Influence Maximization on Dynamic Social Networks with Conjugate Learning Automata

Chong Di*, Fangqi Li*, Shenghong Li
School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University
Shanghai, China
*Equal Contribution
{dichong95, solour_lfq, shli}@sjtu.edu.cn

***Abstract*—Selecting the optimal subset from all vertices as seeds to maximize the influence in a social network has been a task of interest. Various methods have been proposed to select the optimal vertices in a static network, however, they are challenged by the dynamics, i.e. the time-dependent variation of the social network structure. Such dynamics hinder the paradigm for static networks and leaves a seemingly unbridgeable gap between algorithms of influence maximization on static networks and those on dynamic ones.**

In this paper, we extend our previous work and demonstrate that conjugate learning automata (an elementary variant of reinforcement learning) that have been successfully applied to maximize influence on static networks can be applied to dynamic networks as well. The network dynamics is measured by the variation of the influence range and absorbed into the learning procedure. Our proposal delicately formulates the effect of network dynamics: the more the influence range varies, the more likely the seeds are to be learned from scratch. Under this assumption, the continuity of the network variation is fully taken advantage of. Experimental results on both synthetic and real-world networks verify the privileges of our proposal against alternative methods.

***Index Terms*—influence maximization, learning automata, social network**

I. Introduction

The online social network has undergone diversified studies concerning its community structure, swarm behavior, etc. Among them, the task of influence maximization (IM) is of particular interest and significance [1]. In IM, a social network is formulated as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ denotes N participants of this network, while their interconnections are embedded in $\mathcal{E}$. IM aims to locate the optimal subset $\mathcal{S}^* \subset \mathcal{V}$ with K participants as seeds such that information propagation from them can affect as many participants as possible. Current mainstream solutions to IM are greedy algorithms, topology-based methods and heuristic ones [1]. It is remarkable that the solutions of IM can be applied to various scenarios with slight modification, examples including [2] [3].

However, the structure of an online social network could vary with time, i.e. both $\mathcal{V}$ and $\mathcal{E}$ might change. Figure. 1 illustrates an example of dynamic network. The challenges derived from network dynamics split into two aspects: On one hand, it is hard to evaluate the influence of a topological change in a complex network, so it is unclear how the optimality of seeds from the previous snapshot can be preserved. For example, in Figure. 1, when $t = 0, 1$ the optimal seed is v_1, however, at $t = 2$, the optimal seed becomes v_2. On the other hand, if we abandon the previous knowledge and run an IM algorithm for any new snapshot independently, then it has to be very efficient, otherwise, we might fail to catch up with the update of the network structure. To address the two challenges, the evolved version of an IM algorithm on dynamic social networks should: (1). Not only consider local topology. (2). Take advantage of the historical information to cut down redundancy. There have been some trial solutions to IM on a dynamic social network by probing [4] [5] or formulating it as a bandit task [6], etc.

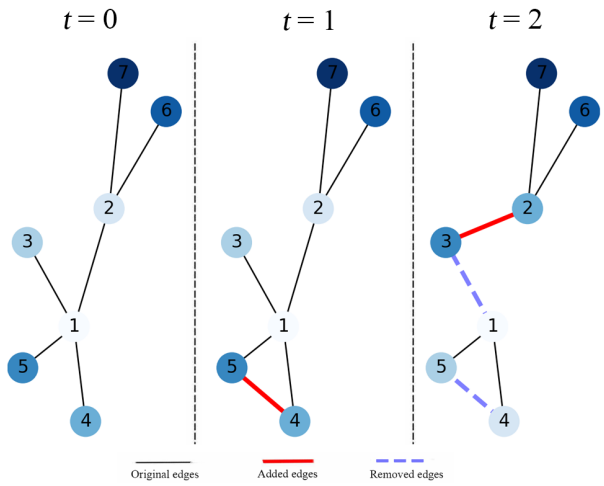

Fig. 1. The toy examples with $N = 7$, $K = 1$. The optimal seed is v_1, v_1, v_2 for $t = 0, 1, 2$.

Recently some researches applied Learning Automaton (LA), an elementary paradigm in reinforcement learning, to find a solution to IM [7]. By modifying the traditional LA, Conjugate Learning Automata (CLA) has been proposed to evade potential adverse pitfalls that impair the performance of greedy algorithms and is comparatively efficient [8]. In CLA, K individual LAs cooperate to find the optimal set of seeds through a learning procedure in which each LA is responsible

for finding one candidate seed.

In this paper, we extend CLA to adapt to dynamic networks and propose Dynamic Conjugate Learning Automata (DyCLA). Roughly speaking, in DyCLA, each individual LA chooses a candidate seed if its learning process converges. Once the variation of the network structure takes place, the learning processes of individual LAs are rewound and a forked learning process is rehearsed in the new network. If the influence range only changes slightly, then the rewinding is also slight so DyCLA can recall previous knowledge and quickly converge to a new optimum. If the influence range changes drastically, then the rewinding is thorough and DyCLA explores the optimal seeds from scratch. The first case demonstrates the efficiency of DyCLA under the docile variations of the network structure, while the second case reflects the flexibility of DyCLA to discover a completely different subset of seeds.

The contributions of this paper are:

1) We apply the learning automata theory to the problem of influence maximization in dynamic social networks and propose Dynamic Conjugate Learning Automata.
2) The proposed method is apt in reflecting the continuous dynamics of the network, and is comparatively efficient.
3) Experimental results on both synthetic and real-world datasets verify the efficacy of the proposed method.

This paper proceeds as follows: Section II reviews the formulation of IM and some attempts to address IM in dynamic networks. Section III presents the proposed method DyCLA. Section IV is devoted to experimental results and subsequent discussions. Section V concludes the paper.

II. Related Works

People have long been studied how the decisions of people are affected by their neighbors and friends or how the "word-of-mouth" affects people's behavior. Specifically, researchers have been paying attention to the diffusion processes of information among participants in social networks. Various general diffusion models have been proposed, including the most basic independent cascade and linear threshold model [1]. Motivated by marketing and advertising, Domingos and Richardson proposed the fundamental algorithmic problem of IM [9] [10] which aims to choose the few key individuals in a social network as the source nodes, or seeds, to maximize the spread of influence, i.e., the number of influenced nodes.

Problem 1. Information Maximization. Given a network $\mathcal{G}$ and the number of seeds K, an information maximization algorithm aims to identify the optimal seed set $\mathcal{S}^*$ such

$$\mathcal{S}^* = \arg \max_{\mathcal{S} \subseteq \mathcal{V}, |\mathcal{S}|=K} \sigma(\mathcal{S}), \tag{1}$$

where $\sigma(\mathcal{S})$ is the number of nodes influenced by seed set $\mathcal{S}$ and is defined according to some diffusion model.

The problem is essentially a combinatorial optimization problem and is NP-hard [1]. The naive greedy method is the simplest approach, it chooses the estimated optimal node one by one using Monte-Carlo simulations. The ratio between the outcome of the greedy method and the optimal one is lower bounded by $(1-\frac{1}{e})$ [1]. Following these groundbreaking works, abundant greedy-based strategies have been proposed to improve the efficiency in choosing the seeds. Among them, the Cost-Effective Lazy Forward algorithm (CELF) [11] that takes advantage of the submodularity of the problem to compute marginal influential gain is the state-of-the-art and is 700 times faster than the naive greedy method. Apart from greedy algorithms, the heuristic methods that explore the properties of $\mathcal{G}$ rather than taking $\sigma(\cdot)$ as a *black box* further improve the efficiency of seed set choosing by narrowing the candidate set. Examples included degree-based methods and community-based methods [12]. However, the structures of social networks are often intractable in reality, and the outcome of a heuristic method can be arbitrarily bad due to the lack of a theoretical boundary.

LA, an elementary variant of reinforcement learning, is known for its adaptivity in a stochastic environment. Therefore it has been successfully applied to solve the information maximization problem since $\sigma(\cdot)$ is a random function. It turns out that the LA-based approach can be even faster than traditional greedy-based methods such as CELF when a single LA is utilized as an appropriate optimizer to find the seeds following the greedy paradigm [7]. The CLA based method is also proposed in the IM problem to obtain better-than-greedy results while preserving some degree of efficiency [8].

The methods discussed above all operate in a static social network where the network structure is invariant. For a dynamic social network that better depicts the reality, $\mathcal{G}$ itself varies between different network snapshots, which brings new challenges.

Problem 2. IM in Dynamic Social Networks. For a series of snapshots of a network in different time: $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_T$ and the budget of seed set K, IM in dynamic social networks aims to maximize the snapshot-wise spread of influence, i.e., the total number of influenced nodes in all snapshots by choosing the time-dependent optimal set of seeds $\mathcal{S}^*(t)$ in each snapshot $t(t = 0, 1, 2, \cdots, T)$ such that

$$\mathcal{S}^*(t) = \arg \max_{\mathcal{S}(t) \subseteq \mathcal{V}(t), |\mathcal{S}(t)|=K} \sigma_t(\mathcal{S}(t)). \tag{2}$$

The methods designed for static social networks have to learn the new set of seeds for each snapshot independently, even if the difference between snapshots turns out to be trivial. This inter-snapshot similarity leaves adequate space for improvement. Zhuang [4] proposed an approximate method Maximum Gap Probing to track the change of dynamic networks through probing a small portion of the network. Inspired by Zhuang, Han [5] improved the efficiency by probing communities instead of nodes. However, both methods in [4] and [5] have the following limitations: (1). They both consider local topology which could turn out to be unreliable. (2). They are both degree-based (and thus heuristic) methods in which the nodes with top K highest degree are chosen as the seeds, which might be ineffective.

In this paper, we generalize LA-based IM algorithms to dynamic networks and provide a new solution to **Problem 2**.

III. Proposed Method

A. Learning Automaton

As an elementary paradigm in reinforcement learning, an LA adaptively explores the optimal action that maximizes the reward among all possible choices by interacting with a stochastic environment. An LA with its environment is formalized as a triplet $< \mathcal{A}, \mathcal{B}, \mathbf{D} >$, where $\mathcal{A} = \{\alpha_1, \alpha_2, \cdots\}$ is the set of possible actions, $\mathcal{B} = \{\beta_1, \beta_2, \cdots\}$ is the set of possible feedback from the environment, and $\mathbf{D}$ is the reward matrix of the environment following

$$\Pr\{\beta_q | \alpha_r\} = d_{r,q}, \beta_q \in \mathcal{B}, \alpha_r \in \mathcal{A}. \tag{3}$$

When $\mathbf{D}$ is fixed, the environment is called *stationary*, otherwise, it is an *non-stationary* environment. For most LA schemes, the training is equivalent to tuning the normalized action probability vector $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, \cdots]$ to maximize the expected reward $\sum_{r,q} d_{r,q} \cdot \mathbf{P}_r \cdot \beta_q$. The training process consists of a number of iterations, during the i-th iteration, the LA selects the action $\alpha(i)$ according to $\mathbf{P}(i)$

$$\Pr\{\alpha(i) = \alpha_r\} = \mathbf{P}_r(i). \tag{4}$$

The environment receives $\alpha(i)$ and returns the feedback $\beta(i)$ satisfying (3). The LA receives $\beta(i)$ and updates $\mathbf{P}(i)$ into $\mathbf{P}(i+1)$ according to some specific strategy, this is often done jointly with some additional information especially estimators [13] denoted by $\mathbf{E}$, formally

$$\mathbf{P}(i+1), \mathbf{E}(i+1) = \mathcal{L}\left(\mathbf{P}(i), \mathbf{E}(i), \alpha(i), \beta(i)\right). \tag{5}$$

The internal state of an LA constitutes of both $\mathbf{P}$ and $\mathbf{E}$. An LA gets converged and terminates its training when $\max_r \{\mathbf{P}_r\} > \mathcal{T}$, where $\mathcal{T}$ is a predefined threshold.

B. Conjugate Learning Automata in Influence Maximization

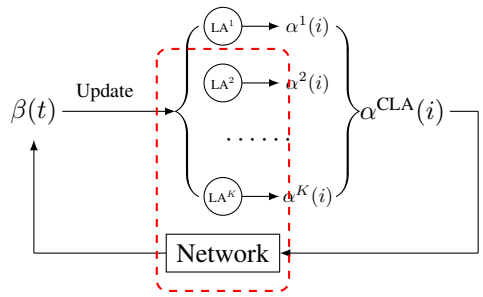

Fig. 2. CLA with K LAs. The external environment of LA^1 is besieged by the dashed red line.

The CLA follows the paradigm of automata game [14] and is capable of avoiding adverse pitfalls that confine the performance of ordinary greedy algorithms in IM [8]. In the context of IM, the CLA consists of the following elements: (where we use superscript to denote the index of seeds and subscript to that of vertices)

- *K individual LAs:* $\text{LA}^1, \text{LA}^2, \cdots, \text{LA}^K$ (where K is the number of seeds), each of which selects one seed, thus $\mathcal{A} = \mathcal{V}^K$. We illustrate a CLA comprising K individual LAs in Figure. 2.
- *Action selection:* At the i-th iteration, an individual learning automaton LA^k selects an action $\alpha^k(i) \in \mathcal{V}$ according to its probability vector $\mathbf{P}^k(i)$. These individual choices are then connected as $\alpha^{\text{CLA}}(i) = \mathcal{S}(i) = \left\{\alpha^1(i), \alpha^2(i), \cdots, \alpha^K(i)\right\}$, the action of the CLA.
- *Environment:* The stochastic environment of CLA is the stochastic propagation function $\sigma(\cdot)$ and $\mathcal{B} = \mathbb{N}$, $\mathbf{D}$ is determined by both the network structure and the set of seeds. A response $\beta(i) = \sigma(\alpha^{\text{CLA}}(i))$ is returned.
- *Learning scheme:* The environment w.r.t. an individual LA^k consists of both the static network and other $(K-1)$ LAs, and could vary during the learning procedure. To ensure the convergence of LA^k, when LA^k is updating its internal states, all the rest $(K-1)$ LAs are kept fixed. Meanwhile, it is undesirable that only one LA converges while others are left totally untrained. So we set a temporary halt threshold $\delta < \mathcal{T}$. The learning of CLA consists of a number of rounds, at each round, each of the K LAs undergoes training while the probability vectors of other $(K-1)$ LAs are fixed. Training only one LA during one round is no different from training an LA in a stable environment, only with the halt condition substituted by $\max_r \{\mathbf{P}_r\} > \delta$. After one round terminated, we increase the value of δ and start another round. The procedure is finished when $\delta \geq \mathcal{T}$. This scheme is summarized as in Algorithm. 1.

Algorithm 1 CLA for IM

Input Initial Temporary threshold: $\delta = \delta_0 \in (0, 1)$.
Input Iterative increment: $\Delta\delta \in (0, 1)$.
Input Convergence threshold: $\mathcal{T} \in (0, 1)$.
Initialize Convergence flag: $\mathcal{I} = 1$.
Initialize K LAs.
Initialize $i = 0$
repeat
  for $k = 1$ to K **do**
    repeat
      $\alpha^{\text{CLA}}(i) = \left\{\alpha^1(i), \alpha^2(i), \cdots, \alpha^K(i)\right\}$, s.t.(4).
      $\beta(i) = \sigma(\alpha^{\text{CLA}}(i))$.
      Update $\mathbf{P}^k(i+1), \mathbf{E}^k(i+1)$ using (5).
      $\mathbf{P}^s(i+1), \mathbf{E}^s(i+1) = \mathbf{P}^s(i), \mathbf{E}^s(i), \forall s \neq k$.
      $++i$.
    until $\max_n \{\mathbf{P}_n^k(i)\} \geq \delta$.
  end for
  if $\delta < \mathcal{T}$ **then**
    $\delta = \min\{\delta + \Delta\delta, \mathcal{T}\}$.
  else
    $\mathcal{I} = 0$.
  end if
until $\mathcal{I} = 0$.
Output $\mathcal{S} = \left\{v_{\arg\max_{n=1}^{N} \{\mathbf{P}_n^k(i)\}}\right\}_{k=1}^{K}$.

Having formulated the update framework of CLA, we spec-

ify the update strategy of one individual LA, namely line 12 in Algorithm. 1. We extend the classic LA algorithm, discretized general pursuit algorithm (DGPA) [15] which enjoys concise structure, low computation cost and analytic optimality. The extended version (eDGPA) for the k-th individual LA is summarized in Algorithm. 2, with $\mathbf{E} = (\mathbf{Z}, \mathbf{R})$.

Algorithm 2 The extended version of DGPA, eDGPA as $\mathcal{L}$

1: **Initialize** Update Step Size: $\Delta = \frac{1}{\mathcal{R} \cdot N}$, where $\mathcal{R}$ is the resolution parameter.
2: **Initialize** Action probability vector: $\mathbf{P}^k(0) = \frac{1}{N}\mathbf{1}_N$.
3: **Initialize** Action selection times vector: $\mathbf{Z}^k(0) = \mathbf{1}_N$.
4: **Initialize** Rewards estimation vector: $\mathbf{R}^k(0) = \mathbf{1}_N$.
5: **Input** $i, \alpha(i), \beta(i)$:
6: Suppose $\alpha(i) = v_n$.
7: **Update** $\mathbf{R}_n^k(i+1) = \frac{\mathbf{Z}_n^k(i) \cdot \mathbf{R}_n^k(i) + \beta(i)}{\mathbf{Z}_n^k(i)+1}$,
8: **Update** $\mathbf{R}_s^k(i+1) = \mathbf{R}_s^k(i), \forall s = 1, 2, \cdots, N, s \neq n$.
9: **Update** $\mathbf{Z}_n^k(i+1) = \mathbf{Z}_n^k(i) + 1$.
10: **Update** $\mathbf{Z}_s^k(i+1) = \mathbf{Z}_s^k(i), \forall s \neq n$
11: $W_n(i) = |\left\{s : s = 1, \cdots, N, \mathbf{R}_s^k(i+1) > \mathbf{R}_n^k(i+1)\right\}|$.
12: **Update** $\mathbf{P}_s^k(i+1) = \min\{\mathbf{P}_s^k(i) + \frac{\Delta}{W_n(i)}, 1\}, \forall s = 1, 2, \cdots, N : \mathbf{R}_s^k(i+1) > \mathbf{R}_n^k(i+1)$,
13: **Update** $\mathbf{P}_s^k(i+1) = \max\{\mathbf{P}_s^k(i) - \frac{\Delta}{N - W_n(i)}, 0\}, \forall s = 1, 2, \cdots, N : \mathbf{R}_s^k(i+1) < \mathbf{R}_n^k(i+1)$,
14: **Update** $\mathbf{P}_n^k(i+1) = 1 - \sum_{s=1, s\neq n}^{N} \mathbf{P}_s^k(i+1)$.

C. Conjugate Learning Automata for Dynamic Networks

Before formalizing Dynamic CLA (DyCLA) that adapts to dynamic networks, we shall consider two extreme cases:

- *Docile variation*: If the influence range of the current choice of seeds is not dramatically affected, then the algorithm should only make a slight revision instead of restarting from the beginning. In this case the historical knowledge can be partially trusted and taken advantage of. As Figure. 1 ($t = 1$).
- *Drastic variation*: If the influence range of the current set of seeds is rapidly increased/decreased, then the algorithm should erase a larger portion of (potentially all) memory and restart itself on this probably new network. In this manner, the algorithm can cope with some peculiar structure variations that potentially bring significant change to the network behavior. As Figure. 1 ($t = 2$).

To conclude, the variation of the network structure results in the variation of the influence range, and should be reflected by the variation of the algorithm' s estimation of the current network. Since CLA has a collection of probability vectors and estimators as a model of memory, it is straightforward to incorporate the observations from the two cases above into CLA:

1) Firstly, after the variation of the structure takes place, the propagation range of the current set of seeds $\mathcal{S}$ becomes $\sigma'(\mathcal{S})$. The significance of this variation is measured by the difference in influence range

$$\Delta\sigma = |\sigma'(\mathcal{S}) - \sigma(\mathcal{S})|. \tag{6}$$

2) Secondly, the convergence of any individual LA in the CLA is relaxed by a parameterized *smoothing function* $f(\Delta\sigma, \cdot)$ that maps an N-dimensional simplex to another N-dimensional simplex, during which the maximal component of the input is reduced. The larger $\Delta\sigma$ is, the more closely the output turns to be a uniform distribution. Essentially, $f(\Delta\sigma, \cdot)$ is the inverse of the update process of an LA. A small $\Delta\sigma$ cancels only the influence of the latest few rounds in Algorithm. 1, and CLA should be able to find another optimal solution quickly. A large $\Delta\sigma$ cancels almost all information as if CLA has just been initialized. Figure. 3 visualizes how the smoothing function acts upon the action probability vector $\mathbf{P}$ of a specific LA. Where the portion of the n-th component denotes the corresponding probability $\mathbf{P}_n$.
3) Thirdly, the information in the estimator has to be perturbed. Any components that record the expected propagation range of a choice is added with a zero-mean stochastic *perturbation* $\Delta R(\Delta\sigma)$ whose variance is monotonic with $\Delta\sigma$. Meanwhile, the selection times vector is set to be noninformative. In this manner the individual LAs are encouraged to explore new candidate seeds.

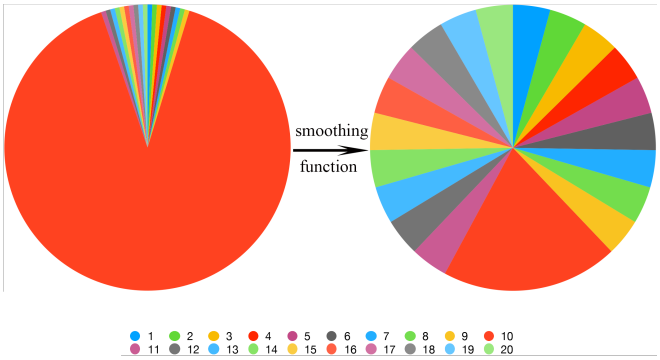

Fig. 3. The visualization of $\mathbf{P}$ being smoothed by the smoothing function, $N = 20$.

Formally, we adopt the following choice of *smoothing function* and *perturbation*

$$f_\phi(\Delta\sigma, \mathbf{P}) : \begin{cases} \mathbf{P}_s = \mathbf{P}_s + \frac{1}{N-1} \cdot \psi, \forall s = 1, 2, \cdots, N, s \neq m \\ \mathbf{P}_m = \mathbf{P}_m - \psi \end{cases}, \tag{7}$$

where $m = \arg\max_n\{\mathbf{P}_n\}$ and

$$\psi = \min\left\{\phi \cdot \frac{\Delta\sigma}{\sigma'(\mathcal{S}) + \sigma(\mathcal{S})} \cdot \mathbf{P}_m, 1\right\}.$$

$$\Delta R(\Delta\sigma) \sim \mathcal{N}(0, \Delta\sigma). \tag{8}$$

At last, the DyCLA can be formulated as Algorithm. 3, where `CLA_for_IM` is Algorithm. 1 with line 5-6 cancelled and $\sigma(\cdot)$ replaced by $\sigma_t(\cdot)$.

Algorithm 3 Dynamic CLA

1: **Input** Initial Temporary threshold: $\delta_0 \in (0, 1)$.
2: **Input** Iterative increment: $\Delta\delta \in (0, 1)$.
3: **Input** Convergence threshold: $\mathcal{T} \in (0, 1)$.
4: **Input** Smoothins parameter: ϕ.
5: **Initialize** K LAs, $i = 0, t = 0$.
6: **Output** $\mathcal{S}(t) = \texttt{CLA\_for\_IM}(\delta_0, \Delta\delta, \mathcal{T}, i, t)$.
7: **repeat**
8: $\sigma_0(t) = \sigma_t(\mathcal{S}(t))$.
9: $++t$. //Network structure changes.
10: $\Delta\sigma(t) = |\sigma_t(\mathcal{S}(t-1)) - \sigma_0(t)|$.
11: **for** $k = 1$ to K **do**
12: $\mathbf{P}^k = f_\phi(\Delta\sigma(t), \mathbf{P}^k)$.
13: $\mathbf{Z}^k = \frac{\mathbf{1}_N^\top \mathbf{Z}^k}{N} \cdot \mathbf{1}_N$.
14: **for** $n = 1$ to N **do**
15: $\Delta R \sim \mathcal{N}(0, \Delta\sigma(t))$.
16: $\mathbf{R}_n^k += \Delta R$.
17: **end for**
18: **end for**
19: **Output** $\mathcal{S}(t) = \texttt{CLA\_for\_IM}(\delta_0, \Delta\delta, \mathcal{T}, i, t)$.
20: **until** No more snapshots.

The advantages of the proposed DyCLA method over established methods are as follows:

- The significance of a network structure variation is measured only through the corresponding difference in the influence range. For large networks, it is generally unfair to evaluate the impact of a structure variation only through local topology as in [4] [5].
- DyCLA does not explicitly distinguish a static network from dynamic ones (let $\Delta\sigma = 0$ in line 10 in Algorithm. 3 then line 11-18 keep the CLA intact and line 19 makes no difference to original $\mathcal{S}$). Therefore our proposal degenerates gracefully to its counterpart for static networks, while for some proposals, the version for static networks and that for dynamic ones are differentiated.
- For a fixed snapshot of the network, our proposal yields a relatively sophisticated choice of $\mathcal{S}$. Due to the inconsistency between static and dynamic cases and the demand of efficiency, many proposals have cut down the resource devoted to a fixed snapshot in time series. Meanwhile, DyCLA with good consistency and high efficiency can preserve the same efficacy compared with algorithms that are specialized for static networks.

IV. Experimental Results and Discussions

A. Experimental Settings

All experiments are conducted under the popular weighted cascade model [12], where a node $v \in \mathcal{V}$ is influenced by edge $(u, v) \in \mathcal{E}$ with probability $\frac{1}{\text{in-degree}(v)}$ in directed graphs or $\frac{1}{\text{degree}(v)}$ in undirected graphs. For comparison, the representative greedy-based algorithm CELF [11] and the greedy LA-based algorithm IMLA [7] serve as the baselines. The heuristic methods have generally smaller spread range and are saved from comparisons. For parameters in DyCLA, the resolution parameter $\mathcal{R}$ for eDGPA is set as K. $(\delta_0, \Delta\delta, \mathcal{T})$ are parameterized as $(\frac{1}{K}, \frac{1}{2K}, 0.999)$ in all experiments, which is the same as [8]. For CELF, the number of Monte-Carlo simulations is uniformly set to 10000, following the consensus in literature.

B. Toy Example

Figure. 4 visualizes the change of $\mathbf{P}$ across snapshots in Figure. 1. It can be observed that: (1). DyCLA correctly converges to the optimum at all snapshots. (2). When the variation of the influence is docile ($t = 1$), the smoothing function helps to memorize useful knowledge and accelerate convergence. (3). When the influence range of outdated seeds is significantly declined, DyCLA can veto the previous decision and converge to a better choice ($t = 2$).

C. Large-Scale Networks

Further evaluations on large-scale networks are conducted on both synthetic datasets and real-world datasets. The metrics of interest are the required number of interactions with the network, i.e. the times we calculate $\sigma(\cdot)$, and the spread range of different algorithms in different snapshots. The synthetic dataset with 800 vertices and 6 snapshots is randomly generated. The structure variation from $t = 0$ to $t = 4$ are docile (the optimal set of seeds are identical), while a drastic variation is coined before $t = 5$. For real-world dataset, we adopt Enron dataset [16] which collects e-mail interconnections from 150 senior executives, altogether 2359 users are involved. We use the record from December 1999 to April 2000 as 6 snapshots (each snapshot corresponds to one month) that constitute the real-world dynamic social network dataset. In both cases we let $K = 5$. Figure. 5 and Figure. 6 present the experimental results, from which we can observe:

- DyCLA converges quickly after docile variations and thousands of times of interactions are reduced compared with algorithms for static networks, this fact verifies its effectiveness, as shown in Figure. 5(a) and Figure. 6(a).
- Figure. 5(a) and Figure. 6(a) indicates that smaller ϕ leads to higher efficiency, *vice versa*. This is in accordance with the intuition since smaller ϕ means less cancellation of the memory and faster convergence.
- Figure. 5(b) and Figure. 6(b) shows that if ϕ is too small then DyCLA might fail to discover the optimal seeds, while a larger ϕ ensures correct convergence. Intuitively, if DyCLA erases little knowledge, then it would overfit the previous network and be trapped in a local optimum. Therefore ϕ reflects the trade-off between accuracy and efficiency in DyCLA.

V. Conclusion

In this paper we propose DyCLA to address the problem of influence maximization in dynamic social networks. By

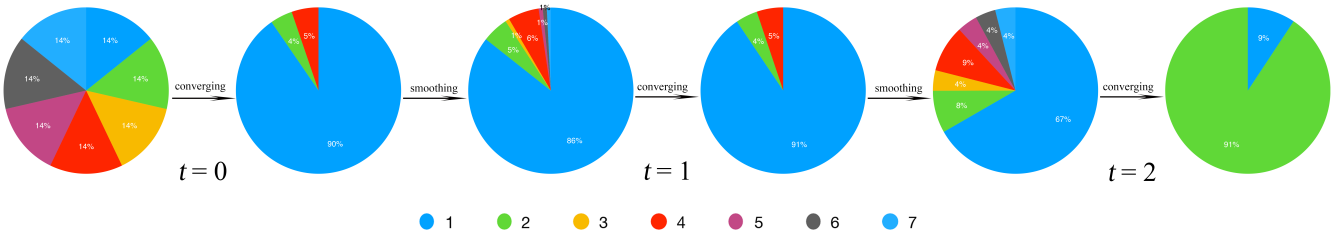

Fig. 4. The changes of $\mathbf{P}$ during the IM process among snapshots.

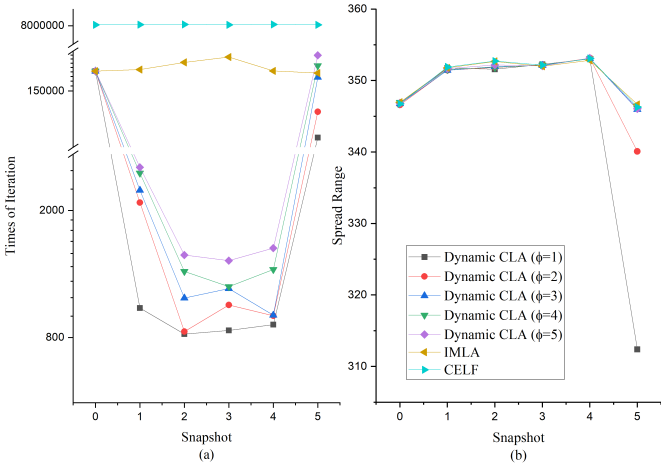

Fig. 5. The required number of interactions with the network and the spread range of different algorithms in synthetic network with $N = 800, K = 5$.

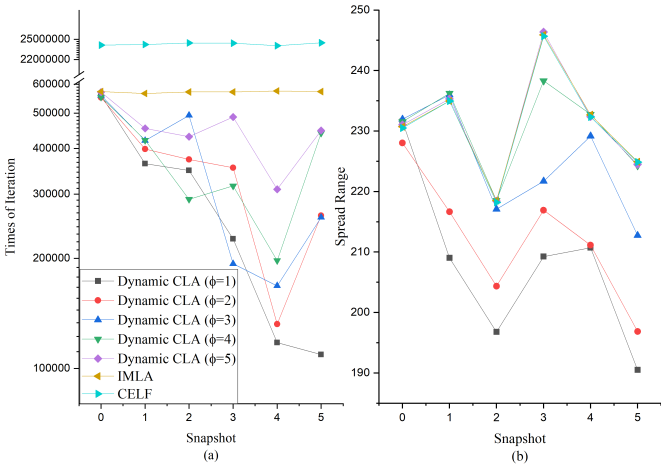

Fig. 6. The required number of interactions with the network and the spread range of different algorithms in real-world network with $N = 2359, K = 5$.

incorporating the inverse of the update procedure, DyCLA straightforwardly generalizes its counterpart for static social networks. This generalization comprehensibly reflects our intuition about the structure variation. Experimental results verify the privileges of DyCLA against established methods.

ACKNOWLEDGMENT

This work was supported by the National Nature Science Foundation of China under Grant 61971283.

REFERENCES

[1] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 137–146.

[2] C Wei, C Wang, and Y Wang. "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*. ACM, 2010, pp.1029-1038.

[3] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. "Cost-effective outbreak detection in networks" in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp.420-429.

[4] H Zhuang, Y Sun, J Tang, J Zhang, and X Sun. "Influence maximization in dynamic social networks." *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013.

[5] M Han, M Yan, Z Cai, Y Li, X Cai, and J Yu. "Influence maximization by probing partial communities in dynamic online social networks." *Transactions on Emerging Telecommunications Technologies* 28.4 (2017): e3054.

[6] Y Bao, X Wang, Z Wang, C Wu, and Francis C.M. Lau. "Online influence maximization in non-stationary social networks." *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, 2016.

[7] H Ge, J Huang, C Di, J Li, and S Li. "Learning automata based approach for influence maximization problem on social networks." *IEEE Second International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2017, pp.108-117.

[8] C Di, F Li, K Qi, and S Li. "Maximizing Influence on Social Networks with Conjugate Learning Automata." *2019 IEEE Global Communications Conference*. IEEE, 2019.

[9] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 57–66.

[10] Richardson, Matthew, and Pedro Domingos. "Mining knowledge-sharing sites for viral marketing." *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002.

[11] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. "Cost-effective outbreak detection in networks" in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp.420-429.

[12] Y Li, J Fan, Y Wang, and KL Tan. "Influence maximization on social graphs: A survey." *IEEE Transactions on Knowledge and Data Engineering* 30.10 (2018): 1852-1872.

[13] Narendra, Kumpati S., and Mandayam AL Thathachar. *Learning automata: an introduction*. Courier Corporation, 2012.

[14] Fu, King-Sun, and Timothy J. Li. "Formulation of learning automata and automata games." *Information Sciences* vol.1, no.3, pp.237-256, 1969.

[15] Agache, Mariana, and B. John Oommen. "Generalized pursuit learning schemes: new families of continuous and discretized learning automata." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* vol.32, no.6, pp.738-749, 2002.

[16] L Tang, H Liu, J Zhang, Z Nazeri. "Community evolution in dynamic multi-mode networks." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.