Information Sciences 569 (2021) 138–156

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Leave or not leave? Group members' departure prediction in dynamic information networks

Xinrui Wang [a], Hong Gao [a,*], Zhipeng Cai [b], Jianzhong Li [a]

[a] *School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China*
[b] *Department of Computer Science, Georgia State University, Atlanta, GA, USA*

A R T I C L E   I N F O

Article history:
Received 21 June 2020
Received in revised form 23 November 2020
Accepted 2 April 2021
Available online 8 April 2021

2010 MSC:
00-01
99-00

Keywords:
Dynamic information network
Unsupervised group members' departure prediction
Users' group-combined score
Sliding window

A B S T R A C T

Dynamic information networks, containing evolving nodes and links, exist in various applications. For example, in a Facebook network, nodes represent users, links represent friendship, and users often form different groups. Over time, some users will leave some groups. Thus, for both users and groups, it's meaningful to predict which users would leave which groups. Existing studies consider that a user is more likely to leave if most of his/her friends leave. However, in reality, the low-degree nodes usually make up the majority of all nodes and there exist some users with very few friends but keeping staying in a group. Therefore, it might lead to precision loss if only neighborhood information is utilized, but previous work ignored this. To improve prediction precision, we firstly introduce novel definitions of a group's activeness and a user's own activeness respectively. Then we propose the group-combined activeness score of each user so that a user with a lower score would be more possible to leave. After that, we present an unsupervised prediction algorithm to continuously predict group members' departure behaviors in a dynamic information network. Experiments on several real datasets show the effectiveness, efficiency and superiority of our algorithm compared with the state-of-the-art competitors.

1. Introduction

Information networks, containing a large number of objects interacting with each other, are omnipresent in practical applications. Examples include online bibliographic databases like DBLP,[1] mobile internet applications like Brightkite,[2] social websites like Facebook,[3] etc. In reality, information networks are usually dynamic where objects and links among them change drastically over time. There have been a large number of papers focused on dynamic information network mining and analysis, such as influence maximization [7,11,36,39], recommendations [33,34,41], community detection [29], link prediction [28,37], etc.

In recent years, one of the most interesting problems in dynamic information networks is to help a user to decide whether to join in a community [3,8] or adopt a product [13]. All the work follows the basis that the decision of a user mainly depends

* Corresponding author.
E-mail address: honggao@hit.edu.cn (H. Gao).
[1] http://dblp.uni-trier.de/
[2] https://brightkite.com/
[3] https://www.facebook.com/

https://doi.org/10.1016/j.ins.2021.04.015
0020-0255/© 2021 Elsevier Inc. All rights reserved.

on the decision of his/her neighbors, i.e., a user is more likely to join in a community or adopt a product if most of his/her friends do. Therefore, the number of a user's friends and the interactions among them are usually extracted and analyzed in these studies.

Similarly, a user may also decide to depart from a community or abandon a product. As far as we known, existing studies of this problem are mainly based on that an individual decides to leave if most of his/her friends leave [21,32]. There also exists some work paying attention to preventing a user from leaving a community [6,35] but they all assume that a user will not leave if he/she has a sufficient number of friends keeping engaged.

However, we observe that there exist a large number of nodes with low degree in real-word information networks, especially which follow the power law (the majority of nodes have low degree but a small number of nodes have high degree). In addition, the community might be formed by an abstract concept (e.g. a common research topic, the same geographical region, etc.) so that the nodes in it have relatively few links among themselves, which we call both such a community and a traditional tightly-knit community as a *group* in our paper. There is little neighborhood information available for predicting departure behaviors of the users with few friends. It might be unreasonable if we predict that all these users would leave based on previous work [21,32]. In the following, we present a few examples and discuss the motivation of our paper.

Group Members' Departure Examples. A Facebook information network is composed of a large number of users. Links exist between them if they are friends. The users often form different small online chat groups so that they can share something they are all interested in more conveniently. Fig. 1 shows an example of a Facebook information network with four snapshots (G_1 to G_4). The blue nodes belong to group S^1, the red nodes belong to group S^2. Note that a user can belong to several groups simultaneously, like user j in snapshot G_2. The number marked on each node represents the number of messages a user sends in a group at the timestamp of the snapshot, and the same color corresponds to the same group (the numbers are all set to 1 only for easy illustration). It is obvious that users between different groups can be friends but it is not the focus of our paper, so we omit the edges whose endpoints are in different groups for convenience. According to previous work [21,32], in snapshot G_4, user g would be predicted to leave group S^1 because its degree is 0 (which means that he/she has no friends in group S^1 at the fourth timestamp). However, in reality, g keeps staying in S^1 after the fourth timestamp due to his/her long-term interest about the information shared by the others in S^1. Observing G_1 to G_4, we find that g has sent messages in S^1 on three snapshots, and the group S^1 keeps active from the first timestamp to the fourth timestamp since its members keep chatting during the period. Motivated by this, we propose to utilize both a user's activity information in a group and the group's own activeness to predict whether the user would leave the group rather than only his/her neighborhood information.

As another example, a DBLP information network consists of many authors. Links between them represent the cooperative relationships. Each group corresponds to a research topic. An author belongs to a topic group if he/she has published a paper on the topic. There exist some authors keeping publishing papers on a topic for several years but having very few cooperators. It might lead to some precision loss if only the number of an author's cooperators was utilized for predicting his/her abandonment on a topic. In this paper, we predict whether an author would abandon a research topic based on both his/her own publications and the development of the topic over time.

As far as we known, given the groups on several snapshots in a dynamic information network, we are the first to combine groups' activeness with users' own activeness to predict users' departure behaviors to improve the prediction precision. The first challenge is how to define a group's activeness and a user's activeness, and how to compute these measurements efficiently. Another challenge is how to combine them into the definition of a user's group-combined activeness score so that a user with a lower score would have a larger probability to leave the group. In addition, an unsupervised prediction method is necessary since it is labor-intensive, difficult, sometimes even impossible to label really leaving users in real dynamic information networks. This makes the problems more demanding because we have to depend on limited historical information to give reasonable definitions of a user's activeness, a group's activeness and a user's group-combined activeness score so as to acquire satisfactory prediction results.

In summary, the contributions of the paper are as follows.

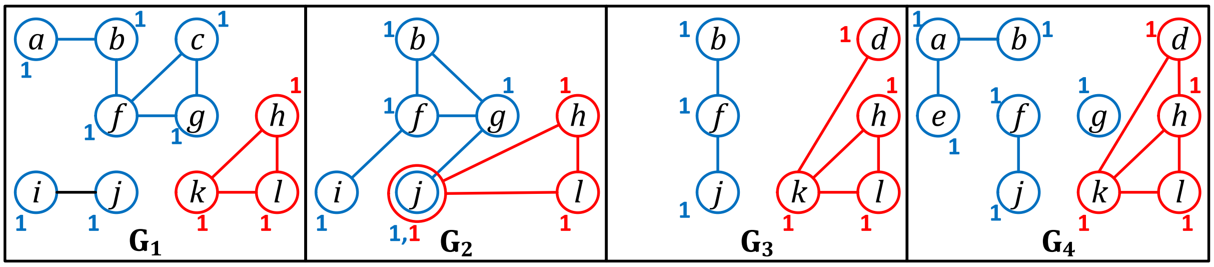


Fig. 1. Four Snapshots of a Dynamic Facebook Information Network (the blue nodes belong to group S^1, the red nodes belong to group S^2).

1. We study the problem of group members' departure prediction in dynamic information networks based on both groups' activeness and users' own activeness to improve the prediction precision.
2. we define a group's activeness considering both its structure features (stability and continuity) and members' activity information. We also propose some techniques to efficiently compute each group's activeness.
3. We respectively give the definitions of a user's time-dependent activeness and a user's group-combined activeness score. After that, we present an unsupervised algorithm to predict whether a user would leave a group with a sliding window model.
4. We perform a large number of experiments on several real-world dynamic information networks which show the effectiveness, efficiency and superiority of our algorithm compared with the best existing algorithms.

The remaining of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the problem definition of group members' departure prediction in dynamic information networks. Section 4 introduces a group's activeness, a user's time-dependent activeness and a user's group-combined activeness score, and presents techniques to compute them efficiently. After that, the group members' departure prediction algorithm is proposed. Section 5 tests the effectiveness, efficiency and superiority of our prediction algorithm. Finally, Section 6 concludes the paper.

2. Related work

We review the existing studies related to group members' departure prediction from four categories, namely, disease propagation, relation discovery and link prediction, user engagement, and group formation and evolution.

Disease Propagation. In decades, there has been growing interest in infectious disease propagation models which transform the process of disease propagation into mathematical models based on some theories like the percolation theory [9,22]. In general, infective individuals infect randomly chosen individuals, or recover and acquire immunity (or die) at some probabilities. Motivated by these models, some work studies the problems like the spread of rumors or news, participation of users in a community, adoption or recommendation of products in many kinds of information networks [16,18,20]. In addition, the disease propagation models often include a recovery component which considers the recovery process as a reversal of the infected process, and assumes that an infected individual recovers based on properties of the immune system. However, the experiments conducted in [32] shows that in information networks, the dynamics of users' departure behaves differently from the dynamics of their joining in. Additionally, the propagation probabilities are difficult to be acquired for isolated users. Therefore, the recovery component in disease propagation models can't be adapted to group members' departure prediction.

Relation Discovery and Link Prediction. Recently, there has been much attention on relation discovery which finds latent or missing relations between entities in different kinds of applications [2,14,27]. Moreover, relation discovery has been utilized for collective classification, community search, and community detection [19,31,40]. Nevertheless, relation discovery aims to find those relations which had been previously ignored or missing but actually exist between entities. Thus, relation discover can't be utilized to predict, in the future, which members would leave which groups.

In addition, many recent studies have focused on link prediction which predicts links between objects in networks [28,37]. However, link prediction pays attention to those links that are between individuals, while group members' departure prediction concentrates on the relationship that is between an individual and a whole group. As a result, link prediction can't apply to group members' departure prediction.

User Engagement. User engagement in social networks, referring to the extent to which a user participates (or is encouraged to participate) in a community [21], has been extensively studied in recent years [4,23]. For users' departure prediction, Wu et al. [32] firstly present a supervised classification method by collecting many neighborhood features and then observe that the departure of nodes is proportional to their position in the network (i.e., nodes in the fringe of the network have higher probabilities to leave). After that, Malliaros et al. [21] give a more refined model. They consider that the property of a user's departure is complementary to that of the user's engagement, and then utilize the core number of a user to measure his/her engagement level (the core number of each node is obtained by k-core decomposition [5]) so that a user would leave if his/her core number is less than a given integer. In addition, Bhawalkar et al. and Zhang et al. all agree that a user may leave a community if he/she has less than k friends engaged [6,35] when studying the problems of preventing community unraveling. In practice, there exist some users with very few friends but keeping engaged in the groups. However, all the above algorithms ignore this and predict that these users would leave. In this paper, to improve the prediction precision, we combine a group's activeness with a user's activeness to predict whether the user would leave the group. What's more, our algorithm is completely unsupervised, which is exceedingly suitable for real-world dynamic information networks.

Group Formation and Evolution. Previous work has extensively investigated the formation and evolution of groups in many practical applications [10,15,26,38]. However, they mainly focus on analyzing the whole group rather than the individual members. Backstrom et al. and Michael Chwe help users to decide whether to join in a group [3,8]. Han et al. [12] predict who will be invited to join the groups on a real-world social instant messaging network WeChat with a probabilistic graph model. But these problems are essentially different from group members' departure prediction. By analyzing the growth of Facebook, Ugander et al. [30] find that a user with higher structural diversity (denoted by the number of connected components in his/her contact neighborhood) has higher probability to join and be engaged after joining. Nevertheless, for low-degree users, the number of connected components in their neighborhood is often really small and even equals to 0. Obviously, it is unreasonable and leads to great precision loss if all the low-degree users with low structural diversity are predicted to leave. By studying the social group evolution in two datasets, the co-authorship network and the phone-call network, Palla et al. [24] find that the users with more neighbors outside a group are more likely to leave the group. However, they also ignore that the low-degree users might have no neighbors outside the group so that all these users would be predicted to stay. Thus, this method would also cause precision loss. We compare its performance with our algorithm by some experiments in Section 5.

3. Problem definition

We present some basic definitions and formalize the group members' departure prediction problem in this section. Table 1 summarizes the mathematical notations used throughout this paper.

Information Network. We model each *information network* as an undirected graph $G = (V, E)$, where V is the vertex set, $E \subseteq V \times V$ is the edge set. Without ambiguity, we will interchangeably use the terms *network* and *information network* in the rest of the paper.

Group. Given a network $G = (V, E)$, a *group* is a subset of vertices $S \subseteq V$ with some kinds of common features in reality, such as some people who focus on the same research topic, who travel in the same geographic region or who join in the same online chat group.

Note that the vertices inside a group S might not be cohesively connected by internal edges within S. For example, in a Brightkite network, the vertices represent users and the edges between them represent the friendships. A group is the users who check in at the same region. There exist some users who have few friends but we can find which group they belong to according to the regions where they check in.

Given a group S, we denote $G[S] = (S, E(G[S]))$ as an induced subgraph of G corresponding to the group S if and only if $S \subseteq V$ and $E(G[S]) = \{(v, u) | v, u \in S, (v, u) \in E\}$. In Fig. 1, $G[S^1]$ is in blue color, $G[S^2]$ is in red color.

Dynamic Information Network. A *dynamic information network*, denoted as $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$, is an infinite sequence of information networks, where $G_t = (V_t, E_t)$ $(t \geqslant 1)$ is a snapshot of the network associated with timestamp t. We use $\mathcal{S}_t = \{S_t^1, S_t^2, \ldots, S_t^{|\mathcal{S}_t|}\}$ to denote the set of groups on the snapshot G_t. Note that S_t^i and $S_{t'}^i$ represent the same group S^i at different timestamps t and t'. Then $G[S_t^i] = (S_t^i, E(G[S_t^i]))$ is the induced subgraph of the snapshot G_t corresponding to the group S_t^i. Without ambiguity, we will interchangeably use S_t^i to represent a group and its corresponding label in the rest content.

Given a snapshot $G_t = (V_t, E_t)$ and a group S_t^i, for any two vertices $v, u \in S_t^i$, if the edge $(v, u) \in E(G[S_t^i])$, we mark (v, u) with a group label S_t^i. Then we denote $L_t(v, u) = \{S_t^i\}$ as the label set marked on the edge (v, u) on the snapshot G_t.

For each vertex $v \in V_t, P_t(v) = \{(S_t^i, w_t^i(v)) | w_t^i(v) \in R^+\}$ is a set of group-weight pairs, where each pair $(S_t^i, w_t^i(v))$ represents that v belongs to group S_t^i and has a corresponding weight $w_t^i(v)$. The weight might be the number of papers an author published about a research topic, the number of check-ins a person did in a region, or the number of messages a user sent in an online chat group at timestamp t. In addition, for each vertex $v \in S_t^i$, we denote $N_t^i(v) = \{u | (v, u) \in E(G[S_t^i]), u \in S_t^i\}$ as the neighbor set of v in group S_t^i, and $d_t^i(v) = |N_t^i(v)|$ as the degree of v in group S_t^i.

Sliding Window. Given a dynamic network $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$, a starting time r and a fixed *window size* τ, the *sliding window* W_r is the set of τ snapshots: $G_r, G_{r+1}, \ldots, G_{r+\tau-1}$. In the rest of the paper, we use the term *window* to refer to the *sliding window*.

When new snapshots arrive, the window slides to incorporate μ new snapshots of the dynamic network, and then the oldest μ snapshots will be discarded from the current window. μ denotes the *slide size* which characterizes the evolving speed of the dynamic network. W_r's next window is defined as $W_{r+\mu}$, which consists of τ snapshots: $G_{r+\mu}, G_{r+1+\mu}, \ldots, G_{r+\tau-1+\mu}$.

Core Number. Letting H be a subgraph of $G[S_t^i]$, we define H as a *k-core* $(k \in N)$ of $G[S_t^i]$ iff $\forall v \in V(H) : d_H(v) \geqslant k$ and H is a maximum subgraph with this property, where $V(H)$ is the vertex set of H and $d_H(v)$ is the degree of v in H. Then a vertex v in S_t^i has a *core number* k in $G[S_t^i]$ if it belongs to a k-core of $G[S_t^i]$ but not to any $k+1$-core of $G[S_t^i]$. The work in [5] presents an algorithm to compute the core number of each node in a graph with the time complexity $O(e_0)$ where e_0 is the number of all edges in the graph. According to the work made by Malliaros et al. [21], a member whose core number in $G[S_t^i]$ is less than a given integer would leave the group S_t^i at timestamp t. Therefore, the time complexity of the algorithm in [21] to predict which user would leave which group at timestamp t is $O(m'e_1)$ where m' is the number of all groups on the snapshot G_t and e_1 is the maximal number of edges of these groups, i.e., $e_1 = \max\{|E(G[S_t^1])|, |E(G[S_t^2])|, \ldots, |E(G[S_t^{|\mathcal{S}_t|}])|\}$.

Static Triangle of A Group. Given a snapshot G_t of a dynamic network $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$ and a group S_t^i on G_t, we denote $STA_t^i(v, u, w)$ as a static triangle of S_t^i on G_t if and only if $v, u, w \in S_t^i$, and $(v, u), (v, w), (u, w) \in E(G[S_t^i])$. A static triangle of a group shows a strong and stable relationship among three group members. Therefore, the number of static triangles of a group S^i at the timestamp t, denoted as α_t^i, shows the degree of *stability* of S^i at t.

Table 1
The Summary of Notations.

Notation	Description
t, x	a timestamp
G_t	a snapshot of a dynamic information network associated with t
v, u, w	a user in a dynamic information network
S^i	a group in a dynamic information network
S^i_t	the group S^i on G_t
W_r	a sliding window with the start timestamp r, window size τ and slide size μ
$w^i_t(v)$	the weight of v in S^i_t
α^i_t	the number of static triangles of S^i at t
β^i_{t+1}	the number of dynamic triangles of S^i from t to $t+1$
σ^i_t	the sum of the weights of all vertices in S^i_t
$\Delta^i_{t+1}(\alpha)$	$\alpha^i_{t+1} - \alpha^i_t$
$\Delta^i_{t+1}(\sigma)$	$\sigma^i_{t+1} - \sigma^i_t$
$GA_r(S^i)$	the activeness of S^i in W_r
$UA_r(v, S^i)$	the activeness of v in S^i in W_r
$AS_r(v, S^i)$	the group-combined activeness score of v in S^i in W_r
θ	an activeness threshold
λ	a time-dependent parameter
φ	the real size of the ground truth assumption

Dynamic Triangle of A Group. Given two contiguous snapshots G_t and G_{t+1} of a dynamic network $\mathscr{G} = (G_1, G_2, \ldots, G_t, \ldots)$ and a group S^i (which is represented as S^i_t on G_t and S^i_{t+1} on G_{t+1} respectively), we denote $DYN^i_{t+1}(v, u, w)$ as a dynamic triangle of S^i from the timestamp t to $t+1$ if and only if the following two conditions are satisfied simultaneously:

1. Two vertices of v, u, w are in S^i_t and the edge between these two vertices belongs to $E(G[S^i_t])$, or v, u, w are all in S^i_t and one or two edges of $(v, u), (v, w), (u, w)$ belong to $E(G[S^i_t])$.
2. v, u, w form a static triangle, $STA^i_{t+1}(v, u, w)$, in S^i_{t+1} on G_{t+1}.

For example, in Fig. 1, from G_1 to G_2, user b, f and g form a dynamic triangle of group S^1, and user h, j and l form a dynamic triangle of group S^2. From G_2 to G_3, user h, k and l form a dynamic triangle of group S^2.

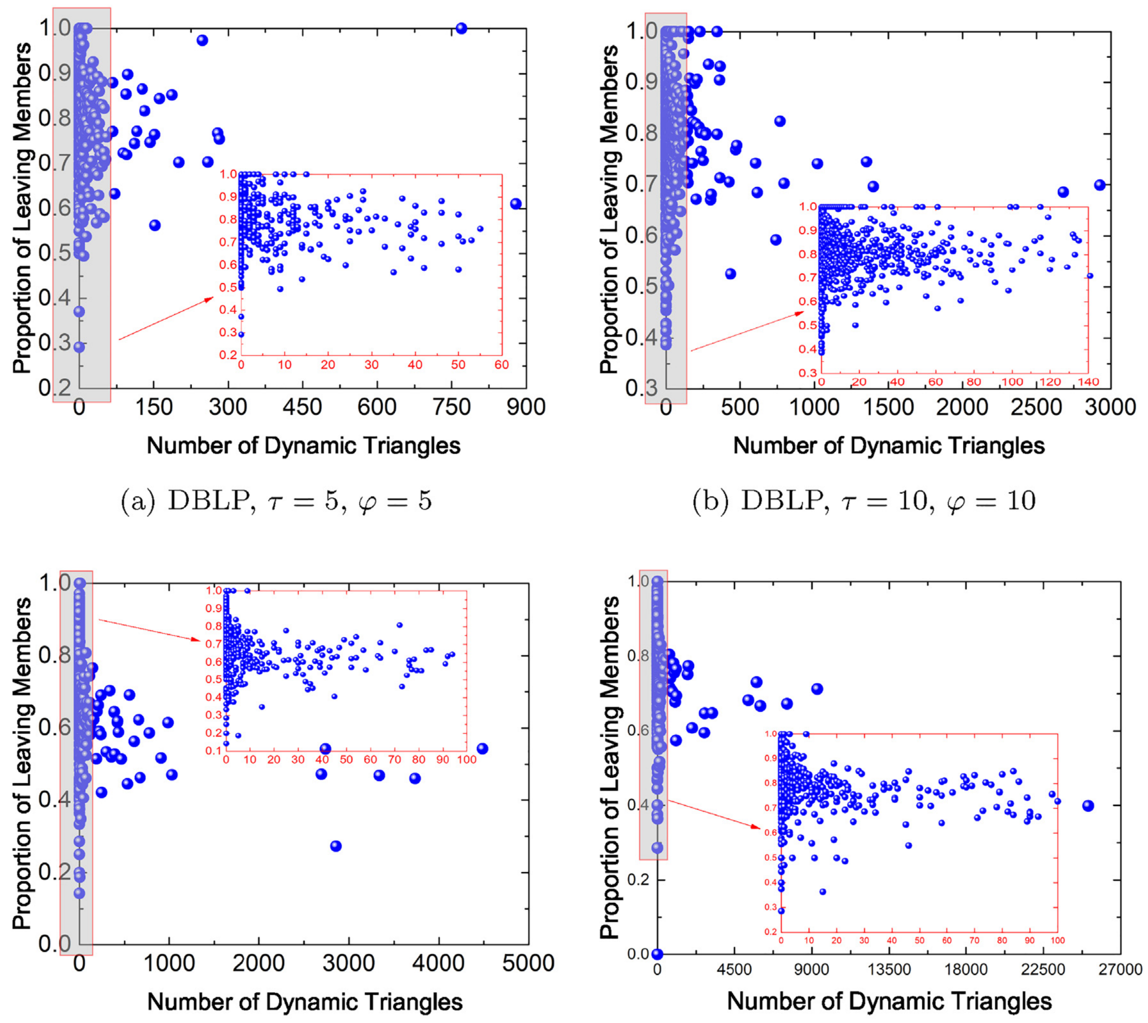
A dynamic triangle of a group between two contiguous snapshots indicates that the relationships among these three group members are continuously strengthened from t to $t+1$. Thus, the number of dynamic triangles of a group S^i from the timestamp t to timestamp $t+1$, denoted as β^i_{t+1}, shows the degree of *continuity* of S^i from t to $t+1$.

To the best of our knowledge, we are the first to utilize the number of dynamic triangles between each two snapshots in a group to measure the continuity of the group. Based on a lot of quantitative studies, group members' departure behaviors are actually related to the number of dynamic triangles of each group (i.e., continuity of each group). Fig. 2 presents some experimental results in two real-world dynamic networks DBLP and Brightkite. In each subfigure, a bule data point represents a group. As shown in all subfigures, the fewer dynamic triangles a group contains, the larger proportion of leaving members it has. That is, the members in a group which has a smaller number of dynamic triangles are more likely to leave the group. More details about the parameters τ and φ are left in Section 5.

Group Members' Departure. Given the current window $W_r : G_r, G_{r+1}, \ldots, G_{r+\tau-1}$ of a dynamic network $\mathscr{G} = (G_1, G_2, \ldots, G_t, \ldots)$ and an *activeness threshold* θ, we predict a member v would leave a group S^i at the ending timestamp $r + \tau - 1$ if his/her *group-combined activeness score* in the window W_r, denoted as $AS_r(v, S^i)$, is below θ. The group-combined activeness score of a user v in a group S^i is related to both the *user's activeness* in S^i in W_r, denoted as $UA_r(v, S^i)$, and the *group's activeness* in W_r, denoted as $GA_r(S^i)$. We leave the details of the definitions and computation of them in Section 4.

Group Members' Departure Prediction Problem. The group members' departure prediction problem in a dynamic network can be summarized as follows:

- Input: A dynamic network $\mathscr{G} = (G_1, G_2, \ldots, G_t, \ldots)$, the set of groups $\mathscr{S}_t$ on each snapshot $G_t = (V_t, E_t)$, the set of group-weight pairs $P_t(v)$ for each vertex $v \in V_t$, the label set $L_t(v, u)$ for each edge $(v, u) \in E_t$, a sliding window with the window size τ and the slide size μ, a time-dependent parameter λ, and an activeness threshold θ.
- Output: All pairs of user-groups (v, S^i) where $v \in \bigcup_{x=r}^{r+\tau-1} V_x$ and $S^i \in \bigcup_{x=r}^{r+\tau-1} \mathscr{S}_x$ such that $AS_r(v, S^i) < \theta$ at the ending time of each current window W_r when the window slides.



(a) DBLP, $\tau = 5$, $\varphi = 5$

(b) DBLP, $\tau = 10$, $\varphi = 10$

(c) Brightkite, $\tau = 5$, $\varphi = 5$

(d) Brightkite, $\tau = 10$, $\varphi = 10$

Fig. 2. Correlation Between the Number of Dynamic Triangles and Members' Departure Behaviors in Groups. (In each subfigure, each blue data point represents a group. In addition, for each subfigure, a zoomed view is provided for a specific region with lots of data points.)

4. Group members' departure prediction

In this section, we firstly give the definition of a group's activeness based on both its structure features and its members' activity information, and devise some strategies to compute a group's activeness efficiently in Section 4.1. Then we introduce the notion of a user's time-dependent activeness considering that a user's activeness is more relevant to his/her more recent activities in Section 4.2. After that, we define a user's group-combined activeness score in Section 4.3. Finally, we present the group members' departure prediction algorithm in Section 4.4.

4.1. Group's activeness

To define the activeness of a group, both its structure features and group members' activities should be considered. Firstly, if a group contained more static triangles on the historical snapshots, the relationships among group members were stabler during the period and the group would have a relatively lower probability to disband now and in the future. Secondly, if there existed some dynamic triangles in a group between each two contiguous historical snapshots, it means that the group has attracted new users to join in or some members have developed new relationships in the group so that the group has the ability to keep its vitality. Thirdly, if the members' participation frequency in a group was always high during the period of several timestamps (e.g., the number of messages sent in an online chat group remained stable or was increasing), the group is more likely to have great significance in reality and not easy to be unraveled.

Therefore, given a window $W_r : G_r, G_{r+1}, \ldots, G_{r+\tau-1}$ of a dynamic network $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$, we firstly need to extract the following information for the definition of the *activeness of a group* S^i:

1. Structure Features for Stability: the number of static triangles of S^i on each snapshot in W_r, i.e., α^i_x $(r \leqslant x \leqslant r + \tau - 1)$.
2. Structure Features for Continuity: the number of dynamic triangles of S^i between each two contiguous snapshots in W_r, i.e., β^i_x $(r + 1 \leqslant x \leqslant r + \tau - 1)$.
3. Members' Participation Frequency Information: the sum of the weights of all vertices in S^i on each snapshot in W_r, i.e., $\sigma^i_x = \sum_{v \in S^i_x} w^i_x(v)$ $(r \leqslant x \leqslant r + \tau - 1)$.

The number of static triangles of S^i_t can be computed by the algorithm *forward* in [17] with the time complexity of $O((e_2)^{1.5})$ and the space complexity of $O(e_2)$ where e_2 is the number of edges in $E(G[S^i_t])$. Next, we present an efficient algorithm for counting dynamic triangles of a group between two contiguous snapshots.

Algorithm 1: Dynamic Triangle Counting Algorithm

```
Input  : The adjacency list representation of $G[S^i_t]$ and $G[S^i_{t+1}]$
Output: $\beta^i_{t+1}$, the total number of dynamic triangles of $S^i$ from $t$ to
        $t+1$
1:  $\beta^i_{t+1} \leftarrow 0$;
2:  for each vertex $v$ in $S^i_t$ in increasing order of $ID(v)$ do
3:      if $v \in S^i_{t+1}$ then
4:          $ComNei \leftarrow N^i_t(v) \bigcap N^i_{t+1}(v)$;
5:          for each vertex $u \in ComNei$ with $ID(u) > ID(v)$ do
6:              $A \leftarrow N^i_t(v) \bigcap N^i_t(u)$;
7:              $B \leftarrow N^i_{t+1}(v) \bigcap N^i_{t+1}(u)$;
8:              for each $w \in B$ and $w \notin A$ do
9:                  $\beta^i_{t+1} \leftarrow \beta^i_{t+1} + 1$;
10: return $\beta^i_{t+1}$;
```

Dynamic Triangle Counting. Algorithm 1 illustrates how to count the number of dynamic triangles of a group S^i from the timestamp t to $t + 1$. Note that each vertex v in V_t is given an exclusive ID, denoted as $ID(v)$ ($ID(v)$ is an integer greater than 0), and the adjacency lists are constructed in increasing order of the vertices' IDs.

Firstly, the algorithm finds all edges coexisting in $E(G[S^i_t])$ and $E(G[S^i_{t+1}])$ (lines 2–4). Then for each common edge, it computes the common neighbors of its two end points in S^i_t, stored in set A, and those in S^i_{t+1}, stored in B respectively (lines 5–7). After that, each vertex, which is in B but not in A, with two end points of the common edge forms a dynamic triangle of the group S^i from the timestamp t to $t + 1$ (lines 8–9). Note that an auxiliary structure is needed to store already-obtained dynamic triangles every time when finishing executing line 9 to avoid repeating counting.

The time complexity of Algorithm 1 is $O(d'e_3)$ where d' is the maximum degree of all the vertices in $S^i_t \bigcup S^i_{t+1}$ and $e_3 = max\{|E(G[S^i_t])|, |E(G[S^i_{t+1}])|\}$. The space complexity is $O(e_3 + d')$.

After acquiring all the information mentioned above, we utilize a voting rule to record the variation trend of a group's stability as well as that of its members' participation frequency in each current window.

Voting Rule. To reflect the variation trend of a variable, the characteristic curve is commonly used. However, in real-world dynamic networks, it is premium to plot the curves of both the number of static triangles (α^i_t) and the sum of all vertices' weights (σ^i_t) for a large number of groups in each current window. Therefore, we introduce *a voting rule*, which is low-cost compared with characteristic curves, to describe the variation trend of a variable.

Firstly, in each current window $W_r : G_r, G_{r+1}, \ldots, G_{r+\tau-1}$, for each group S^i, we denote the difference of the total number of static triangles of S^i between two contiguous snapshots as $\Delta^i_{x+1}(\alpha) = \alpha^i_{x+1} - \alpha^i_x$ $(r \leqslant x \leqslant r + \tau - 2)$, and the difference of the sum of the weights of all vertices in S^i between two contiguous snapshots as $\Delta^i_{x+1}(\sigma) = \sigma^i_{x+1} - \sigma^i_x$

($r \leqslant x \leqslant r + \tau - 2$). Then the voting rule for recording the variation trends of both variables for the group S^i in W_r is presented as follows:

1) When $x = r$

$$Vote^i_{x+1}(*) = \begin{cases} +1, & if \quad \Delta^i_{x+1}(*) \geqslant 0 \\ -1, & if \quad \Delta^i_{x+1}(*) < 0 \end{cases}$$

2) When $x > r$

$$Vote^i_{x+1}(*) = \begin{cases} +1, & if \quad \Delta^i_{x+1}(*) \geqslant 0, \Delta^i_x(*) < 0 \\ -1, & if \quad \Delta^i_{x+1}(*) < 0, \Delta^i_x(*) \geqslant 0 \\ Vote^i_x(*) + 1, & if \quad \Delta^i_{x+1}(*) \geqslant 0, \Delta^i_x(*) \geqslant 0 \\ Vote^i_x(*) - 1, & if \quad \Delta^i_{x+1}(*) < 0, \Delta^i_x(*) < 0 \end{cases}$$

where $(*)$ represents (α) or (σ).

The rule is reasonable because the voting number will be larger if the variable value in the window keeps increasing or remains unchanged over time, and it will be smaller if the variable value in the window keeps decreasing.

For example, in Fig. 1, $\alpha^1_1 = 1, \alpha^1_2 = 1, \alpha^1_3 = 0, \alpha^1_4 = 0$, then $\Delta^1_2(\alpha) = 0, \Delta^1_3(\alpha) = -1, \Delta^1_4(\alpha) = 0$. Therefore, $Vote^1_2(\alpha) = 1$, $Vote^1_3(\alpha) = -1, Vote^1_4(\alpha) = 1$. Furthermore, $\sigma^1_1 = 7, \sigma^1_2 = 5, \sigma^1_3 = 3, \sigma^1_4 = 6$, then $\Delta^1_2(\sigma) = -2, \Delta^1_3(\sigma) = -2, \Delta^1_4(\sigma) = 3$. Thus, $Vote^1_2(\sigma) = -1, Vote^1_3(\sigma) = -2, Vote^1_4(\sigma) = 1$.

Setting the window size $\tau = 10$, we apply the voting rule to the groups in the DBLP network. Fig. 3 shows the characteristic curves and the voting rule curves of the variables (α^i_x and σ^i_x) for a group S^i in a window W_r. Intuitively, in each subfigure, two kinds of curves are similar. When the variable value increases (decreases resp.), the voting rule curve goes up (down resp.). It indicates that the voting rule effectively records the variation trend of a variation.

Group's Activeness. Considering both structure features and members' activity information, we define the *activeness of a group* S^i in the current window W_r as follows:

$$GA_r(S^i) = [\sum_{x=r+1}^{r+\tau-1} (Vote^i_x(\alpha) + Vote^i_x(\sigma)) + h(\tau)] \times [(\sum_{x=r+1}^{r+\tau-1} \beta^i_x) \times \frac{\eta^i_r}{\tau - 1} + 1] \tag{1}$$

where $h(\tau)$ is a function of the window size τ to avoid the first factor being 0 (e.g. $h(\tau) = \tau(\tau - 1)$), and η^i_r is the number of timestamps when $\beta^i_x \neq 0$ ($r + 1 \leqslant x \leqslant r + \tau - 1$) in W_r. Similarly, it adds 1 in the second factor to avoid the second factor being 0.

The definition is reasonable because a group is more active if (1) the number of static triangles it contains on each snapshot keeps unchanged or increasing over time; (2) the sum of weights of all nodes in it on each snapshot keeps unchanged or increasing over time; (3) it contains more dynamic triangles between each two contiguous snapshots of the window.

4.2. User's time-dependent activeness

Obviously, a user's activity information in a group directly reflects his/her own activeness in the group. Based on our further observations, the activeness of a user in a group is more related to his/her more recent activities during a period of time. Taking an online chat group as an example, a user might be more active and have a less probability to leave the group if he/she sent messages at the more recent timestamps. Therefore, we introduce the definition of the *time-dependent activeness of a user* so that the time factor can be integrated into the activeness definition.

For each current window $W_r : G_r, G_{r+1}, \ldots, G_{r+\tau-1}$ of a dynamic network $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$, the time-dependent activeness of a user v in a group S^i is defined as follows:

$$UA_r(v, S^i) = \sum_{x=r}^{r+\tau-1} f_\lambda(x) \times w^i_x(v) \tag{2}$$

where $w^i_x(v)$ is the weight of the user v in the group S^i_x ($r \leqslant x \leqslant r + \tau - 1$), and $f_\lambda(x)$ is a function of the timestamp x with a time-dependent parameter λ. Note that $f_\lambda(x)$ must satisfy three conditions: (1) $f_\lambda(x)$ is always positive; (2) $f_\lambda(x)$ is bigger when x increases; (3) the slope of $f_\lambda(x)$ is bigger when x increases. For example, $f_\lambda(x) = e^{\lambda(x-r)}$ where e is the base of the natural logarithm ($e \approx 2.718$). Owing to the function $f_\lambda(x)$, the more recent weights of a user play more important roles in the final result of the user's own activeness.

4.3. User's group-combined activeness score

After acquiring a user's own activeness in a group and the group's activeness, we introduce the definition of a *user's group-combined activeness score* to predict whether the user would leave the group. Intuitively, the lower group-combined activeness score a user has, the more possibly he/she would leave the group. For each current window W_r, the group-combined activeness score of a user v in a group S^i is defined as follows:

$$AS_r(v, S^i) = UA_r(v, S^i) \times GA_r(S^i) \tag{3}$$

where $UA_r(v, S^i)$ is the activeness of the user v in S^i in W_r and $GA_r(S^i)$ is the activeness of the group S^i in W_r.

According to the definition, a user would have a relatively high probability to leave a group if the user's own activeness in the group, the group's activeness or both of them are low. Take an online chat group as an example. When the first factor is low, it means that the user v kept silent at the most recent timestamps, and he/she might lose interest in the group and want to leave. When the second factor is low, the relationships among the group members were relatively unstable and weakly developed over time, and the members chatted infrequently in the group, it indicates that the group might lose its attraction and the members would leave the group one after another.

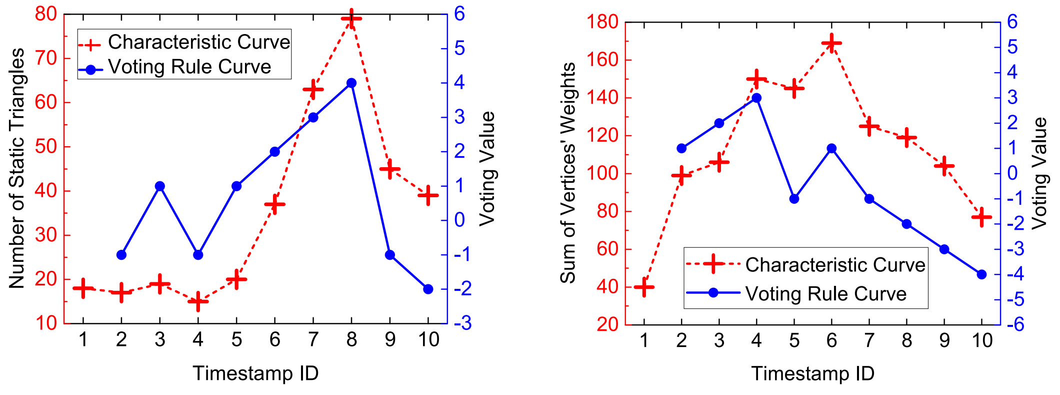
Mentioned in Section 3, a user is predicted to leave a group if his/her group-combined activeness score is below the activeness threshold θ. Therefore, the number of the final predicted leaving user-group pairs depends on θ. We leave the details about the setting strategy of θ in Section 5.

Algorithm 2: Group Members' Departure Prediction Algorithm

```
Input  : $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$, $\mathcal{S}_t$ on each $G_t = (V_t, E_t)$, $P_t(v)$ for
         each vertex $v \in V_t$, $L_t(v, u)$ for each edge $(v, u) \in E_t$, $\tau$, $\mu$, $\lambda$,
         and $\theta$
Output: All pairs of user-groups $(v, S^i)$ such that $AS_r(v, S^i) < \theta$
1:  $t \leftarrow 1$, $W_r.start \leftarrow t$, $W_r.end \leftarrow W_r.start + \tau - 1$;
2:  for each snapshot $G_t$ do
3:      for each group $S_t^i \in G_t$ do
4:          compute $\sigma_t^i$ and $\alpha_t^i$;
5:          if $t \neq W_r.start$ then
6:              compute $\beta_t^i$ invoking Algorithm 1;
7:              compute $\Delta_t^i(\alpha)$ and $\Delta_t^i(\sigma)$;
8:              compute $Vote_t^i(\alpha)$ and $Vote_t^i(\sigma)$ using the voting rule;
9:      if $t = W_r.end$ then
10:         for each group $S^i \in (\bigcup_{x=W_r.start}^{W_r.end} \mathcal{S}_x)$ do
11:             compute $GA_r(S^i)$ using Eq. (1);
12:             for each vertex $v \in S^i$ do
13:                 compute $UA_r(v, S^i)$ using Eq. (2);
14:                 compute $AS_r(v, S^i)$ using Eq. (3);
15:                 output $(v, S^i)$ with $AS_r(v, S^i) < \theta$;
16:         $W_r.start \leftarrow W_r.start + \mu$; $W_r.end \leftarrow W_r.start + \tau - 1$;
17:     $t \leftarrow t + 1$;
```

4.4. Group members' departure prediction algorithm

Algorithm 2 illustrates how to continuously predict the group members' departure behaviors in a dynamic information network with a sliding window model. In each current window, for each group, the algorithm computes the sum of the weights of its vertices (σ_t^i) and the number of its static triangles (α_t^i) on each snapshot (line 4). It also obtains the number of its dynamic triangles (β_t^i) between each two contiguous snapshots with Algorithm 1 (line 6) and records the variation trends of both α_t^i and σ_t^i in the current window with the voting rule (lines 7–8). Then, at the ending timestamp of each win-



(a) Variation Trend of the Number of Static Triangles of a Group in DBLP (b) Variation Trend of the Sum of Vertices' Weights of a Group in DBLP

Fig. 3. Illustration of Voting Rule's Effect. (Voting Rule Curves are in blue, while Characteristic Curves are in red.)

dow, for each group, it computes the activeness of the group according to Eq. (1) (line 11), and for each user in the group, it computes the user's time-dependent activeness and obtains its group-combined activeness score according to Eqs. (2) and (3) respectively (lines 12–14). After that, a user whose group-combined activeness score is below the activeness threshold θ would be predicted to leave the corresponding group (line 15). Finally, the window slides (line 16) and new prediction results of the next window will be outputted (going back to line 2).

The time complexity of Algorithm 2 for each window is $O(\tau m((e_4)^{1.5} + de_4 + n))$ where τ is the window size, m is the number of groups in the window, e_4 is the maximum number of edges of each group in the window, n is the maximum number of nodes of each group in the window, and d is the maximum degree of each node in the window. Since a amount of raw data of each two contiguous snapshots can be deleted every time when finishing executing line 6, the memory usage of Algorithm 2 can be tolerant to handle a large number of snapshots of a window in real-world dynamic networks. The space complexity of Algorithm 2 for each window is $O(m(e_4 + d + \tau n))$.

5. Performance evaluation

We have conducted extensive experiments to evaluate our proposed algorithms. The experimental results are reported in this section.

5.1. Experimental setup

We introduce the experimental setup in this subsection, which includes the algorithms, datasets, evaluation parameters and metrics, and our experimental environments.

Algorithms. We implement our Group Members' Departure Prediction algorithm (GMDP) and two state-of-the-art competitors, the Core Number-based method (CoreN) proposed by Malliaros et al. [21], and the method based on neighbors Outside and In groups (Out-In) proposed by Palla et al. [24], for experimental evaluation.

- **CoreN:** At the ending timestamp of each current window W_r, the core number of a user in a group less than a given integer would leave the group. The time complexity of CoreN is $O(m'e_1)$ where m' is the number of all groups on the snapshot $G_{r+\tau-1}$ and e_1 is the maximal number of edges of each group on $G_{r+\tau-1}$. More details can be seen in Section 3 in the part of the definition of *Core Number*.
- **Out-In:** For a user in a group, the number of his/her neighbors outside (in resp.) the group is defined as *Nout* (*Nin* resp.). Then at the ending timestamp of W_r, the users in a group with large $\frac{Nout}{Nout+Nin}$ would leave the group. The time complexity of Out-In is $O(m'n_{max}d_{max})$ where m' has the same definition mentioned above, n_{max} is the maximum number of nodes of each group on the snapshot $G_{r+\tau-1}$, and d_{max} is the maximum degree of each node on $G_{r+\tau-1}$.
- **GMDP:** In our GMDP algorithm, for all users appearing on any snapshot of W_r, we predict whether they would leave the groups at the ending timestamp of W_r.

Setting the slide size μ as 1, when the window slides, by respectively performing CoreN, Out-In and GMDP in each current window, we can continuously obtain the predicted results of each algorithm at each timestamp.

Datasets. Four real datasets are used in our experiments.

The first two datasets we used are the Brightkite dataset[4] and Gowalla dataset.[5] Brightkite and Gowalla are both mobile internet applications. Once a user checked in at a place, they would post notes and photos of the place and other users could comment on those posts. Brightkite contains $4,491,143$ checkins of $58,228$ users from April 2008 to October 2010 and $214,078$ friendships among these users. Gowalla consists of $6,442,890$ check-ins of $196,591$ users from February 2009 to October 2010 and $950,327$ friendships among these users. In both Brightkite and Gowalla networks, we consider each user as a vertex. Links between users represent their friendship. We manually divide the land into a number of geographic regions, each of which is 7200 km^2. The users checking in at the same region form a group. In Brightkite, each snapshot contains the data of a month and has more than 1000 groups. In Gowalla, each snapshot contains the data of a week and has more than 900 groups.

The third dataset is the US Patent dataset.[6] It consists of 3 million US patents granted from January 1963 to December 1999. In Patent, we consider each inventor as a vertex. There is an edge between a pair of inventors if they have at least one co-operative invention. The inventors who invent the same class of patents form a group. There are 6 groups in Patent: (1) Chemical Group, (2) Computers and Communications Group, (3) Drugs and Medical Group, (4) Electrical and Electronic Group, (5) Mechanical Group, (6) Others Group. Each snapshot contains the data of one year.

The fourth dataset is the online bibliographic database DBLP.[7] It includes the published papers of more than 1.5 million authors from 1960 to 2019. In DBLP, the authors are the vertices and the co-operations among them are the edges. The authors publishing papers in the same conference form a group and there are more than 4700 groups. Each snapshot contains the data of a year.

Parameters. Due to lack of ground truth, motivated by the work in [32], we introduce a *ground truth assumption* which considers that a member really leaves a group at timestamp $W_r.end$ if he/she is absent from the group from timestamp $W_r.end + 1$ to $W_r.end + \varphi$. We call φ the *real size*. For example, in Brightkite, if a user hasn't check in at a region for contiguous φ months, we consider that he/she actually loses interest in the region for travel, shopping, etc. Similarly, in Patent, if an inventor hasn't invented the same class of patents for contiguous φ years, we consider that he/she actually abandons inventing this class of inventions. There exist a very small number of users who would come back after some time which is longer than φ. However, it happens very little and would not influence our performance evaluation.

We test the performance of GMDP, CoreN and Out-In by varying 4 parameters, namely the real size φ, the window size τ, the activeness threshold θ and the time-dependent parameter λ. Since each dataset has its own characteristics, we adopt different configurations for these parameters in different datasets. The ranges and the default values of the parameters for Brightkite, Patent and DBLP are shown in Table 2. Without otherwise stated, when varying a parameter, other parameters are set to their default values. Note that when setting $\theta = top\text{-}30\%$, it means that we set θ as the 30%-quantile of all users' group-combined scores in the current window.

In addition, to ensure the best performance of CoreN, we conduct a few parameter tests and determine that a user whose core number in a group is less than 2 would leave the group in Brightkite or Gowalla. And an inventor (author resp.) whose core number in a group is less than 3 (16 resp.) would leave the group in Patent (DBLP resp.).

For Out-In, it is even harder to decide the proportion of leaving users. Since the number of the users with $\frac{Nout}{Nout+Nin} \neq 0$ is quite small in Patent and DBLP, we consider that all of such users would leave. In Brightkite, by conducting a few parameter tests, we determine that the users whose $\frac{Nout}{Nout+Nin}$ rank at top 10% would leave considering both the precision and recall of the results. Similarly, in Gowalla, we respectively determine that the users with $\frac{Nout}{Nout+Nin}$ ranking at top 1% and top 30% would leave when comparing the performance of Out-In with other algorithms.

Metrics. For the current window W_r, we use DT to represent the set of really departing user-group pairs obtained by the ground truth assumption and DP to represent the set of predicted departing user-group pairs outputted by GMDP, CoreN or Out-In. Then the *precision* of each algorithm is $\frac{|DT \bigcap DP|}{|DP|}$, the *recall* is $\frac{|DT \bigcap DP|}{|DT|}$, and the *F1-Measure* is $\frac{2precision* \ recall}{precision+recall}$. In this paper, we measure these three metrics for GMDP, CoreN and Out-In to show the effectiveness and superiority of GMDP. We also measure the average running time of GMDP for each window in different datasets to show its efficiency.

Environments. All the experiments are implemented on a Microsoft Windows 10 machine with an Intel(R) Core i7-6700 CPU 3.4 GHz and 32 GB main memory. Programs are compiled by Microsoft Visual Studio 2017 with C++ language.

5.2. Effectiveness

Due to similar data characteristics, the variation trends of the precision, recall and F1-measure in Brightkite and Gowalla are alike when varying four parameters mentioned above respectively. Therefore, we only show the results in Brightkite, Patent and DBLP in this subsection. Note that all the results shown in Figs. 4 to 15 are the average values of several contiguous windows.

Varying Real Size. Figs. 4–6 show the precision, recall and F1-measure of GMDP, CoreN and Out-In when varying the real size φ. For Brightkite and DBLP, GMDP significantly outperforms CoreN and Out-In in terms of the precision, which indicates

[4] http://snap.stanford.edu/data/loc-brightkite.html
[5] http://snap.stanford.edu/data/loc-gowalla.html
[6] http://www.nber.org/patents/
[7] https://dblp.dagstuhl.de/xml/release/

Table 2
Parameter Configuration for Datasets.

Dataset	Parameter	Range	Default Value
Brightkite	φ	$\{3,5,7,9,11\}$	5
	τ	$\{3,5,7,9,11\}$	5
	θ	$\{top\text{-}1\%, top\text{-}10\%, top\text{-}20\%, top\text{-}30\%, top\text{-}40\%\}$	*top*-30%
	λ	$\{0.01, 0.1, 1, 3, 5, 7\}$	5
Patent	φ	$\{3,4,5,6,7\}$	5
	τ	$\{3,4,5,6,7\}$	5
	θ	$\{top\text{-}0.1\%, top\text{-}1\%, top\text{-}10\%, top\text{-}20\%, top\text{-}30\%, top\text{-}40\%\}$	*top*-30%
	λ	$\{0.001, 0.01, 0.1, 1, 3, 5\}$	0.1
DBLP	φ	$\{3,4,5,6,7\}$	5
	τ	$\{3,4,5,6,7\}$	5
	θ	$\{top\text{-}0.1\%, top\text{-}1\%, top\text{-}10\%, top\text{-}20\%, top\text{-}30\%, top\text{-}40\%\}$	*top*-30%
	λ	$\{0.1, 1, 3, 5, 7, 9\}$	3

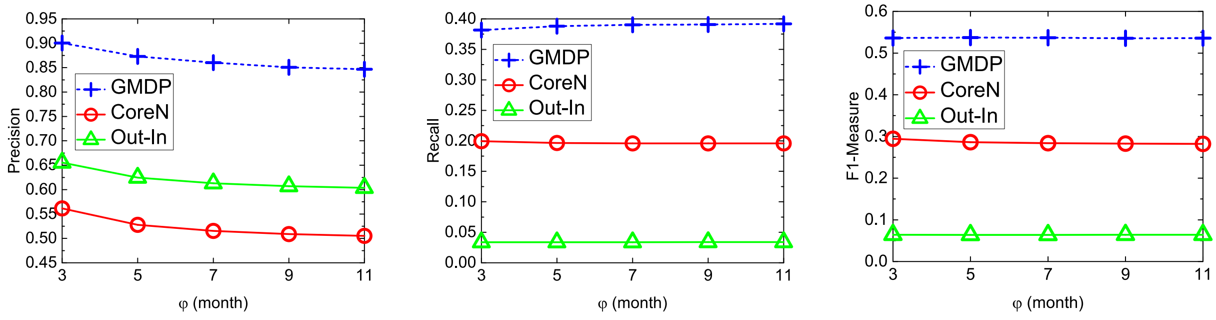


Fig. 4. Brightkite, varying Real Size φ.

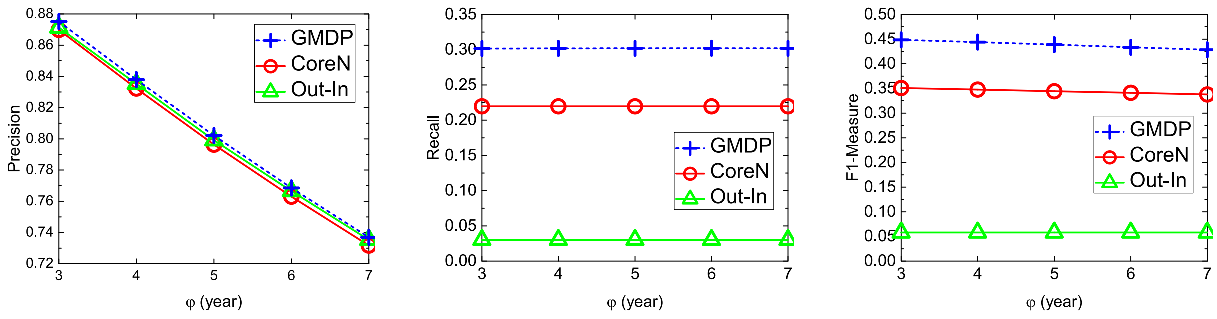


Fig. 5. Patent, varying Real Size φ.

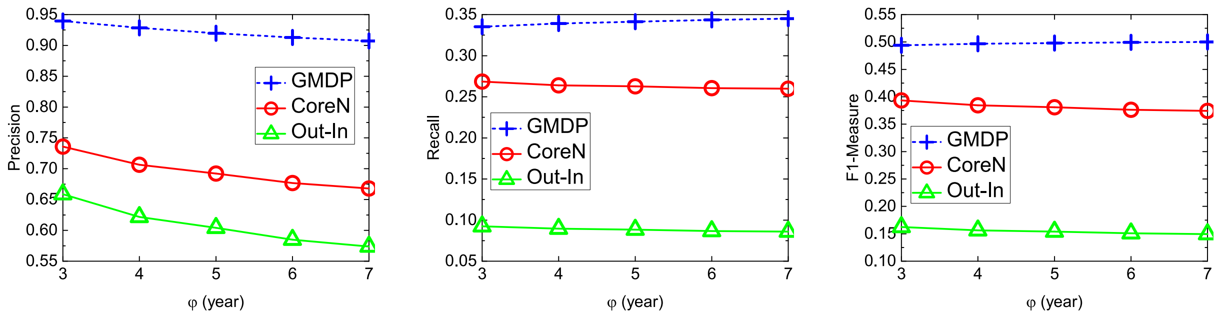


Fig. 6. DBLP, varying Real Size φ.

that GMDP really improve the prediction effect in practical applications. For Patent, with respect to the precision, GMDP is only a little superior to CoreN and Out-In due to the characteristics of Patent itself. Specifically, in Patent, most inventors in the current window W_r won't invent the same class of patents from timestamp $W_r.end + 1$ to $W_r.end + \varphi$. Based on the ground truth assumption, the majority (about 80%) of all inventors are put into *DT*. Recall that CoreN and Out-In are both neighbor-based methods, so they both might lead to precision loss for low-degree nodes which have little neighborhood information available for prediction. However, in Patent, since the number of really leaving inventors (i.e., $|DT|$) is too large, it happened that most of low-degree nodes are included in *DT*. Moreover, since the number of inventors who keep inventing the same class of patents continuously for several years is relatively small, for GMDP, the effects of the voting rule, the groups' continuity measurement and the users' time-dependent activeness measurement all degrade. As a result, as shown in Fig. 5, the precision (i.e., $\frac{|DT \bigcap DP|}{|DP|}$) of GMDP, CoreN and Out-In are similar. That is, in Patent, these three algorithms have similar prediction effect. When the real size grows, the precision of GMDP, CoreN and Out-In in three datasets are all decreasing. It is because that a bigger real size means a stricter ground truth assumption so that the number of the really departing user-group pairs, $|DT|$, decreases. In Patent, the precision of GMDP, CoreN and Out-In decrease similarly when φ increases for the following two reasons. Firstly, as explained above, for the same φ, these three algorithms have similar precision (i.e., $\frac{|DT \bigcap DP|}{|DP|}$). Secondly, for these three algorithms, φ only influences *DT*, while *DP* doesn't change with φ. Consequently, when φ increases, the precision of these three algorithms decrease similarly in Patent as shown in Fig. 5.

For three datasets, the recall of GMDP is much higher than that of CoreN and Out-In, since GMDP predicts for all users appearing in the window whether they would leave at the ending timestamp while CoreN and Out-In predict only for users appearing on the last snapshot of the window, that is, GMDP can find more departing users than CoreN and Out-In.

In addition, as for F1-Measure, GMDP also performs better than CoreN and Out-In in three datasets due to the superiority on both the precision and recall.

In a word, by varying φ, it indicates that even in a strict ground truth assumption (when φ is large), GMDP is still more effective and superior to both CoreN and Out-In.

Varying Window Size. Figs. 7–9 compare the performance of GMDP, CoreN and Out-In by varying the window size τ. For Brightkite and DBLP, when the window size increases, the precision of GMDP increases as well. Its precision keeps over than 90% with the window size no less than 7 (4 resp.) for Brightkite (DBLP resp.), which indicates that the predicted results of GMDP could be satisfactory as long as enough historical information is available. However, the precision of CoreN is only 52.79% (69.21%) and that of Out-In is 62.47% (60.42% resp.) for Brightkite (DBLP resp.), far lower than that of GMDP. For Brightkite, as the window size grows, there are more user-group pairs waiting to be predicted at the ending timestamp of the window, so the recall of three algorithms decrease. But the recall of CoreN and Out-In decrease more rapidly than that of GMDP since CoreN and Out-In only predict the departure behaviors for the users on the last snapshot. The F1-measure of GMDP firstly increases and then slowly decreases while that of CoreN and Out-In keep decreasing as the window size grows. For DBLP, when the window size increases, the recall of CoreN and Out-In decline due to the same reason as Brightkite. However, the recall of GMDP keeps stable, which shows that GMDP maintains good performance even though more user-group pairs need to be predicted in a window. The F1-measure of GMDP remains unchanged while that of CoreN and Out-In drop as the window size grows.

For Patent, the precision of GMDP firstly decreases and then increases as the window size grows, but always higher than that of CoreN and Out-In. Since the number of inventors who keep inventing the same class of patents continuously for several years is relatively small, the effects of the voting rule, the groups' continuity measurement and the users' time-dependent activeness measurement all degrade no matter how large the window size is. Therefore, GMDP is only a little superior to CoreN and Out-In as for the precision. The recall of GMDP is almost unchanged while that of CoreN and out-In are decreasing rapidly when the window size increases. The recall of GMDP is higher than that of CoreN unless the window size is 3. It is because that in Patent, when the window size is 3, the number of the users on the first two snapshots but not on the last snapshot is really small. When the window size further increases, the number of users on other snapshots but not on the last snapshot grows rapidly, so CoreN misses predicting more users' departure behaviors and the recall of CoreN drops a lot. The F1-measure of three algorithms have the same variation trend as the recall of them when the window size increases.

All in all, by varying τ, it reflects that even with a small τ, GMDP outperforms CoreN and Out-In as for the precision. It also shows that the effect of GMDP gets better if τ is larger. However, there is no need to set τ as large as possible since moderate τ already can return sufficiently satisfying results.

Varying Activeness Threshold. Figs. 10–12 depict the impact of the activeness threshold θ on the precision, recall and F1-measure of GMDP. The variation trends of these metrics for three datasets are similar when the activeness threshold increases.

Firstly, the precision of GMDP gradually decreases when θ increases from *top*-10% to *top*-40% in Brightkite (from *top*-1% to *top*-40% in Patent). And in DBLP, the precision of GMDP gradually slides when θ grows. It indicates that the definition of a user's group-combined score is reasonable, that is, the user with a lower group-combined score has a relatively higher probability to leave the group. However, when θ is very small (like *top*-1% in Brightkite and *top*-0.1% in Patent) so that only tens to hundreds of user-group pairs are outputted as the final predicted results, the precision of GMDP in both datasets are not as good as we expect. The reason is that a group might have very low activeness and then all users in it have lower group-

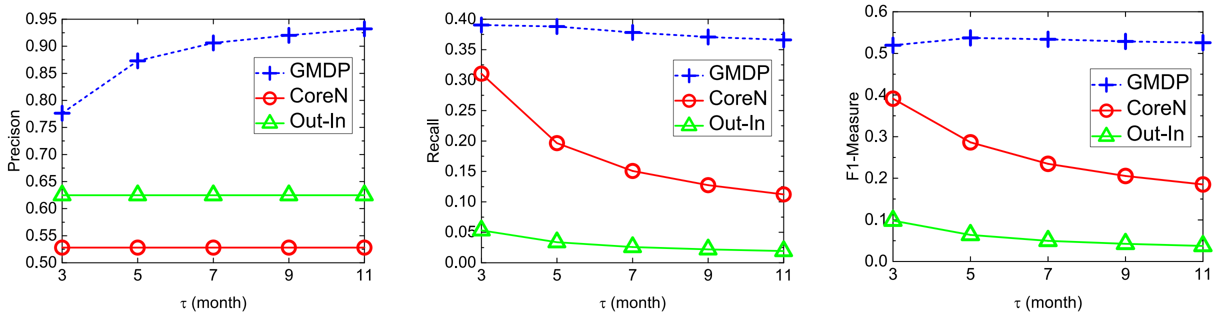


Fig. 7. Brightkite, varying Window Size τ.

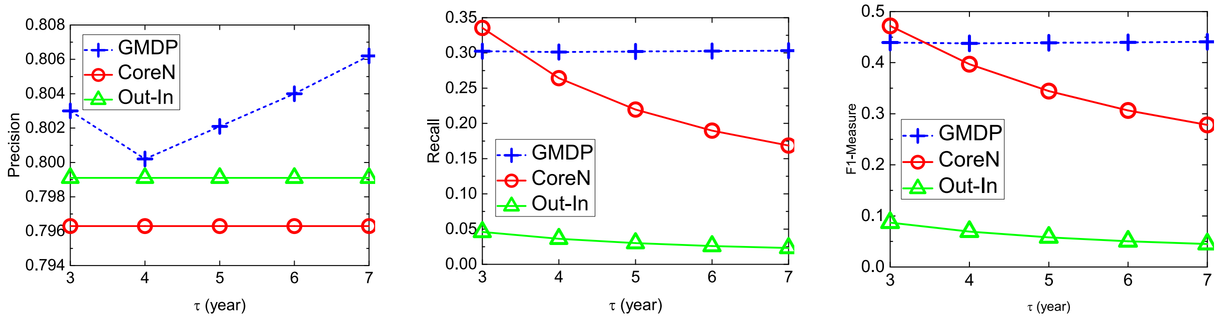


Fig. 8. Patent, varying Window Size τ.

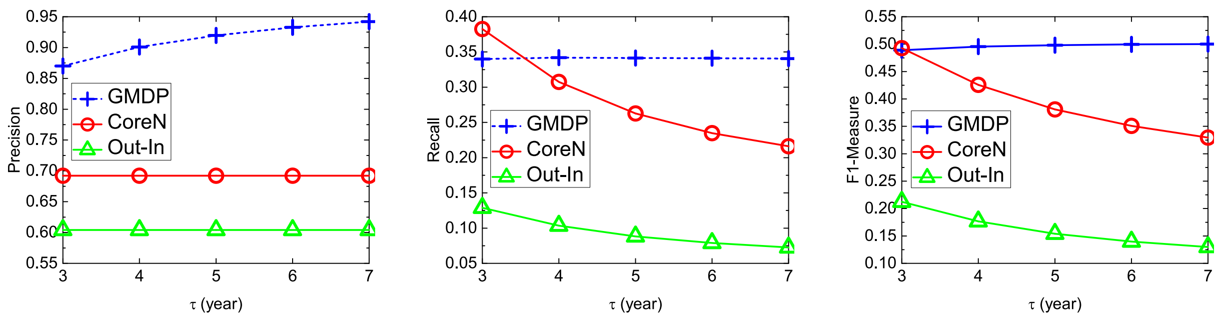


Fig. 9. DBLP, varying Window Size τ.

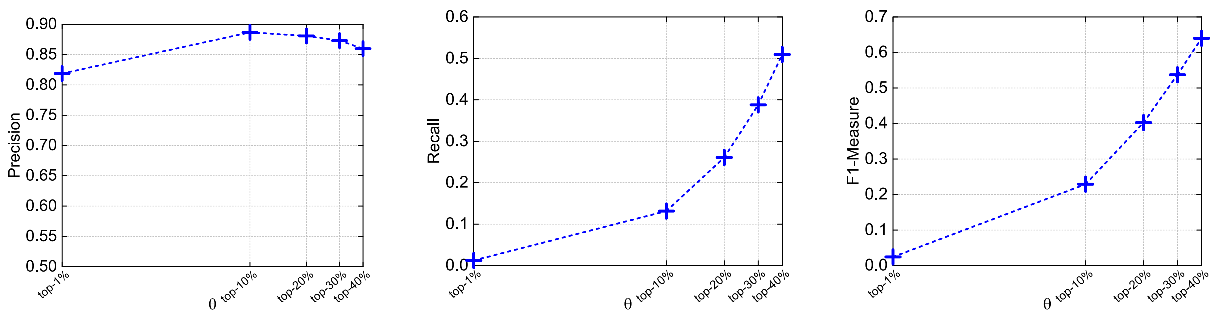


Fig. 10. Brightkite, varying Activeness Threshold θ.

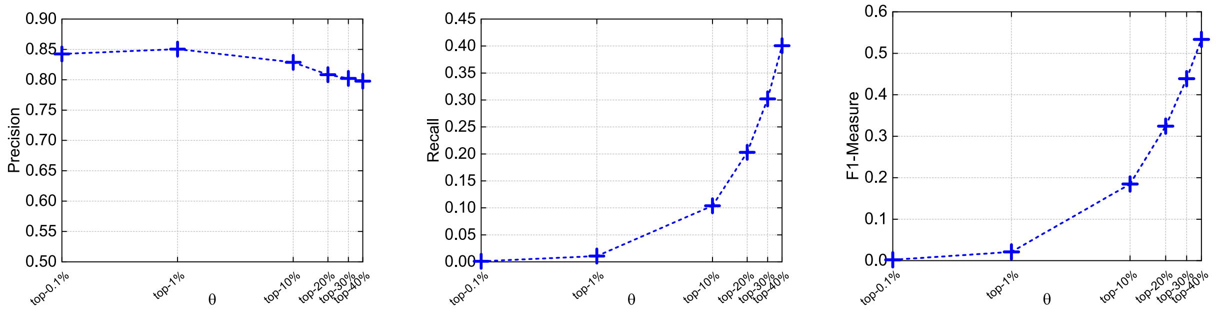


Fig. 11. Patent, varying Activeness Threshold θ.

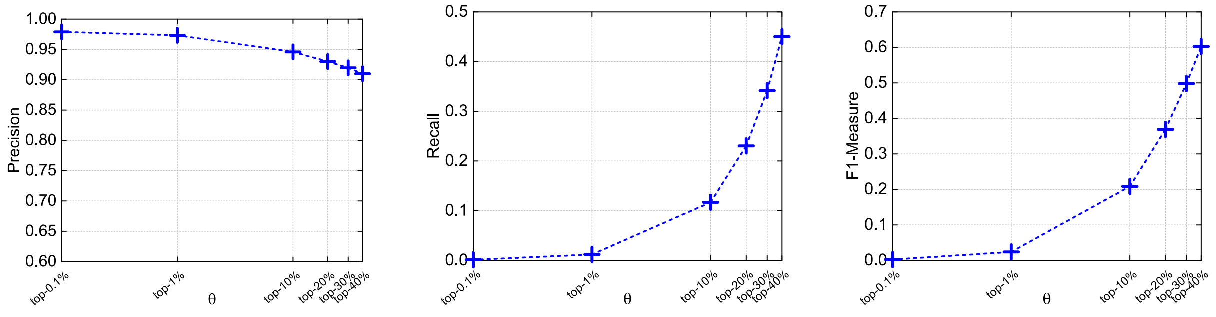


Fig. 12. DBLP, varying Activeness Threshold θ.

combined scores compared with users in other groups. As a result, all the users in this group are predicted to leave, which might reduce the precision.

Secondly, the recall of GMDP for three datasets increases as θ increases. For GMDP, a user whose group-combined activeness score is below θ would be predicted to leave the corresponding group. Thus, θ determines how many user-group pairs would be outputted as the prediction results after running GMDP. Specifically, small θ means a small number of outputted user-group pairs, while large θ means a large number of outputted user-group pairs. When varying $\theta, |DT|$ doesn't change. However, for GMDP, $|DP|$ is increasing when θ grows. Thus, the recall (i.e., $\frac{|DT \bigcap DP|}{|DT|}$) of GMDP increases dramatically as θ increases in Figs. 10–12. Actually, according to the ground truth assumption, when $\varphi = 5$, the really departing user-group pairs make up about $60\%, 80\%$ and 80% of the whole user-group pairs in Brightkite, Patent and DBLP respectively. Thus, more users' departure behaviors are rightly predicted by GMDP when increasing θ. It just indicates that GMDP is quite reasonable and effective for group members' departure prediction.

Thirdly, when θ grows, the F1-Measure of GMDP for three datasets increases as well. As shown in Figs. 10–12, like the recall, the F1-Measure is also so different to θ. It is because that the F1-Measure is $\frac{2precision* \ recall}{precision+recall}$, and the recall changes more dramatically than the precision when θ changes. Consequently, the F1-Measure has similar increasing trend as the recall.

When φ and τ are set to their default values, the precision of CoreN are $52.79\%, 79.63\%$ and 69.21%, and that of Out-In are $62.47\%, 79.91\%$ and 60.42% in Brightkite, Patent and DBLP respectively. The recall of CoreN are $19.65\%, 21.97\%$ and 26.28%, and that of Out-In are $3.37\%, 3.01\%$ and 8.83% in Brightkite, Patent and DBLP respectively. When θ is small, in three datasets, although the recall of GMDP is lower than that of CoreN and Out-In, the precision of GMDP is much higher than that of CoreN and Out-In. In many practical applications, we pay more attention to the precision than to the recall of the predicted results, so GMDP is really useful. When θ is large, in Brightkite and DBLP, both the precision and the recall of GMDP are much higher than those of CoreN and Out-In, which shows the superiority of our algorithm compared with the other two competitors. In Patent, due to the characteristics of the dataset itself, the precision of GMDP is only a little higher than that of CoreN and a little smaller than that of Out-In when θ is *top*-40%. However, the recall of the GMDP is significantly higher than that of CoreN and Out-In.

In sum, GMDP is superior to CoreN and Out-In in terms of precision no matter that θ is small like *top*-1% or large like *top*-30%. Since the number of the final predicted leaving user-group pairs depends on θ, users can set the value of θ according to their own needs in different practical applications.

Varying Time-Dependent Parameter. Setting $f_\lambda(x) = e^{\lambda(x-r)}$ where r is the start timestamp of each current window W_r, we test the performance of GMDP with different values of λ for three datasets. As shown in Fig. 13, when $\lambda \in [0.01, 7]$, the precision of GMDP for Brightkite is always over 83% and the recall is about 37% (both much higher than those of the other

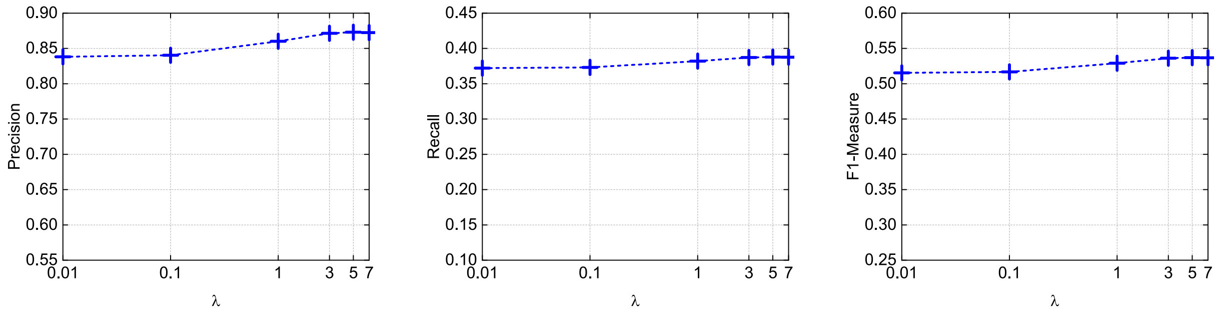


Fig. 13. Brightkite, varying Time-Dependent Parameter λ.

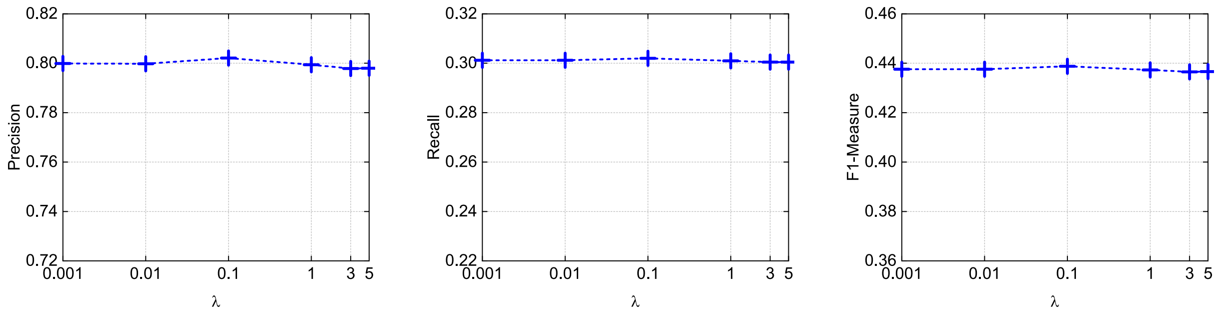


Fig. 14. Patent, varying Time-Dependent Parameter λ.

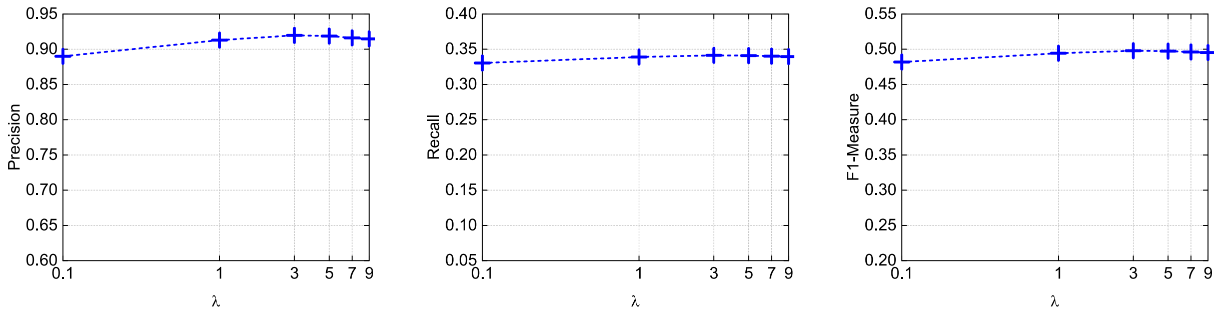


Fig. 15. DBLP, varying Time-Dependent Parameter λ.

two algorithms, CoreN and Out-In). When other parameters are set to their default values shown in Table 2, the best value of λ for Brightkite is 5. In Fig. 14, when $\lambda \in [0.001, 5]$ ($\lambda \in [0.001, 1]$ resp.), the precision of GMDP for Patent is always a little higher than that of CoreN (Out-In resp.). The recall of GMDP is much higher than that of CoreN and Out-In. When other parameters are set to their default values shown in Table 2, the best value of λ for Patent is 0.1. In Fig. 15, when $\lambda \in [0.1, 9]$, both the precision and recall of GMDP for DBLP are much higher than those of CoreN and Out-In. When other parameters are set to their default values shown in Table 2, the best value of λ for DBLP is 3.

In three datasets, the precision of GMDP firstly increases and then decreases when λ increases due to the characteristics of the function $f_\lambda(x) = e^{\lambda(x-r)}$ itself. When $\lambda(x - r)$ is too small or too large, the slope of $f_\lambda(x)$ is also too small or too large, so the effect of the user's time-dependent activeness measurement on the user's group-combined score degrades. According to the experimental results, there is no need to find the best λ for each dataset respectively since the results are satisfactory enough when λ is in a small interval near 1. Thus, we recommend users to set λ as 1 in practical applications to avoid spending lots of time searching for the best λ. In addition, labeling the really departing user-group pairs is usually premium and even impossible in reality so that the best λ can't be learned from historical data. It is a good choice to sacrifice a little precision to save lots of running time in large-scale real-world datasets.

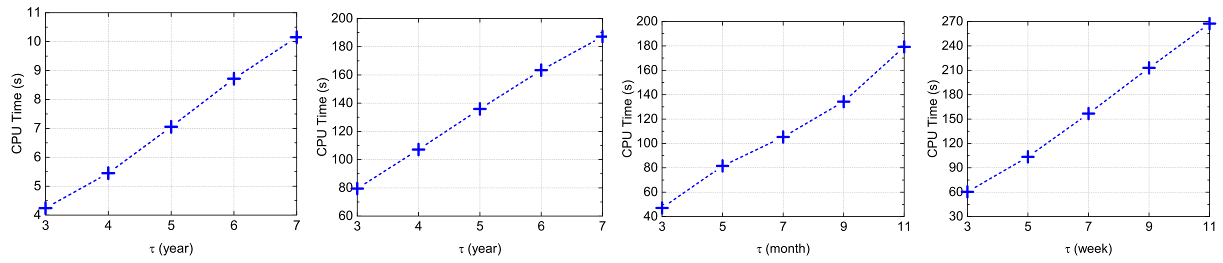
5.3. Efficiency

Fig. 16 shows the average running time for each window when performing GMDP in four datasets by varying the window size τ. For Patent, Brightkite and DBLP, other parameters are set to their default values shown in Table 2. For Gowalla, we set $\varphi = 12, \lambda = 5, \theta = top\text{-}30\%$. The running time is increasing when τ grows. When τ reaches 7, the running time of GMDP for Patent is only about 10 s in Fig. 16a but both the precision and the recall of GMDP are higher than those of CoreN and Out-In in Fig. 8 (the recall of GMDP is about 2 times as high as that of CoreN, 13 times as high as that of Out-In). For DBLP, when τ is 7, GMDP costs about 187 s in Fig. 16b, whereas it outperforms CoreN and Out-IN as for both the precision and recall in Fig. 9 (the precision of GMDP is over 94%, about 1.4 times as high as that of CoreN, 1.6 times as high as that of Out-In). When τ reaches 11, the running time of GMDP for Brightkite is about 179 s in Fig. 16c but the precision of GMDP is over 93% (about 1.8 times as high as that of CoreN, 1.5 times as high as that of Out-In) in Fig. 7. We have mentioned above that CoreN and Out-In are only conducted on the last snapshot of each window while GMDP is performed in the whole window, and the time complexity of CoreN and Out-In are $O(m'e_1)$ and $O(m'n_{max}d_{max})$ respectively, both lower than that of GMDP, $O(\tau m((e_4)^{1.5} + de_4 + n))$. Therefore, it makes no sense to compare GMDP's running time with CoreN and Out-In.

The experiments in this subsection illustrate that we can spend tolerable time to get better prediction results by GMDP in terms of both the precision and the recall. Since the prediction results of GMDP can be good enough only with a small window size, as for the time complexity of GMDP, τ can be seen as a constant in many practical applications. Table 3 shows some statistics of the four datasets and all the data in it is the average value of several windows. As shown in Fig. 16 and Table 3, the running time of DBLP, Brightkite and Gowalla are all longer than that of Patent because the number of groups of them are all larger than that of Patent. This motivates us that if the number of the groups is extremely large in some real-world networks, we can divide the groups into several parts to do parallel computing. The time saving strategies will be one of our future work.

5.4. Application examples in Gowalla

In Gowalla, the data of one week forms a snapshot. When setting $\tau = 5$ and $\varphi = 12$, it means that for all the users who had ever appeared at a region in the past five weeks, we predict who would never come back to the same region in the next twelve weeks. In Fig. 17, setting $\lambda = 1$ (according to the recommendation in the part of Varying Time-Dependent Parameter in SubSection 5.2), then it shows the precision, recall and F1-measure of weekly predicted results returned by GMDP, CoreN and Out-In for 10 contiguous weeks. When $\theta = top\text{-}30\%$, the precision of GMDP is 2 to 3 times as high as that of CoreN and the recall of GMDP is over 2 times as high as that of CoreN. When $\theta = top\text{-}1\%$, although the recall of GMDP is about 1.65% to 1.80%, the precision of GMDP is about 73.53% to 85.16% (about 3 to 4 times as high as that of CoreN). The precision of GMDP is about 2 times as high as that of Out-In no matter it determined that the users with $\frac{Nout}{Nout+Nin}$ ranking at top 30% or top 1%



(a) Patent (b) DBLP (c) Brightkite (d) Gowalla

Fig. 16. Average Running Time for Each Window in Four Datasets.

Table 3
Statistics of Datasets.

Dataset	m	e_4	n
Patent	6	25,000	26,000
DBLP	4,700	2,000	1,500
Brightkite	1,000	2,000	600
Gowalla	900	11,000	2,300

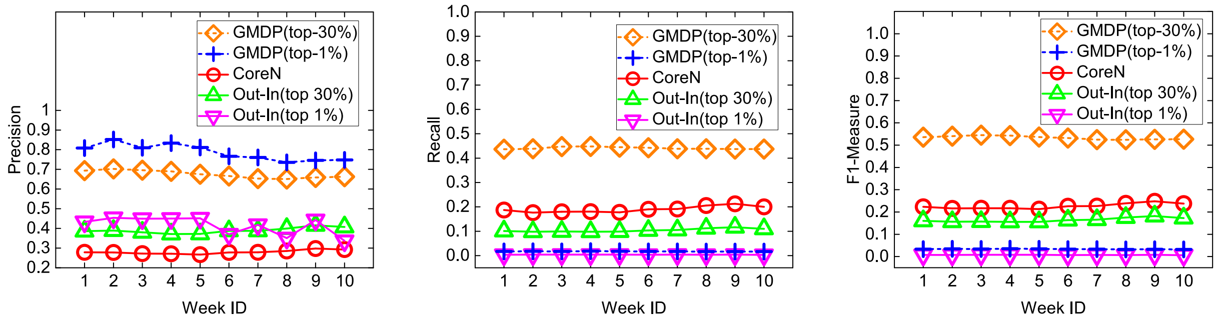


Fig. 17. Gowalla, Weekly Prediction.

would leave. In addition, the recall of GMDP with $\theta = top\text{-}30\%$ ($\theta = top\text{-}1\%$ resp.) is higher than that of Out-In when users with $\frac{Nout}{Nout+Nin}$ ranking at top 30% (top 1% resp.) would be predicted to leave by Out-In.

The predicted results by our GMDP algorithm have many applications in reality. For example, every weekend, by performing our algorithm, a tourist resort or a big shopping mall located in a region could push some newly promotional information to the persons who are predicted to lose interests in the place in order to attract them to come back (setting a small θ is better since the precision of the results is higher and the number of the predicted user-group pairs is not large so that the promotional costs could be saved). The predicted results could also be used for further analyzing (e.g. clustering the predicted leaving users by their occupations and hobbies, etc.) to help the tourist resort or the shopping mall to adjust the management tactics (setting a large θ is better so that more predicted results could be acquired for analyzing).

6. Conclusion

In this paper, a novel unsupervised prediction algorithm, GMDP, has been developed to improve prediction precision of the group members' departure prediction problem in dynamic information networks. In GMDP, for each member, the group-combined activeness score is first introduced so that the member with a lower score would be more likely to leave the group. Experiments on extensive real-world dynamic information networks verify that GMDP is practically efficient, and in most cases, GMDP greatly improve the prediction precision compared with the best existing methods. Specifically, in datasets Brightkite, Gowalla and DBLP, GMDP significantly outperforms CoreN and Out-In in terms of the precision. While in the dataset Patent which has the unique characteristic that most of the inventors don't keep inventing the same class of patents contiguously for several years, GMDP is only a little superior to CoreN and Out-In as for the precision.

In the future, we will modify GMDP to further improve the prediction precision for the specific dataset Patent. Besides, we will devise some time saving strategies and implement parallel computing to make GMDP more efficient for many other real-world dynamic information networks with a larger number of groups, nodes and edges. Moreover, after modeling more other datasets (e.g., arXiv[8] and Web of Science[9]) as dynamic information networks, we will try to expand our GMDP algorithm into other applications like disambiguating authors' names [1] and predicting the trends of scientific topics [25].

CRediT authorship contribution statement

Xinrui Wang: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **Hong Gao:** Conceptualization, Formal analysis, Resources, Data curation, Writing - review & editing, Supervision, Funding acquisition. **Zhipeng Cai:** Validation, Investigation, Writing - review & editing, Project administration. **Jianzhong Li:** Resources, Supervision, Funding acquisition.

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.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61632010, No. U19A2059, No. 61732003, No. 61832003 and No. U1811461.

[8] https://arxiv.org/
[9] https://clarivate.com/webofsciencegroup/solutions/web-of-science-core-collection/

References

[1] D.R. Amancio Jr., O.N.O. da Fontoura Costa L., Topological-collaborative approach for disambiguating authors' names in collaborative networks, Scientometrics 102 (2015) 465–485.
[2] E. Amitay, D. Carmel, N. Har'El, S. Ofek-Koifman, A. Soffer, S. Yogev, N. Golbandi, Social search and discovery using a unified approach, WWW (2009) 1211–1212.
[3] L. Backstrom, D.P. Huttenlocher, J.M. Kleinberg, X. Lan, Group formation in large social networks: membership, growth, and evolution, SIGKDD (2006) 44–54.
[4] R. Baeza-Yates, M. Lalmas, User engagement: the network effect matters!, CIKM (2012) 1–2.
[5] V. Batagelj, M. Zaversnik, An o(m) algorithm for cores decomposition of networks, CoRR (2003), cs.DS/0310049.
[6] K. Bhawalkar, J.M. Kleinberg, K. Lewi, T. Roughgarden, A. Sharma, Preventing unraveling in social networks: The anchored k-core problem, SIAM Journal on Discrete Mathematics 29 (2015) 1452–1475.
[7] X. Chen, G. Song, X. He, K. Xie, On influential nodes tracking in dynamic social networks, SIAM (2015) 613–621.
[8] M.S.Y. Chwe, Communication and coordination in social networks, The Review of Economic Studies 67 (2000) 1–16.
[9] P.S. Dodds, D.J. Watts, A generalized model of social and biological contagion, Journal of Theoretical Biology 232 (2005) 587–604.
[10] N. Ducheneaut, N. Yee, E. Nickell, R.J. Moore, The life and death of online gaming communities: a look at guilds in world of warcraft, CHI (2007) 839–848.
[11] D. Dupuis, C. du Mouza, N. Travers, G. Chareyron, Real-time influence maximization in a RTB setting, Data Science and Engineering 5 (2020) 224–239.
[12] Y. Han, J. Tang, Who to invite next? predicting invitees of social groups, IJCAI (2017) 3714–3720.
[13] A. Harkins, Network Games with Perfect Complements, Warwick University Draft, 2013, unpublished.
[14] E. Hyvönen, H. Rantala, Knowledge-based relation discovery in cultural heritage knowledge graphs, in: Proceedings of the Digital Humanities in the Nordic Countries 4th Conference, Copenhagen, Denmark, March 5–8, 2019, 2019, pp. 230–239..
[15] S.R. Kairam, D.J. Wang, J. Leskovec, The life and death of online groups: predicting group growth and longevity, WSDM (2012) 673–682.
[16] D. Kempe, J.M. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, SIGKDD (2003) 137–146.
[17] M. Latapy, Main-memory triangle computations for very large (sparse (power-law)) graphs, Theoretical Computer Science 407 (2008) 458–473.
[18] J. Leskovec, L.A. Adamic, B.A. Huberman, The dynamics of viral marketing, Transactions on the Web 1 (2007) 5.
[19] H. Li, Z. Nie, W. Lee, C.L. Giles, J. Wen, Scalable community discovery on textual data with relations, CIKM (2008) 1203–1212.
[20] J. Li, T. Sellis, J.S. Culpepper, Z. He, C. Liu, J. Wang, Geo-social influence spanning maximization, ICDE (2018) 1775–1776.
[21] F.D. Malliaros, M. Vazirgiannis, To stay or not to stay: modeling engagement dynamics in social graphs, CIKM (2013) 469–478.
[22] M.E. Newman, Spread of epidemic disease on networks, Physical Review E 66 (2002) 016128.
[23] H.L. O'Brien, E.G. Toms, What is user engagement? A conceptual framework for defining user engagement with technology, Journal of the Association for Information Science and Technology 59 (2008) 938–955.
[24] G. Palla, A. Barabasi, T. Vicsek, Quantifying social group evolution, Nature 446 (2007) 664–667.
[25] Prabhakaran, V., Hamilton, W.L., McFarland, D.A., Jurafsky, D., 2016. Predicting the rise and fall of scientific topics from trends in their rhetorical framing, in: ACL..
[26] J. Qiu, Y. Li, J. Tang, Z. Lu, H. Ye, B. Chen, Q. Yang, J.E. Hopcroft, The lifecycle and cascade of wechat social messaging groups, WWW (2016) 311–320.
[27] I.T. Ribón, G. Palma, A. Flores, M. Vidal, Considering semantics on the discovery of relations in knowledge graphs, EKAW (2016) 666–680.
[28] Y. Sun, J. Han, C.C. Aggarwal, N.V. Chawla, When will it happen?: relationship prediction in heterogeneous information networks, WSDM (2012) 663–672.
[29] Y. Sun, J. Tang, J. Han, M. Gupta, B. Zhao, Community evolution detection in dynamic heterogeneous information networks, Workshop on MLG (2010) 137–146.
[30] J. Ugander, L. Backstrom, C. Marlow, J.M. Kleinberg, Structural diversity in social contagion, Proceedings of the National Academy of Sciences of the United States of America 109 (2012) 5962–5966.
[31] Z. Wang, C. Wang, W. Wang, X. Gu, B. Li, D. Meng, Adaptive relation discovery from focusing seeds on large networks, ICDE (2020) 217–228.
[32] S. Wu, A.D. Sarma, A. Fabrikant, S. Lattanzi, A. Tomkins, Arrival and departure dynamics in social networks, WSDM (2013) 233–242.
[33] M. Ye, P. Yin, W. Lee, Location recommendation for location-based social networks, ACM-GIS (2010) 458–461.
[34] H. Yin, Q. Wang, K. Zheng, Z. Li, J. Yang, X. Zhou, Social influence-based group representation learning for group recommendation, ICDE (2019) 566–577.
[35] F. Zhang, W. Zhang, Y. Zhang, L. Qin, X. Lin, OLAK: an efficient algorithm to prevent unraveling in social networks, VLDB 10 (2017) 649–660.
[36] J. Zhang, C. Wang, J. Wang, J.X. Yu, Inferring continuous dynamic social influence and personal preference for temporal behavior prediction, VLDB 8 (2014) 269–280.
[37] J. Zhang, C. Wang, P.S. Yu, J. Wang, Learning latent friendship propagation networks with interest awareness for link prediction, SIGIR (2013) 63–72.
[38] T. Zhang, P. Cui, C. Faloutsos, Y. Lu, H. Ye, W. Zhu, S. Yang, Come-and-go patterns of group evolution: A dynamic model, SIGKDD (2016) 1355–1364.
[39] J. Zhao, S. Shang, P. Wang, J.C.S. Lui, X. Zhang, Tracking influential nodes in time-decaying dynamic interaction networks, ICDE (2019) 1106–1117.
[40] Z. Zhu, X. Fan, X. Chu, J. Bi, HGCN: A heterogeneous graph convolutional network-based deep learning model toward collective classification, KDD (2020) 1161–1171.
[41] Chen Jihong, Co-purchaser Recommendation for Online Group Buying, Data Science and Engineering (2020), https://doi.org/10.1007/s41019-020-00138-w.