Array 15 (2022) 100226

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Maximization influence in dynamic social networks and graphs

Gkolfo I. Smani *, Vasileios Megalooikonomou

Department of Computer Engineering and Informatics, University of Patras, 26504, Greece

ARTICLE INFO

Keywords:
Maximization influence
Dynamic social networks
Diffusion
Social influence
Graphs

ABSTRACT

Social influence and influence diffusion have been extensively studied in social networks. However, most existing works on influence diffusion focus on static networks. In this paper, we study the problem of maximizing influence diffusion in dynamic social networks, i.e. networks that change over time. We propose the following algorithms under the Linear Threshold (LT) and Independent Cascade (IC) models: (a) the DM algorithm which is an extension of MATI algorithm and solves the Influence Maximization (IM) problem in dynamic networks, (b) the DM-C algorithm which is a latter version of DM and solves the IM problem using k-core decomposition and the core number information, (c) the DM-T algorithm which is another version of DM, that uses K-truss decomposition and the truss number information in order to solve the IM problem. Experimental results show that our proposed algorithms increase diffusion performance by 2 times compared with several state of the art algorithms and achieve comparable results in diffusion with the Greedy algorithm. Also, the proposed algorithms are 8.5 times faster in computational time compared with previous methods.

1. Introduction

Recently, social networks are playing a fundamental role in information propagation, since more and more people prefer to publicize their views or ideas on the networks. One of the main research interests is to understand the way of influence and information spread in social networks. For example, a company wants to market a new product through the effect of "word of mouth" in the social networks. It wishes to find and convince a small subset of users (seed users) to adopt the product so as to trigger a large cascade of further adoptions via social influence. Fundamentally, we need to understand the influence diffusion by answering questions such as: how to select the seed users so that the total number of triggered users who adopt the product can be maximized (a.k.a. influence maximization) [1,9,11,19,20,21].

A natural problem for social influence is how to find the initial users that will eventually influence the largest number of users, which is known as Influence Maximization (IM). Given a social network G and an integer k, IM's goal is to select k seed users in G in hope that adopting a promoted product or idea can maximize the expected number of final adopted users through word-of-mouth effect [1,9,20]. Initially proposed by Kempe et al. [8], the problem of IM has been intensively studied by a plethora of subsequent projects, improvements or modifications from multiple aspects, including estimation of influence size, adaptive seeding, boosting seeding, and many others.

The main task in IM lies in estimating the expected size of influenced users of each alternative seed set based on each user's activation probabilities, referring to the probability that a user successfully influences his social neighbors after being influenced. The influences among users are quantified by those activation probabilities [1,16,20,21]. While existing literature works well in finding the most influential seed users, they are all constrained to the assumption that the number of nodes in the network, along with their edges in between, are fixed during influence diffusion. Consequently, it violates real practices as many realistic social networks are usually growing over time.

In this paper, we study the problem of influence maximization (IM) on dynamic social networks (DSNs) which are changing over time, and specifically under the Linear Threshold (LT) and Independent Cascade (IC) models. According to both, at any discrete time step a user can be either active or inactive (for example, has adopted the product or not) and the information propagates until no more users can be activated.

The main contributions of this work can be summarized as follows:

- We introduce efficient IM algorithms for dynamically changing networks.
- The proposed algorithms take advantage of node metrics in order to stratify the nodes – a procedure that facilitates tracking changes.
- The proposed approaches are evaluated on large scale real-world graphs.

* Corresponding author.
E-mail addresses: gsmani@upatras.gr (G.I. Smani), vasilis@ceid.upatras.gr (V. Megalooikonomou).

https://doi.org/10.1016/j.array.2022.100226
Received 5 May 2022; Received in revised form 4 July 2022; Accepted 5 July 2022
Available online 13 July 2022

- The proposed algorithms perform better than several state of the art algorithms in terms of influence and computation time, while achieve comparable results to MC Greedy algorithm in terms of influence.

2. Related work

Influence Maximization aims at a given number of users that maximize the influence spread over a network. Previous efforts on Influence Maximization can be generally categorized into static methods and dynamic methods. In the case of static methods, there has been a vast amount of literature.

A Monte-Carlo simulation method is proposed by Kempe et al. [8], which estimates $\sigma(S)$ repeating Monte-Carlo simulation, where S is the set of seed nodes and $\sigma(S)$ is the average number of infected vertices. Chen et al. [3] propose PMIA (Prefix Excluding Maximum Influence Arborescence) to find seed vertices focusing on the paths with high information diffusion ratio. Chen et al. [4] also suggested Degree Discount based on node degree where the nodes, which are adjacent to the selected node, are given penalty. This means that, when node v is selected as a seed node and u is its neighbor, it is possible that v propagates information to u, so selecting nodes other than u as seed nodes is better for information diffusion. Cai et al. [2] proposed Holistic Influence Maximization problem as a complementary to Influence Maximization problem. Song et al. [18] proposed Bi-CLKT model that exploits contrastive learning to learn from large amounts of unlabelled data, transforming Knowledge Tracking problem into a graph form.

Two categories of methods have been proposed for the IM problem in dynamic networks: Monte-Carlo simulation-based methods and heuristic-based methods. The previous method is proposed by Habiba and Berger-Wolf [6]. The method estimates the scale of propagation σ(·) by repeating Monte-Carlo simulation in the case of static networks. Since σ(·) is a monotonic and submodular function also in dynamic networks, this method achieves large-scale propagation [6,13,14]. However, the computational cost of this method is high as in static networks [12]. Osawa et al. [14,16] proposed a heuristic method for calculating σ(·) at high speed. After $\sigma(S)$ is computed, seed nodes are obtained by greedy algorithm as in the method by Monte-Carlo simulation. Also, Murata and Koga [12] proposed three methods, Dynamic Degree Discount, Dynamic CI (Dynamic Collective Influence) and Dynamic RIS (Dynamic Reverse Influence Sampling), as extensions of static network methods to dynamic network methods, based on node degree, the degree of distant nodes, and the reachable nodes, respectively. Xue et al. [22] presented a new taxonomy that organizes current dynamic network embedding methods within a novel category hierarchy.

3. Preliminaries and problem statement

A social network is typically modeled as a directed graph $G = (V, E)$, consisting of $|V|$ users represented as nodes and $|E|$ directed edges reflecting the relationship between users. An influence weight $p_{u,v} \in [0, 1]$ is also associated with each edge $(u, v) \in E$, and represents the probability that a node u will influence node v. We assume that T (u) = {τ_1, τ_2, …, τ_M} represents the set of all possible paths that exist in the graph starting from node u and leading to "leaf" nodes and are generated by the Depth-first search (DFS) algorithm. Each path τ_i consists of a sequence of nodes: $\tau_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$. M is the number of all possible paths from a node u and N represents the number of nodes and the index of the terminal node of path τ_i [15].

Let $p^{\tau}_{l,l+1}$, $1 \le l \le N-1$, represent the influence weight (probability) between two successive nodes in path τ. Then F (τ_i) = {f_{i1}, f_{i2}, …., f_{iN}} represents the probability path for every path τ_i starting from node u to be active. Each f_{ij} is equal to $\prod_{l=1}^{j-1} p^{\tau_i}_{l,l+1}$ if $j \ge 1$, and 1 otherwise [15].

Let $\Psi(u, v) = \{\psi_1, \psi_2, .., \psi_L\}$ denote the set of all possible unique paths from a node u to a node v, where each path $\psi_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$ consists of a sequence of nodes, L is the number of all possible paths between nodes u and v, and N represents the number of nodes of path ψ_i, with L ≤ M. Respectively, $\Phi(\psi_i) = \{\varphi_{i1}, \varphi_{i2}, .., \varphi_{iN}\}$ defines the probability for every path ψ_i between nodes u and v, and is calculated in the same way as f_{ij} [16].

Table 1
Notation.

Notation	Description
$p^t_{u,v}$	Influence weight on directed edge (u,v) at time-step t
$\sigma(S^t)$	Influence of a set of nodes S^t to the graph
$A^t(u,v)$	Influence of node u to node v at time-step t
$\Omega^t(u,v)$	Forward cumulative influence of node u to node v, at time-step t
$T^t(u) = \{\tau^t_1, \tau^t_2, \ldots, \tau^t_M\}$	Set of all possible paths starting from node u, at time-step t
$\tau^t_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$	Path consisting of N nodes starting from node u, at time-step t
F (τ^t_i) = {f_{i1}, f_{i2}, …., f_{iN}}	Cumulative probability path for path τ_i, at time-step t
$\Psi^t(u,v) = \{\psi^t_1, \psi^t_2, \ldots, \psi^t_L\}$	Set of all possible paths between nodes u and v, at time-step t
$\psi^t_i = \{n_{i1}, n_{i2}, \ldots, n_{iN}\}$	Path between nodes u and v, at time-step t
$\Phi^t(\psi_i) = \{\varphi_{i1}, \varphi_{i2}, \ldots, \varphi_{iN}\}$	Cumulative probability path for path ψ^t_i, at time-step t

Goyal et al. [5] showed that the spread of a set S of nodes is the sum of the spread of each individual $u \in S$ on the subgraphs induced by the set $V - S + u$:

$$\begin{aligned}\sigma(S) &= \sum_{v \in V}\sum_{X} Pr[X] I(S, v, X) \\ &= \sum_{v \in V} A(S, v) \\ &= \sum_{u \in S} \sigma^{V-S+u}(u)\end{aligned} \tag{1}$$

where X is a possible *live-edge* graph, Pr [X] is the sampling probability of X, I(S, v, X) is an indicator function which equals to 1 if there exists a live path in X from S to v and 0 otherwise, A (S,v) is the probability the single node v to be activated (influenced) by S, and $\sigma^{V-S+u}(u)$ denotes the total influence of u in the subgraph induced by the set $V - S + u$ to denote $((V \backslash S) \cup \{u\})$.

Under the LT model, the calculation of influence after a node x addition to a set of nodes S is given by the following equation:

$$\sigma(S + x) = \sigma(S) + \sigma(x) - \sum_{y \in S} \Omega(x, y) - \sum_{y \in S} \Omega(y, x) \tag{2}$$

Under the IC model the following heuristics [13] are used:

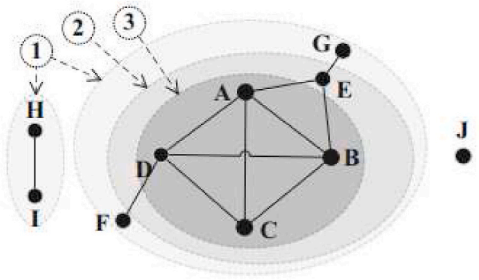
1. for each path originating from node x or a node of seed set S, we keep the subpaths before finding node of $S \cup \{u\}$.
2. $\sigma(S + x)$ is equal to the sum of the influence probabilities that correspond to each of these subpaths.

Also, the forward cumulative influence Ω(u,v) corresponds to the influence of node u to v and to the nodes that can be found right after v in the paths T(u) of node u.

In this paper, we model a dynamic social network $G = \{G^1, G^2, \ldots, G^T\}$ as a set of network snapshots evolving over time. We assume that the nodes remain the same while the edges in each network snapshot change through time. This is used as assumption in other papers [4,11]. Each snapshot graph $G^t = (V, E^t)$ is modeled as a directed network which includes edges appearing during time $t = 1, 2, \ldots, T$. Moreover, an influence weight $p^t_{u,v} \in [0, 1]$ is also associated with each directed edge $(u, v) \in E^t$, and represents the probability of node u to influence node v at time t.

Our goal is to discover a set of seed sets, $S^t, t = 1, 2, \ldots, T$, whose size is k, such that it maximizes the influence $\sigma(S^t)$.

Below, we present the basic notations used in this paper (see



Core number	Vertices
0	J
1	F, G, H, I
2	E
3	A, B, C, D

(a) *k-cores* **(b)** Core numbers

Fig. 1. Example of the k-core decomposition.

Table 1).

Below, some useful definitions and theorems from Refs. [7,16,17] are given for completeness:

Definition 1 *(k-core subgraph)*: Let H be a subgraph of G, (i.e., $H \subseteq G$). Subgraph H is defined to be a *k-core subgraph* of G, denoted by C_k, if it is a maximal connected subgraph in which all nodes have degree at least k.

Definition 2 *(node's core number)*: A node i has *core number* $c_i = k$, if it belongs to a k-core but not to any $(k+1)$-core.

It is evident that if all the nodes of the graph have degree at least one, i.e., $d(v) \geq 1, \forall v \in V$, then the 1-core subgraph corresponds to the whole graph, i.e., $C_1 \equiv G$. Furthermore, assuming that $C_i,\ i = 0, 1, 2, \ldots, k_{max}$ is the i-core of G, then the k-core subgraphs are nested, i.e.:

$$C_0 \supseteq C_1 \supseteq C_2 \supseteq \ldots \supseteq C_{k_{max}}$$

Typically, subgraph $C_{k_{max}}$ is called maximal k-core subgraph of G. Fig. 1 illustrates an example of a graph and the corresponding k-core decomposition.

Definition 3 (*Induced Core Subgraph (ICS)*): The *core subgraph* of G induced by a node $v \in V$, noted ICS(v)=(V_I^V, E_I^V) is the maximal connected subgraph of the $k_G(v)$ such as:

1. $v \in V_I^V$, v is in H (i.e., the vertex inducing the ICS is in the ICS).
2. $\forall v \in V_I^V$, $k_G(u) = k_G(v)$; all nodes of ICS(v) has a coreness exactly equal to (an no greater than) $k_G(v)$.

Theorem 1. *(k-core update theorem)*: *Given a graph G and two nodes u and v in G, the insertion or deletion of an edge between u and v*:

- *if* $k_G(u) > k_G(v)$, *may impact the nodes that belong to ICS(v).*
- *if* $k_G(u) < k_G(v)$, *may impact the nodes that belong to ICS(u).*
- *if* $k_G(u) = k_G(v)$, *may impact the nodes that belong to the union of ICS(v) and ICS(u).*

The coreness of such a node may:

- *remain unchanged.*
- *in the case of an edge addition, increase by* 1.
- *in the case of a suppression, decrease by* 1.

The K-truss decomposition extends the notion of k-core decomposition using triangles, i.e., cycle subgraphs of length 3 [16].

Definition 4 *(Triangle subgraph)*: *Let* $G = (V, E)$ *be a graph. We define as a triangle* Δ_{uvw} *a cycle subgraph of nodes* $u, v, w \in V$. *Additionally, the set of triangles of G is denoted by* Δ_G.

Definition 5 *(Edge* support*)*: *The* support *of an edge* $e = (u, v) \in E$ *is defined as* $\sup(e, G) = |\{\Delta_{uvw} : \Delta_{uvw} \in \Delta_G\}|$ *and expresses the number of triangles that contain edge e.*

Definition 6 *(K-truss subgraph)*: *The K-truss,* $K \geq 2$, *denoted by* $T_K = (V_{T_K}, E_{T_K})$, *is defined as the largest subgraph of G, where every edge is contained in at least* $K-2$ *triangles within the subgraph, i.e.,* $\forall e \in E_{T_K}$, $\sup(e, T_K) \geq K-2$.

Definition 7 *(Edge truss number)*: *The truss number of an edge* $e \in E$ *is defined as* $t_{edge}(e) = \max\{K : e \in E_{T_K}\}$. *Thus, if* $t_{edge}(e) = K$, *then the edge belongs to* T_K *but not to* T_{K+1}, *i.e.,* $e \in E_{T_K}$ *but* $e \notin E_{T_{K+1}}$. *We use* K_{max} *to denote the maximum truss number of any edge* $e \in E$.

Definition 8 *(Node truss number)*: *The truss number of a node* $v \in V$, *denoted by* t_v *is the maximum truss number of its incident edges, i.e.,* $t_v = \max\{t_{edge}(e),\ e = (u, v)\ \forall u \in N(v)\}$, *where* $N(v)$ *is the set of neighborhood nodes of v.*

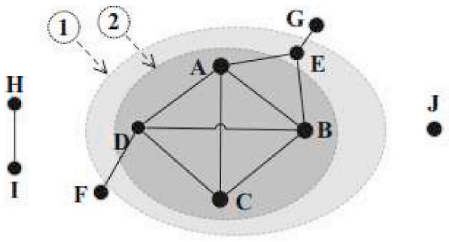
Definition 9 *(K-class)*: *The K-class of graph* $G = (V, E)$ *is denoted as* $\Phi_K = \{e : e \in E,\ \tau_{edge}(e) = K\}$.

Definition 10 *(K-truss decomposition). The K-truss decomposition is defined as the task finding the K-truss subgraphs of G, for all* $2 \leq K \leq K_{max}$. *That is, the K-truss can be obtained by the union of all edges that have truss number at least K, i.e.,* $E_{T_K} = \cup_{j \geq K} \Phi_j$.

Fig. 2 *illustrates an example of a graph and the corresponding K-truss decomposition.*

Theorem 2. *If an edge e is inserted into G, then the* $t_{edge}(.)$ *value of any edge can increase by at most* 1.

Theorem 3. *If an edge e is inserted into G, then for every other edge whose* $t_{edge}(.)$ *value increases from K to* $K+1$, *a triangle is formed with either e or with at least one other edge whose* $t_{edge}(.)$ *value also increases from K to* $K+$



Truss number	Edges
4	(A, B), (A, C), (A, D), (B, C), (B, D), (C, D)
3	(A, E), (B, E)
2	(D, F), (E, G), (H, I)

(a)k-trusses *(b) Trusses*

Fig. 2. Example of K-truss decomposition

1.

We also present the following observations that we made during our study [17].

***Observation 1**: If an edge e is deleted from G, then the $t_{edge}(.)$ value of any edge can decrease by at most* 1.

***Observation 2**: If an edge e is inserted between nodes $u,v \in V_G$, then t_u, t_v values can increase by at most* 1.

***Observation 3**: If an edge e is deleted between nodes $u,v \in V_G$, then t_u, t_v values can decrease by at most* 1.

4. Proposed methods

In this section, we present our proposed methods for the Influence Maximization Problem in Dynamic Networks (DNs): DM, DM-C and DM-T. The proposed methods are extensions of static method MATI [16].

4.1. DM algorithm

DM algorithm is an extension of the static method MATI. Based on the functions A, Ω and influence function σ, we use functions A^t, Ω^t at timestamp t, for $t = 1, 2, \ldots, T$. DM LT is the DM algorithm under Linear Threshold (LT) model, shown in Algorithm 1, while DM IC is the DM algorithm under Independent Cascade (IC) model, shown in Algorithm 2.

Algorithm 1: DM LT

```
Input: G^0, k, T ▷k = number of seed nodes, T = max time-step
Initialize: S^t = 0, ∀t = 1, 2, ..., T
for t = 0 to T do
  if t == 0 then
    Calculate A^t, Ω^t
  Calculate Q^t
  for i = 1 to k do
    s, σ(s) = Q^t.top()
    S^t = S^t ∪ {s}
    U = V\S^t
    for each u ∈ U do
    σ(u) = Q^t(u)
    for each v ∈ S^t do
        σ(u) − = Ω^t(v, u)
        σ(u) − = Ω^t(u, v)
        Q^t.add((u, σ(u)))
    Update G^t to G^{t+1}
    Update A^t, Ω^t
return S = {S^1, S^2, ...., S^T}
```

Algorithm 2: DM IC

```
Input: G^0, k, T ▷k = number of seed nodes, T = max time-step
Initialize: S^t = 0, ∀t = 1, 2, ..., T
for t = 0 to T do
  if t == 0 then
    Calculate A^t
  Calculate Q^t
  for i = 1 to k do
    s, σ(s) = Q^t.top()
    S^t = S^t ∪ {s}
    U = V\S^t
  for each u ∈ U do
      σ(S^t + u) = |S^t ∪ {u}|
      σ(u) + = influence(S^t)
      σ(u) + = influence(S^t ∪ {u})
      Q^t.add((u, σ(S^t ∪ u) − σ(S^t)))
  Update G^t to G^{t+1}
  Update A^t
return S = {S^1, S^2, ...., S^T}
```

4.2. DM-C: DM core algorithm

In DM algorithm, for the computation of functions A^t and Ω^t, the computation of T, F, Φ and Ψ from the beginning for each snapshot under both LT and IC model is required, which is time consuming.

Thus, we propose DM-C algorithm applying k-core decomposition and reclaiming the information of nodes' core number or coreness. Based on Theorem 1, we detect the nodes affected by graph changes - whose coreness changed - and we recalculate those functions for them. In order to reduce the execution time and to be more precise in detection of changes in the graph, we use the core number. Using the knowledge of node's core number we can see where the changes have taken place. Since the computation of paths and factors from the beginning for each snapshot under both LT and IC model is time consuming, after forming the set of changes at the current snapshot G^t, we calculate the new paths and the factors only for the affected nodes by the new paths for computing instances of A^t, Ω^t and the influence of the affected nodes by changes.

Therefore, instead of computing A^t and Ω^t over and over again, we detect where the changes are, and which nodes were infected by them. So, we compute instances of A^t and Ω^t taking into consideration A^{t-1} and Ω^{t-1}, respectively. We name this algorithm DM-C, and more precisely DM-C LT (under LT model) and DM-C IC (under IC model). This approach is 1.5 times faster and more precise in detecting changes than our previous approach, thus detecting new paths. The parts that change in the previous approach are shown in Algorithm 3 (DM-C LT) and Algorithm 4 (DM-C IC), while steps 6–16 and steps 6–15, respectively, remain the same:

Algorithm 3: DM-C LT

```
1   Input: G^0, k, T ▷k = number of seed nodes, T = max time-step
2   Initialize: S^t = 0, ∀t = 1, 2, ..., T
3   for t = 0 to T do
4     if t == 0 then
5a      Apply k-core decomposition
5b      Calculate coreness (G^t)
5c      Hierarch nodes based on their core number c^t(v), v ∈ V
5d      Calculate A^t
          ⋮
17a   for each v ∈ V do
17b       Update c^t(v) to c^{t+1}(v)
17c   CD = {v ∈ V / c^t(v) ≠ c^{t−1}(v)}
17d   Update G^t to G^{t+1} based on CD
18    Update A^t, Ω^t based on CD
19  return S = {S^1, S^2, ...., S^T}
```

Algorithm 4: DM-C IC

```
1   Input: G^0, k, T ▷k = number of seed nodes, T = max time-step
2   Initialize: S^t = 0, ∀t = 1, 2, ..., T
3   for t = 0 to T do
4     if t == 0 then
5a      Calculate coreness (G^t)
5b      Hierarch nodes based on their core number c^t(v), v ∈ V
5c      Calculate A^t
        ⋮
16a   for each v ∈ V do
16b       Update c^t(v) to c^{t+1}(v)
16c   CD = {v ∈ V / c^t(v) ≠ c^{t−1}(v)}
16d   Update G^t to G^{t+1} based on CD
17    Update A^t based on CD
18  return S = {S^1, S^2, ...., S^T}
```

4.3. DM-T: DM truss algorithm

We also propose DM-T algorithm applying K-truss decomposition - an extension of k-core decomposition using triangles - and using the information of edge and node's truss to reduce the computation time of our initial algorithms (DM and DM-C).

Thus, we name these algorithms DM-T LT and DM-T IC, which are almost 2 times faster and more precise than our first approach in detection of changes, using Theorems 2–3 and our Observations 1–3.

The parts that change in the previous approach are shown in Algorithm 5 (DM-T LT) and Algorithm 6 (DM-T IC), while steps 6–16 and

Table 2
Datasets.

Datasets	Description	Nodes	Edges
EmailEuCore	The network was generated using email data from a large European research institution. The e-mails represent communication only between institution members and not with the rest of the world. A directed edge (u, v, t) means that person u sent an email to person v at time t [24].	986	12216 (Temporal), 24929 (Static)
Email-dnc	This is the directed network of emails in the 2015 Democratic National Committee email leak. A directed edge in the dataset denotes that a person has sent an email to another person [23].	92	9800
High school dynamic contact network	These datasets contain the temporal network of contacts between students in a high school in Marseilles, France. In case of multiple active contacts in a given interval, multiple lines start with the same value of t which is measured in seconds [25].	327	188,508
Primary school temporal network data	This dataset contains the temporal network of contacts between the children and teachers. In case of multiple active contacts in a given interval, multiple lines start with the same value of t which is measured in seconds [25].	242	125,775
Hospital ward dynamic contact network	This dataset contains the temporal network of contacts between patients (29), patients and health-care workers (HCWs) and among HCWs (49) in a hospital ward in Lyon, France. In case of multiple active contacts in a given interval, multiple lines start with the same value of t which is measured in seconds [25].	75	32,424
CollegeMsg	This dataset is comprised of private messages sent on an online social network at the University of California, Irvine. Users could search the network for others and then initiate conversation based on profile information. An edge (u, v, t) means that user u sent a private message to user v at time t [24].	1899	59835 (Temporal), 20296(Static)
WikiTalk	This is a temporal network representing Wikipedia users editing each other's Talk page. A directed edge (u, v, t) means that user u edited user v's talk page at time t [24].	1,140,149	7,833,140 (Temporal), 3,309,592 (Static)

steps 6–15, respectively, remain the same:

Algorithm 5: DM-T LT

Input: $\boldsymbol{G}^0$, $\boldsymbol{k}$, $\boldsymbol{T}$ ▷k = number of seed nodes, T = max time-step
Initialize: $S^t = 0, \forall t = 1, 2, ..., T$
for t = 0 to T do
  if $t == 0$ **then**
    Apply K-truss decomposition
    Hierarch nodes based on their truss number $t^t(v)$, $v \in V$
    Calculate A^t
  ⋮
  for each $v \in V$ do
    Update $t^t(v)$ to $t^{t+1}(v)$
  TD $= \{v \in V / t^t(v) \neq t^{t-1}(v)\}$
  Update G^t to G^{t+1} based on TD
  Update A^t, Ω^t based on TD
return $S = \{S^1, S^2, ..., S^T\}$

Algorithm 6: DM-T IC

Input: $\boldsymbol{G}^0$, $\boldsymbol{k}$, $\boldsymbol{T}$ ▷k = number of seed nodes, T = max time-step
Initialize: $S^t = 0, \forall t = 1, 2, ..., T$
for $t = 0$ to T do
  if $t == 0$ then
    Apply K-truss decomposition
    Hierarch nodes based on their truss number $t^t(v)$, $v \in V$
    Calculate A^t
  ⋮
  for each $v \in V$ do
    Update $t^t(v)$ to $t^{t+1}(v)$
  TD $= \{v \in V / t^t(v) \neq t^{t-1}(v)\}$
  Update G^t to G^{t+1} based on TD
  Update A^t based on TD
return $S = \{S^1, S^2, ..., S^T\}$

5. Experiments

5.1. Datasets

In this section, we show experiments on real-world dynamic networks to test the performance of proposed algorithms. The dynamic networks we used in the experiments are listed below. The following table (Table 2) shows the number of nodes and edges of each dataset.

5.2. Evaluation

We compared the performance of the proposed algorithms with the following ones:

- Dynamic Degree Discount [8].
- Dynamic CI [10].
- Dynamic RIS [10].
- Osawa [12].
- MC Greedy [4].

The results of information propagation for different seed sizes k are shown in the following figures with fixed threshold $\theta = 0.1$. The x-axis shows the size of the seed set, while the y-axis shows the number of propagated vertices. Values of x-axis is $\frac{k}{|V|}$*100, i.e. the percentage of seed vertices to all vertices in the network. Values of y-axis is $\frac{\sigma(S)}{|V|}$*100, i.e. the percentage of influence σ(S) to all vertices in the network. In Fig. 3, the comparison of the proposed algorithms for the different datasets is shown. Among the proposed methods, DM-T algorithms (DM-T LT and DM-T IC) achieve higher propagation than the other ones. Specifically, DM-T IC algorithm is better by 15–30% than the rest of proposed ones in terms of diffusion.

As shown in Fig. 4, MC Greedy achieves the highest diffusion in all datasets. Diffusion of the proposed methods are inferior by 5% compared to MC Greedy, although they are better by 20% than the other

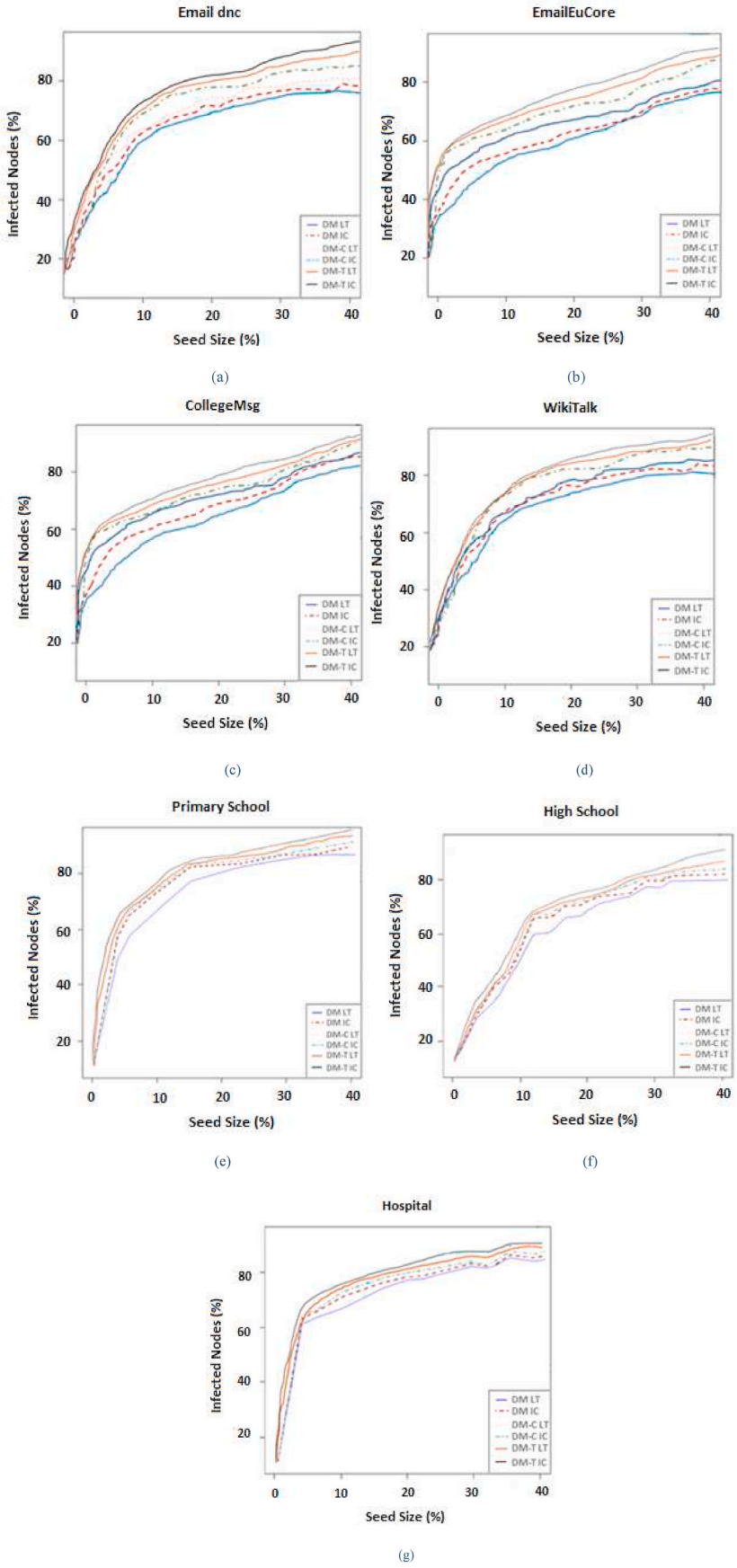


Fig. 3. Comparison of the proposed algorithms DM, DM-C, DM-T in terms of % of infected nodes with varying seed size for different datasets (a) EmailDnc, (b) EmailEuCore, (c) CollegeMsg, (d) WikiTalk, (e) Primary School, (f) High School, (g) Hospital.

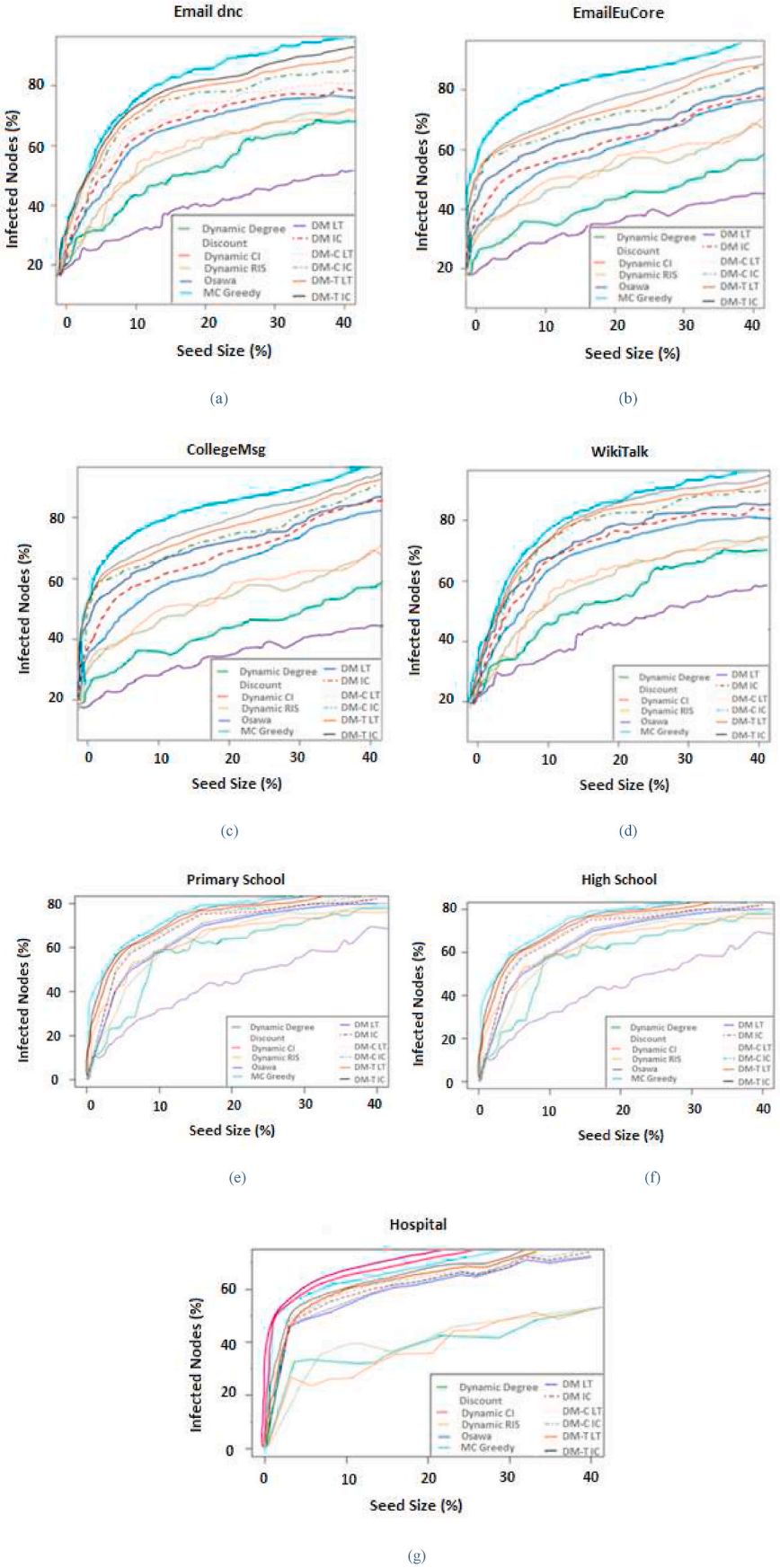


Fig. 4. Comparison in terms of % of infected nodes with varying seed size for different datasets (a) EmailDnc, (b) EmailEuCore, (c) CollegeMsg, (d) WikiTalk, (e) Primary School, (f) High School, (g) Hospital.

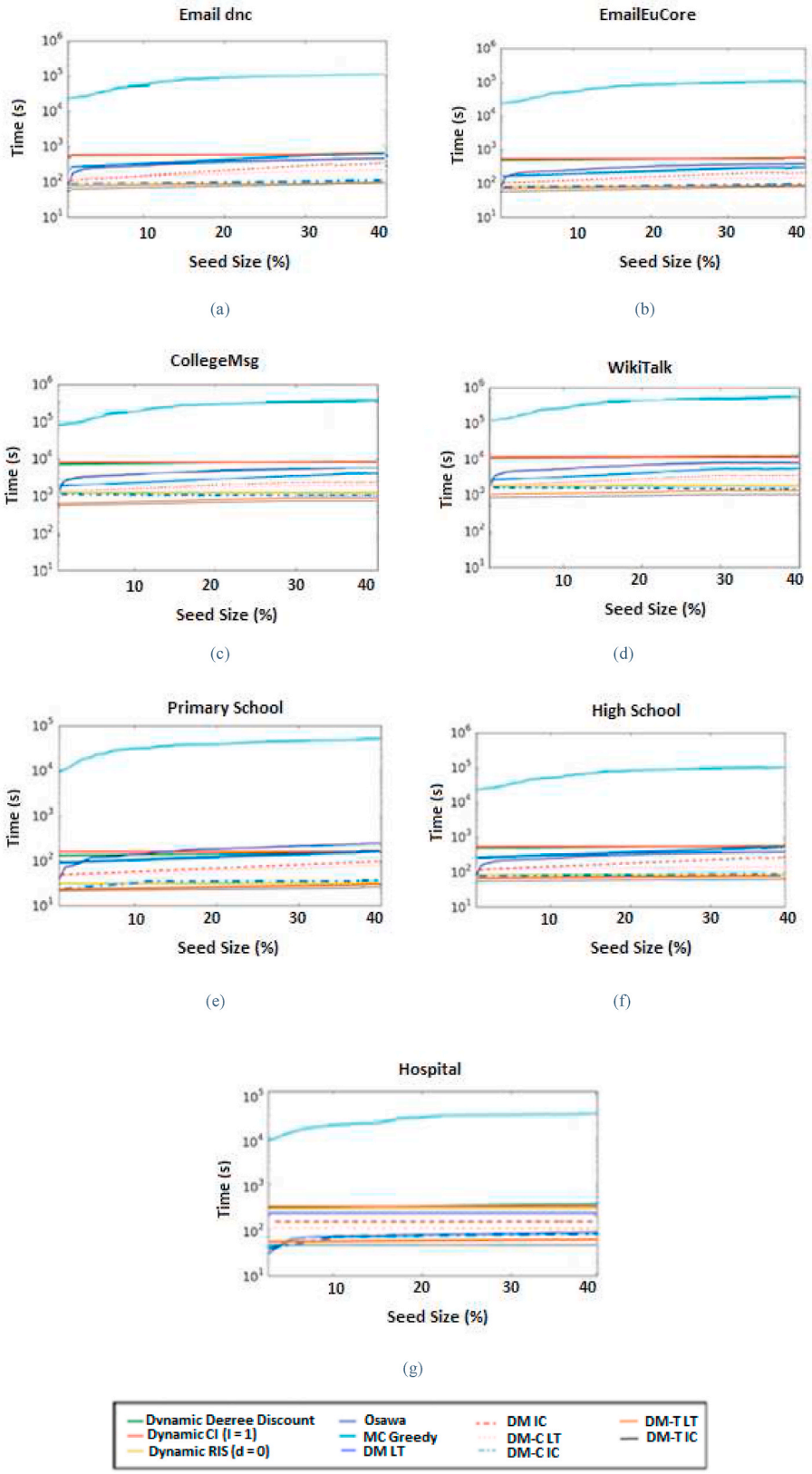


Fig. 5. Comparison of computation time when the seed size k changes for different datasets: (a) Email dnc, (b) EmailEuCore, (c) CollegeMsg, (d) WikiTalk, (e) Primary School, (f) High School, (g) Hospital.

ones (Dynamic Degree Discount, Dynamic CI, Dynamic RIS, Osawa).

Fig. 5 presents the computational time that is needed for θ set to 0.1 and for varying seed sizes. X-axis shows the size of seed vertices, and y-axis shows the computational time in log-scale.

Fig. 5 shows that for all datasets MC Greedy needs several hours to compute seed vertices, while all other methods including the proposed ones can compute seed vertices in several minutes. This shows that MC Greedy is intractable in realistic time for large scale networks. The computational time of proposed methods DM-C LT and DM-C IC is about the same for the Primary School dataset. Similarly, for proposed methods DM-T LT and DM-T IC on the Primary School dataset. Compared with the proposed methods, Dynamic RIS and Dynamic Degree Discount, except for the Primary School dataset - where Dynamic RIS is faster – and the Hospital dataset - where Dynamic Degree Discount and DM-C IC are almost the same – DM-C IC is faster than those or the same. Note that DM-T IC is faster in all the datasets.

6. Conclusion and future work

We proposed DM, DM-C and DM-T – three new methods for the influence maximization problem on dynamic networks which extend methods for static networks. Based on the experiments performed for comparing with previous methods, the proposed methods perform better than the state of the art methods in terms of diffusion, except for MC Greedy which achieves better diffusion. As compared to MC Greedy, our methods are 8.5 times faster. In future work, we plan to investigate the case of partially observed dynamic graphs. In addition, we plan to study the Influence Maximization Problem in Dynamic Networks using models other than LT and IC.

Credit author statement

G.I.Smani: Conceptualization, Methodology, Data curation, Writing – original draft preparation, Validation, Writing- Reviewing and Editing. V. Megalooikonomou: Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Borgs C, Brautbar M, Chayes J, Lucier B. Maximizing social influence in nearly optimal time. In: SODA; 2014. p. 946–57.
[2] Cai T, Li J, Mian A, Li R-H, Sellis T, Yu JX. Target-Aware holistic influence maximization in spatial social networks. 4. In: IEEE transactions on knowledge and data engineering, vol. 34; 2022. p. 1993–2007. https://doi.org/10.1109/TKDE.2020.3003047. 1 April.
[3] Chen S, Wang C, Wang Y. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining -KDD 2010; 2010. p. 1029–38.
[4] Chen S, Wang Y, Yang Y. Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining -KDD 2009; 2009. p. 199–207.
[5] Goyal A, Lu W, Lakshmanan VS. SIMPATH: an efficient algorithm for influence maximization under the linear Threshold model. In: ICDM; 2011.
[6] Habiba CT, Berger-Wolf TY. Maximizing the extent of spread in a dynamic network. 2007. DIMACS Technical Report.
[7] Jakkula VR, Karypis G. Streaming and batch Algorithms for truss decomposition. In: 2019 first international conference on graph computing (GC); 2019. p. 51–9. https://doi.org/10.1109/GC46384.2019.00016. Laguna Hills, CA, USA.
[8] Kempe D, Kleinberg J, Tardos E. Maximizing the spread of influence through a social network. In: KDD; 2003. p. 137–46.
[9] Li G, Chen S, Feng J, Tan K-L, Li W-S. Efficient location-aware influence maximization. In: SIGMOD; 2014. p. 87–98.
[10] Li H, Bhowmick SS, Sun A, Cui J. Conformity-aware influence maximization in online social networks. VLDB J 2015;24(1):117–41.
[11] Li Y, Zhang D, Tan K-L. Real-time targeted influence maximization for online advertisements. PVLDB 2015;8(10):1070–81.
[12] Murata T, Koga H. Extended methods for influence maximization in dynamic networks. Comput Soc Netw 2018;5:8.
[13] Nguyen HT, Thai MT, Dinh TN. Stop-and-stare: optimal sampling algorithms for viral marketing in billion-scale networks. In: SIGMOD; 2016. p. 695–710.
[14] Osawa S, Murata T. Selecting seed nodes for influence maximization in dynamic networks. In: Proceedings of the 6th workshop on complex networks (CompleNet 2015), studies in computational intelligence. Berlin: Springer; 2015. p. 91–8.
[15] Li Rong-Hua, Jeffrey Xu Yu, Mao Rui. Efficient core maintenance in large dynamic graphs. In: IEEE transactions on knowledge and data engineering, vol. 26; 2014. p. 2453–65. 10.
[16] Rossi M, Shi B, Tziortziotis N, Malliaros F, Giatsidis C, Vazirgiannis M. MATI: an efficient algorithm for influence maximization in social networks. PLoS One 2018. https://doi.org/10.1371/journal.pone.0206318. doiQ 13. 1.
[17] Smani GI, Megalooikonomou V. Influence maximization in dynamic social networks and graphs, ICSNAM 2022: 16. In: International conference on social network analysis and mining june 02-03; 2022. Rome, Italy.
[18] Song X., Li J., Lei Q., Zhao W., Chen Y., and Mian A., Bi-CLKT: Bi-graph contrastive learning based knowledge tracing. Know Based Syst 241, 2022.
[19] Tang Y, Shi Y, Xiao X. Influence maximization in near-linear time: a martingale approach. In: SIGMOD; 2015. p. 1539–54.
[20] Tang Y, Xiao X, Shi Y. Influence maximization: near-optimal time complexity meets practical efficiency. In: SIGMOD; 2014. p. 75–86.
[21] Wang X, Zhang Y, Zhang W, Lin X. Distance-aware influence maximization in geo-social network. In: ICDE; 2016. p. 1–12.
[22] Xue G, Zhong M, Li J, Chen C, Kong R. Dynamic network embedding survey. Neurocomputing 2022;472:213–23.
[23] http://networkrepository.com.
[24] https://snap.stanford.edu/data.
[25] http://www.sociopatterns.org.