F_2(s)$ will change. We are now ready to define the second infected region.

Definition 6 (Second infected region (2-IR))**.** *For an additive update (o_a), the influence spread of every node present in 1-IR(o_a) increases which gives the possibility for any node in 1-IR to become a seed node. Hence, the union of 2-Family of all the nodes present in 1-IR(o_a) is called the second infected region 2-IR(o_a). On the contrary, in a reductive update operation o_r, there is no increase in influence spread of any node in 1-IR(o_r). Hence, the union of 2-Family of old seed nodes present in 1-IR(o_r) is considered as the second infected region 2-IR(o_r).*

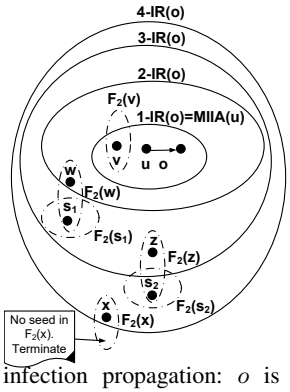


Fig. 3: Iterative infection propagation: o is an additive update operation originating at node u. s_1 and s_2 are two old seed nodes. v, w, z, x are nodes, not necessarily old seed nodes.

$$\textit{2-IR}(o_r) = \{F_2(s) : s \in \textit{ 1-IR}(o_r) \cap S\}$$
$$\textit{2-IR}(o_a) = \{F_2(u) : u \in \textit{ 1-IR}(o_a)\}$$

The time complexity to identify 2-IR is $\mathcal{O}(m(|E| + |V|\log|V|)^2)$, where m is the number of nodes in 1-IR.

Example 2. *In Figure 1, consider the removal of edge AC. Assuming $\theta = 0.07$, 1-IR(o)=$MIIA(A, 0.07) = \{A, L\}$. Now, 2-IR(o)=$F_2(A)$ because A is an old seed node present in 1-IR(o) for this reductive update. Furthermore, because this is a reductive update, the family of A needs to be computed before the update. Therefore, 2-IR(o)=$F_2(A) = \{A, B, C, D, L, E\}$.*

Iterative infection propagation. Whenever there is an update, the infection propagates through the 2-Family of the nodes whose marginal gain changes as discussed above. For $N \geq 3$, the infection propagates from the $(N-1)^{th}$ infected region to the N^{th} infected region through old seed nodes that are present in the 2-Family of nodes in (N-1)-IR.

Definition 7 ($N(\geq 3)$ infected region (N-IR))**.** *The 2-Family of seed nodes, that are in the 2-Family of infected nodes in (N-1)-IR, constitute the N^{th} infected region.*

$$\textit{N-IR} = \{F_2(s) : s \in F_2(u) \cap S, u \in \textit{ (N-1)-IR}\}$$

We demonstrate the iterative computation of infected regions, up to 4-IR for an additive update, in Figure 3. We begin with node u which is the starting node of the update, and $MIIA(u, \theta)$ is the 1-IR. The update being an additive one, union of 2-Family of all the nodes $v \in$ 1-IR is considered as the 2-IR. For all nodes $w \in$ 2-IR, we compute $F_2(w)$. Now, union of 2-Family of all seed nodes $s_1 \in F_2(w)$ is considered as 3-IR. Similarly, 4-IR can be deduced, and as there is no seed present in the 2-Family of all nodes $x \in$ 4-IR, we terminate the infection propagation.

Termination of infection propagation. The infection propagation stops when no further old seed node is identified in the 2-Family of any node in the N^{th} infected region. Due to this, there shall be no infected node present in 2-Family of any uninfected seed node. For a seed set of cardinality k, it can be verified that the maximum value of N can be between 1 and $(k+1)$ for reductive update and between 2 and $(k+2)$ for additive update.

Total infected region (TIR). The union of all infected regions is referred to as the total infected region (TIR).

Algorithm 2 N-Family seeds updating on top of Greedy

Require: Graph $\mathcal{G}(V, E, P)$, total infected region TIR, old seed set S, $|S| = k$, old priority queue Q
Ensure: Compute the new seed set S_{new} of size k

```
S_rem ← S \ TIR
for all u ∈ TIR do
    Q(u) ← σ(u)
while TRUE do
    S_new = Greedy(G, S_rem, k)   /* add S_rem − k seeds */
    S_order ← Sorted nodes of S_new in Greedy inclusion order
    w ← Q[top]
    if (S_order^{k-1}, s_o^k) < MG(S_order, w) then
        for all u ∈ F2(s_o^k) \ S_order^{k-1} do
            Q(u) ← MG(S_order^{k-1}, u)
            S_rem ← S_order^{k-1}
    else
        Output S_order
```

$TIR =$ 1-IR $\cup$ 2-IR $\cup$ 3-IR $\cup \ldots$ until termination

Our recursive definition of TIR satisfies the following.

Lemma 3. *The marginal gain of every node outside TIR does not change, according to the MIA model. Formally, let S be the old seed set, and we denote by S_{rem} the remaining old seed nodes outside TIR, i.e., $S_{rem} = S \setminus$ TIR. Then, the following holds: $MG(S, v) = MG(S_{rem}, v)$, $\forall v \in V \setminus$ TIR.*

Lemma 3 holds because any node outside TIR does not belong to 2-Family of any seed node present in TIR. Hence, by Lemma 1, its marginal gain does not change.

Lemma 4. *Any old seed node outside TIR has no influence on the nodes inside TIR, following the MIA model. Formally, $pp(S_{rem}, u) = 0$, $\forall u \in$ TIR.*

Lemma 4 holds because any uninfected seed node is more than 2-Family away from any node present in TIR (This is how we terminate infection propagation). Hence, there is no node present in TIR that belongs to the family of any seed node outside TIR.

The old seed nodes inside TIR may no longer continue as seeds, therefore we discard them from the seed set, and the same number of new seed nodes are identified. We discuss the updating procedure of seed nodes below.

B. Updating the Seed Nodes

1) Approximation Algorithm: We present our proposed algorithm for updating the seed set in Algorithm 2. Consider Greedy (Algorithm 1) over the MIA model on the initial graph, and assume that we obtained the seed set S, having cardinality k. Since Greedy works in an iterative manner, let us denote by S^{i-1} the seed set formed at the end of the $(i-1)$-th iteration, whereas $s_i \in S$ is the seed node added at the i-th iteration. Clearly, $1 \leq i \leq k$, $|S^{i-1}| = i - 1$, and $S = S^k = \cup_{i=1}^{k} s_i$. Additionally, we use a priority queue Q, where its top node w has the maximum marginal gain $MG(S, w)$ among all the non-seed nodes.

After update o, we compute the total infected region TIR. Consider S_{rem}, of size $|S_{rem}| = k'$, as the set of old seed nodes outside TIR, i.e., $S_{rem} = S \setminus$ TIR. Then, we remove $(k - k')$ old seed nodes inside TIR, and our objective is to identify $(k - k')$ new seeds from the updated graph.

Note that inside S_{rem}, seeds are still sorted in descending order of their marginal gains, computed at the time of insertion in the old seed set S following the Greedy algorithm.

After removing the old seed nodes present in TIR from the seed set, we compute the influence spread $\sigma(u)$ of every node $u \in$ TIR and, we update these nodes u in the priority queue Q, based on their new marginal gains $\sigma(u)$ (lines 1-4). It can be verified that $MG(S_{rem}, u) = \sigma(u)$, for all $u \in$ TIR, due to Lemma 4.

Now, we proceed with greedy algorithm and find the new $(k - k')$ seed nodes. Let us denote by S_{new} the new seed set (of size k) found in this manner (line 6). Next, we sort the seed nodes in S_{new} in their *appropriate* inclusion order according to the Greedy algorithm over the updated graph (line 7). This can be efficiently achieved by running Greedy *only* over the seed nodes in S_{new}, while computing their influence spreads and marginal gains in the updated graph. The sorted seed set is denoted by S_{order}. Let us denote by s_o^k the last (i.e., k-th) seed node in S_{order}, whereas S_{order}^{k-1} represents the set of top-$(k-1)$ seed nodes in S_{order}. We denote by w the top-most seed node in the priority queue Q. We terminate our updating algorithm (line 15), if $MG(S_{order}^{k-1}, s_o^k) \geq MG(S_{order}, w)$.

Iterative seed replacement. On the other hand, if $MG(S_{order}^{k-1}, s_o^k) < MG(S_{order}, w)$, we remove the last seed node s_o^k from S_{order}. For every node u in the $F_2(s_o^k) \setminus S_{order}^{k-1}$, we compute marginal gain $MG(S_{order}^{k-1}, u)$ and update the priority queue Q (lines 10-11). Next, we compute a new seed node using Greedy and add it to S_{order}^{k-1}, thereby updating the seed set S_{order}. We also keep the nodes in S_{order} sorted after every update in it. Now, we again verify the condition: if $MG(S_{order}^{k-1}, s_o^k) < MG(S_{order}, w)$, where w being the new top-most node in the priority queue Q, then we repeat the above steps, each time replacing the last seed node s_o^k from S_{order}, with the top-most node from the updated priority queue Q. This iterative seed replacement phase terminates when $MG(S_{order}^{k-1}, s_o^k) \geq MG(S_{order}, w)$. Clearly, this seed replacement can run for at most $|S_{rem}| = k'$ rounds; because in the worst scenario, all old seed nodes in S_{rem} could get replaced by new seed nodes from TIR. Finally, we report S_{order} as the new seed set.

2) Theoretical Performance Guarantee: We show in our extended version [14] that the top-k seed nodes reported by our N-Family method are the same as the top-k seed nodes obtained by running the Greedy on the updated graph under MIA model. Since, the Greedy algorithm provides the approximation guarantee of $1 - \frac{1}{e}$ under the MIA model [5], N-Family also provides the same approximation guarantee.

C. Extending to Batch Updates

We consider the difference of nodes and edges present in two snapshots at different time intervals of the evolving network as a set of batch updates. One straightforward approach would be to apply our algorithm for every update sequentially. However, we develop a more efficient technique as follows. For a batch update consisting of m individual updates, every update o_i has its own TIR$(o_i), i = 1, 2, 3, \ldots, m$. The TIR of the batch update is the union of TIR(o_i), for all $i \in (1, m)$.

$$\text{TIR} = \cup_{i=1}^{m} \text{ TIR}(o_i)$$

Once the TIR is computed corresponding to a batch update, we update the seed set using Algorithm 2. Processing all the updates in one batch is more efficient than the sequential updates. For example, if a seed node is affected multiple times within a batch, during sequential updates we have to check if it remains the seed node every time. Whereas in batch update, we need to verify it only once.

D. Heuristic Solution in IC model

Here we will show how we can develop efficient heuristics by extending the proposed N-Family approach to the IC model. Extension to LT model is given in our extended version [14].

Computing TIR. In the IC model, finding the nodes whose influence spread changes by at least θ (due to an update) is a $\#\mathbf{P}$-hard problem. Hence, computing TIR under IC model is hard as well, and one can *no longer ensure a theoretical performance guarantee* of $(1 - \frac{1}{e})$ as earlier. Instead, we *estimate* TIR analogous to MIA model (discussed in Section IV-A2), which generates high-quality results as verified in our experiments. This is because the maximum influence paths considered by MIA model play a crucial role in influence cascade over real-world networks [5].

We also propose a more efficient heuristic method, based on our experimental results with several evolving networks, by carefully tuning the parameters (e.g., by limiting $N = 1, 2$ in TIR computation) of our N-Family algorithm. Indeed, the major difference in influence spreads between the new seed set and the old one comes from those seed nodes in the first two infected regions (i.e., 1-IR and 2-IR) (We refer to sensitivity analysis experiments in our extended version [14]).

Updating Seed set. Our method In IC model follows the same outline as given in Algorithm 2 for updating the seed set with two major differences. In lines 3 and 11 of Algorithm 2, we compute the marginal gains and update the priority queue, but now we employ more efficient techniques based on the IM algorithm used for the purpose. Moreover, in the extended version [14], we derive two efficient heuristic algorithms, namely, Family-CELF (or, F-CELF) and Family-RRS (or F-RRS) by employing our N-Family approach on top of two efficient IM algorithms CELF [7] and RR sketch [3], respectively.

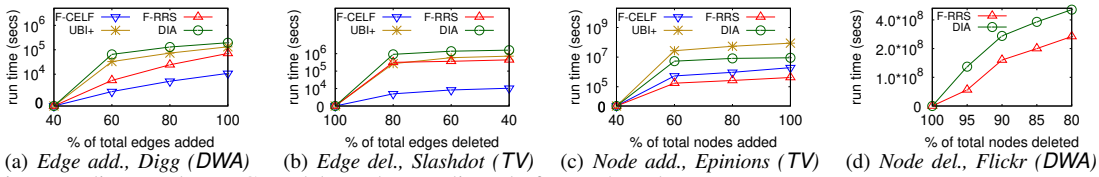
TABLE I: Properties of datasets

Dataset	#Nodes	#Edges	Timestamps	
			From	To
Digg	30 398	85 247	10-05-2002	11-23-2015
Slashdot	51,083	130 370	11-30-2005	08-15-2006
Epinions	131 828	840 799	01-09-2001	08-11-2003
Flickr	2 302 925	33 140 017	11-01-2006	05-07-2007

V. Experimental Results

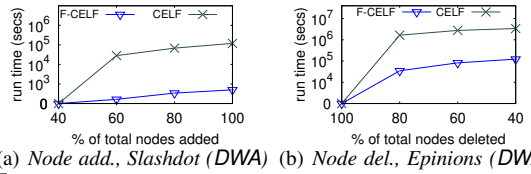
A. Experimental Setup

• **Datasets.** We download four real-world graphs (Table I) from the Koblenz Network Collection (http://konect. uni-



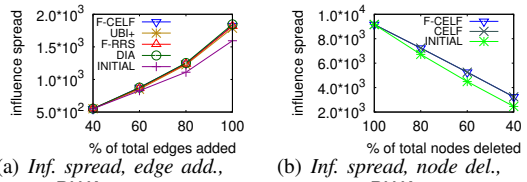
(a) *Edge add., Digg (DWA)* (b) *Edge del., Slashdot (TV)* (c) *Node add., Epinions (TV)* (d) *Node del., Flickr (DWA)*

Fig. 4: Run time to adjust seed set, IC model, seed sets adjusted after each update



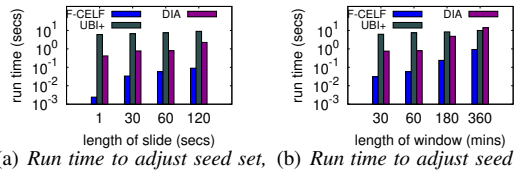
(a) *Node add., Slashdot (DWA)* (b) *Node del., Epinions (DWA)*

Fig. 5: Run time to adjust seed set, MIA model, seed sets adjusted after each update



(a) *Inf. spread, edge add., Digg (DWA) in IC model* (b) *Inf. spread, node del., Epinions (DWA) in MIA model*

Fig. 6: Influence spread, seed sets adjusted after each update



(a) *Run time to adjust seed set, varying L, W = 1 hour* (b) *Run time to adjust seed set, varying W, L = 60 secs*

Fig. 7: Impacts of varying batch sizes, sliding window model, *Twitter*, IC influence prop., seed sets are adjusted after every slide

koblenz.de/ networks/). All these graphs have directed edges, together with time-stamps; and hence, we consider them as evolving networks. If an edge appears for multiple times, we only consider the first appearance of that edge as its insertion time in the graph. The edge counts in Table I are given considering distinct edges only.

• **Influence strength models.** We adopt two popular edge probability models for our experiments. *Those are exactly the same models used by our competitors:* UBI+ [10] *and* DIA [9]. **(1) Degree Weighted Activation (DWA) Model.** In this model [6], [9], [10] (also known as weighted cascade model), the influence strength of the edge (uv) is equal to $1/d_{in}(v)$, where $d_{in}(v)$ is the in-degree of the target node v. **(2) Trivalency (TV) Model.** In this model [6], [9], each edge is assigned with a probability, chosen uniformly at random, from $(0.1, 0.01, 0.001)$.

• **Competing Algorithms. (1) FAMILY-CELF (F-CELF).** [14] This is an implementation of our proposed N-FAMILY framework, on top of the CELF influence maximization algorithm. **(2) FAMILY-RR-Sketch (F-RRS).** [14] This is an implementation of our proposed N-FAMILY framework, on top of the RR-Sketch influence maximization algorithm. **(3) DIA.** The DIA algorithm was proposed in [9], on top of the RR-Sketch. The method generates all RR-sketches only once; and after every update, quickly modifies those existing sketches. After that, *all seed nodes are identified from ground* using the modified sketches. This is the key difference with our algorithm F-RRS, since we generally need to identify only a limited number of new seed nodes, based on the affected region due to the update. **(4) UBI+.** UBI+ [10] performs greedy exchange for multiple times — every time an old seed node is replaced with the best possible non-seed node. Due to efficiency, [10] limits the number of exchanges to k, where k is the cardinality of the seed set. An upper bounding method is used to find such best possible non-seed nodes at every round.

• **Parameters Setup. (1) #Seed nodes.** We vary seed set size from 5∼100 (default 30 seed nodes). **(2) #RR-Sketches.** Our total number of sketches are bounded by $\beta(|V| + |E|)\log|V|$, and we vary β from 2∼512 (default $\beta = 2^5 = 32$ [9]). **(3) Size of family.** The family size $|F_1(u)|$ of a node u is decided by the parameter θ, and we vary θ from 1∼0.01 (default θ=0.1). **(4) #IR to compute TIR.** We consider upto *3-IR* to compute *TIR* (default upto *2-IR*). **(5) Influence diffusion models.** We employ IC [6] and MIA [5] models for influence cascade. Bulk of our empirical results are provided with IC, since this is widely-used in the literature. **(6) #MC samples.** We use 10 000 MC samples to compute the influence spread in IC [6].

Due to lack of space, sensitivity analysis results with respect to above parameters are given in the extended version [14]. The code is implemented in Python, and the experiments are performed on a single core of a 256GB, 2.40GHz Xeon server. All results are averaged over 10 runs.

B. Single Update Results

First, we show results for single update queries related to edge addition, edge deletion, node addition, and node deletion. We note that adding an edge uv can also be considered as an increase in the edge probability from 0 to $P_e(uv)$. Analogously, deleting an edge can be regarded as a decrease in edge probability. Moreover, for the DWA edge influence model, when an edge is added or deleted, the probabilities of multiple adjacent edges are updated (since, inversely proportional to node degree). **(1) Edge addition.** We start with initial 40% of the edges in the graph data, and then add all the remaining edges as dynamic updates. **(2) Edge deletion.** We delete the last 60% of edges from the graph as update operations. **(3) Node addition.** We start with the first 40% of nodes and all their edges in the dataset. We next added the remaining nodes sequentially, along with their associated edges. **(4) Node deletion.** We delete the last 20% of nodes, with all their edges from the graph.

Here, we adjust the seed set after every update, since one does not know apriori when the seed set actually changes, and hence, it can be learnt only after updating them.

Efficiency. In Figure 4, we present the running time to dynamically adjust the top-k seed nodes, under the IC influence

cascade model. We find that F-CELF and F-RRS are always faster than UBI+ and DIA, respectively, by $1\sim2$ orders of magnitude. As an example, for node addition over *Epinions* in Figure 4(c), the time taken by F-CELF is only 2×10^6 sec for about 80K node additions (i.e., 24.58 sec/node add). In comparison, UBI+ takes around 8×10^7 sec (i.e., 1111.21 sec/ node add). Our F-RRS algorithm requires about 4×10^5 secs (i.e., 5.31 sec/ node add), and DIA takes 10×10^6 sec (i.e., 134.68 sec/node add). *These results clearly demonstrate the efficiency improvements by our methods*.

Sketch-based methods are relatively slower (i.e., F-RRS vs. F-CELF, and DIA vs. UBI+) in smaller graphs (e.g., *Digg* and *Slashdot*). This is due to the overhead of updating sketches after graph updates. On the contrary, in our larger datasets, *Epinions* and *Flickr*, the benefit of sketches is more evident as opposed to MC-simulation based techniques. In fact, both F-CELF and UBI+ are very slow for our largest *Flickr* dataset; hence, we only show F-RRS and DIA for *Flickr* in Figure 4(d).

Additionally, in Figure 5, we show the efficiency of our method under the MIA model of influence spread. Since it is non-trivial to adapt UBI+ and DIA for the MIA model, we compare our algorithm F-CELF with CELF [7] in these experiments. For demonstration, we consider *Slashdot* and *Epinions*, together with node addition and deletion, respectively. It can be observed from Figure 5 that F-CELF is about 2 orders of magnitude faster than CELF. *These results illustrate the generality and effectiveness of our approach under difference influence cascading models*.

Influence spread. We show influence spread with updated seed set for IC (Figure 6(a)) and MIA (Figure 6(b)). Note that the competing algorithms, i.e., F-CELF, F-RRS, UBI+, and DIA achieve similar influence spreads with their updated seed nodes. We also show by INITIAL the influence spread obtained by the old seed set in the modified graph. We find that INITIAL achieves significantly less influence spread, especially with more graph updates. *These results demonstrate the usefulness of dynamic IM approaches, and also effectiveness of our algorithm for influence coverage*.

C. Batch Update Results

We demonstrate batch updates with a sliding window model as used in [10]. Initially we consider the edges present in between 0 to W units of time (length of window) and compute the seed set. Next, we slide the window to L units of time. The edges present in between L and $W + L$ are considered as the updated data, and our goal is to adjust the seed set based on the updated data. We delete the edges from 0 to L and add the edges from W to $W + L$. We continue sliding the window until we complete the whole data.

We conducted this experiment using the *Twitter* dataset downloaded from https://snap.stanford.edu/data/ (tweets posted between 01-JUL-2012 to 07-JUL-2012, during announcement of the Higgs-Boson particle), containing $304\,199$ nodes and $555\,481$ edges. Probability of an edge is given by: $1 - e^{\frac{-f}{k}}$, where f is the total number of edges appeared in the window, and k is the constant. We vary W from 30 mins to 6 hrs and L from 1 sec to 2 mins. We set the value of k as 5. On an average, 1.8 updates appear per second. Since the number of edges in a window is small, we avoid showing results with F-RRS. This is because F-CELF performs better on smaller datasets. From the experimental results in Figure 7, we find that *F-CELF is faster than both UBI+ and DIA upto three orders of magnitude*.

VI. Conclusions

We developed a generalized, local updating framework for efficiently adjusting the top-k influencers in an evolving network. Our method iteratively identifies *only* the affected seed nodes due to dynamic updates in the influence graph, and then replaces them with more suitable ones. Our solution can be applied to a variety of information propagation models and influence maximization techniques. Our algorithm, N-Family ensures $(1-\frac{1}{e})$ approximation guarantee with the MIA model, and works well for localized batch updates. Based on a detailed empirical analysis over several real-world and dynamic networks, N-Family improves the updating time of the top-k influencers by $1\sim2$ orders of magnitude, compared to existing algorithms, while ensuring similar influence spreads.

VII. Acknowledgements

The research is supported by MOE Tier-1 RG83/16 and NTU M4081678. First author would like to thank Information Systems and Machine Learning Lab at University of Hildesheim, Germany for partial funding of travel expenses.

References

[1] C. Aggarwal, S. Lin, and P. S. Yu. On Influential Node Discovery in Dynamic Social Networks. In *SDM*, 2012.

[2] C. Aggarwal and K. Subbian. Evolutionary Network Analysis: A Survey. *ACM Comput. Surv.*, 47(1):10:1–10:36, 2014.

[3] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *SODA*, 2014.

[4] W. Chen, L. V. S. Lakshmanan, and C. Castillo. Information and Influence Propagation in Social Networks. In *Synthesis Lec. on Data Management (Morgan & Claypool Publishers)*, 2013.

[5] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, 2010.

[6] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through a Social Network. In *KDD*, 2003.

[7] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective Outbreak Detection in Networks. In *KDD*, 2007.

[8] X. Liu, X. Liao, S. Li, J. Zhang, L. Shao, C. Huang, and L. Xiao. On the Shoulders of Giants: Incremental Influence Maximization in Evolving Social Networks. In *Complexity*, 2017.

[9] N. Ohsaka, T. Akiba, Y. Yoshida, and K.-I. Kawarabayashi. Dynamic Influence Analysis in Evolving Networks. In *PVLDB*, 9(12):1077–1088, 2016.

[10] G. Song, Y. Li, X. Chen, X. He, and J. Tang. Influential Node Tracking on Dynamic Social Network: An Interchange Greedy Approach. *IEEE Trans. Knowl. Data Eng.*, 29(2):359–372, 2017.

[11] K. Subbian, C. Aggarwal, and J. Srivastava. Mining Influencers Using Information Flows in Social Streams. *ACM Trans. Knowl. Discov. Data*, 10(3):26:1–26:28, 2016.

[12] Y. Wang, Q. Fan, Y. Li, and K.-L. Tan. Real-Time Influence Maximization on Dynamic Social Streams. In *PVLDB*, 10(7):805–816, 2017.

[13] H. Zhuang, Y. Sun, J. Tang, J. Zhang, and X. Sun. Influence Maximization in Dynamic Social Networks. In *ICDM*, 2013.

[14] V. K. Yalavarthi and A. Khan. Steering Top-k Influencers in Dynamic Graphs via Local Updates (Extended Version). In *http://arxiv.org/abs/1802.00574*, 2018.

[15] Y. Yang, Z. Wang, J. Pei, and E. Chen. Tracking Influential Nodes in Dynamic Networks. *IEEE Trans. Knowl. Data Eng.*, 29(11):2615–2628, 2017.