Title	Tracking the evolution of communities in dynamic social networks
Authors(s)	Greene, Derek; Doyle, Dónal; Cunningham, Pádraig
Publication date	2010-08-11
Publication information	N. Memon and R. Alhajj (ed.s). 2010 2010 International Conference on Advances in Social Network Analysis and Mining : ASONAM 2010 : proceedings
Conference details	2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2010), 9-11 August 2010, Odense, Denmark
Publisher	IEEE
Link to online version	http://dx.doi.org/10.1109/ASONAM.2010.17
Item record/more information	http://hdl.handle.net/10197/2031
Publisher's version (DOI)	10.1109/ASONAM.2010.17

Tracking the Evolution of Communities in Dynamic Social Networks

Derek Greene
School of Computer Science & Informatics
University College Dublin
derek.greene@ucd.ie

Dónal Doyle
Idiro Technologies
Dublin, Ireland
donal.doyle@idiro.com

Pádraig Cunningham
School of Computer Science & Informatics
University College Dublin
padraig.cunningham@ucd.ie

Abstract

Real-world social networks from a variety of domains can naturally be modelled as dynamic graphs. However, approaches to detecting communities have largely focused on identifying communities in static graphs. Recently, researchers have begun to consider the problem of tracking the evolution of groups of users in dynamic scenarios. Here we describe a model for tracking the progress of communities over time in a dynamic network, where each community is characterised by a series of significant evolutionary events. This model is used to motivate a community-matching strategy for efficiently identifying and tracking dynamic communities. Evaluations on synthetic graphs containing embedded events demonstrate that this strategy can successfully track communities over time in volatile networks. In addition, we describe experiments exploring the dynamic communities detected in a real mobile operator network containing millions of users.

1 Introduction

Social network analysis methods have traditionally focused on the representation of graphs as static networks. This has been the case for the task of community detection, where the goal is to identify meaningful group structures in the network. However, by representing a dynamic source of data as a static network, group structures present over shorter periods of time can be difficult to identify or may be completely ablated. In addition, by discarding temporal information, the detail of the evolutionary behaviour of these groups is lost.

Modelling structural changes in networks is important in a wide range of real-world social network analysis problems, where the data naturally has a temporal aspect. The evolving nature of social media makes it a candidate for this type of analysis. Researchers may be interested in examining the formation and change in communities – such as clusters of frequently interacting authors in the blogosphere [13], or the formation of circles of friends in online networks such as Facebook and Twitter. Other application areas include the analysis of the evolution of research communities within and across academic disciplines [14]. A particularly relevant application is the analysis of mobile subscriber networks [17], where the activity of groups of users over time are potential predictors of future behaviour that is of specific interest to network operators, such as subscriber churn or handset adoption. However, the scale of such networks presents a challenge even for existing static community finding techniques.

A number of researchers have highlighted the importance of identifying the key events that characterise the life cycle of a community of users in a dynamic network [14]. In this paper we describe a model for tracking the evolution and structure of communities over multiple time steps in a dynamic network, where the life-cycle of each community is characterised by a series of significant events. Based on this model, we propose a simple but effective method for efficiently identifying and tracking these dynamic communities, which involves matching communities found at consecutive time steps in the individual snapshot graphs. Unlike other approaches (*e.g.* [14]), the method is independent of the choice of underlying community finding algorithm applied to the step graphs. It can also aggregate information from either disjoint partitions or overlapping groupings of nodes. To evaluate the method, we introduce a procedure for generating synthetic dynamic networks, containing embedded communities and events, to act as a "ground-truth" for validation. We show that our method performs well on this data, where it readily scales to networks consisting of millions of nodes and tens of thousands of communities. In the second part of our evaluation, we describe our experiments on real-world mobile operator call graphs generated over a two month period containing approximately four million unique users.

The remainder of the paper is structured as follows. In the next section we provide a brief overview of existing work in the area of dynamic community finding, also alluding to other related research areas. In Section 3 we outline the proposed model, and provide a detailed description of the associated tracking method. An evaluation of the operation of this method on both synthetic networks and mobile call data is given in Section 4. The paper concludes with a summary and suggestions for plans for future work.

2 Related Work

2.1 Dynamic Community Finding

A significant body of literature exists concerning the problem of finding communities in static graphs. Motivated by the temporal nature of real-world social networks, some of this focus has shifted to the topic of mining dynamic graphs. Palla *et al.* [14] proposed an extension of the popular clique percolation method to identify community-centric events in the evolution of dynamic graphs. This extension involved applying community detection to joint graphs for pairs of consecutive time steps. The resulting clique-based communities are subsequently matched to communities in either of the individual time steps. This approach was applied to both mobile subscriber networks and bibliographic co-authorship graphs. A similar life-cycle model was proposed in [16], where the dynamic community finding approach was formulated as a graph colouring problem. Since the problem is NP-hard, the authors employed a heuristic technique that involves greedily matching pairs of node sets between time steps, in descending order of similarity. This technique was shown to perform well on a number of small well-known social network datasets.

Asur *et al.* [1] described a community event identification strategy which used a matching-based approach, which was implemented in the form of bit operations computed on time step community membership matrices. This strategy was applied to both bibliographic networks and clinic trial data in the context of pharmaceuticals. Unlike other authors, in [1] a significant emphasis was also placed on the life cycle of nodes themselves. However this type of analysis may not always be practical or relevant for larger datasets where network high-level summarisation is the primary objective, rather than ego-centric analysis.

2.2 Other Related Areas

The more general problem of identifying clusters in dynamic data has been studied by a number of authors. Notably, Chakrabarti *et al.* [4] proposed an "evolutionary clustering" framework to handle this problem, whereby both current and historic information was incorporated into the objective of the clustering process. The authors used this to formulate dynamic variants of common partitional and agglomerative clustering algorithms suitable for feature-based data. Evolutionary versions of common spectral clustering algorithms have also been proposed [5].

Set matching heuristics have been applied to other problems that resemble the dynamic community finding task. In data integration tasks, such techniques have been used as part of "late integration" strategies to aggregate previously generated clusterings produced independently on each view of the same network [8]. More generally, the problem of ensemble clustering is concerned with combining a diverse set of clusterings to produce a consensus solution that summarises the information provided by the constituent clusterings [6]. However, the unique temporal aspect of the data in dynamic community detection distinguishes the problem from these other two tasks, where the sequence of groupings to be aggregated is not important.

3 Methods

3.1 Model for Dynamic Community Analysis

In this section, we provide a generalisation of previously proposed models for dynamic community finding, focused around the life cycle of communities. This model is used to frame and motivate the method described in Section 3.2.

Firstly, we represent a dynamic network as a set of *time step* graphs $\{g_1, \ldots, g_l\}$, providing snapshots of the nodes and edges in the overall network at successive intervals. The problem then becomes the identification of a set of k' *dynamic communities* $\mathbb{D} = \{D_1, \ldots, D_{k'}\}$ that are present in the network across one or more time steps. We refer to *step communities* that are identified at individual time steps, which represent specific observations of dynamic communities at a given point in time. Unlike the approach described in [14], these need not necessarily comprise of cliques – the observations can be taken from any disjoint or overlapping grouping that provides assignments for some or all of the nodes in the overall network. We denote the set of k_t step communities identified at time t as $\mathbb{C}_t = \{C_{t1}, \ldots, C_{tk_t}\}$.

Each dynamic community D_i can be represented by a *timeline* of its constituent step communities, ordered by time, with at most one step community for each step t. The diagram in Figure 1 shows a simple case involving three step clusterings containing three dynamic communities. The timelines for these three dynamic communities are straight-forward:

- D_1: $\{C_{11}, C_{21}, C_{31}\}$
- D_2: $\{C_{22}, C_{32}\}$
- D_3: $\{C_{12}, C_{23}\}$

A more complex example is shown in Figure 2. Note that while there appear to be three distinct branches at time $t = 3$, there are in fact four dynamic communities with four corresponding timelines:

- D_1: $\{C_{11}, C_{21}, C_{31}\}$
- D_2: $\{C_{12}, C_{21}, C_{31}\}$
- D_3: $\{C_{13}, C_{22}, C_{32}\}$
- D_4: $\{C_{13}, C_{23}, C_{33}\}$

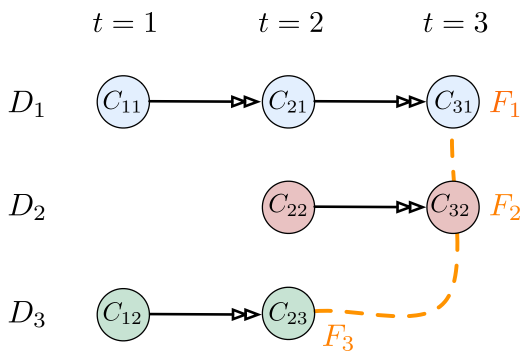


Figure 1: Example of three dynamic communities tracked over three time steps, featuring continuation, birth, and death community life-cycle events.

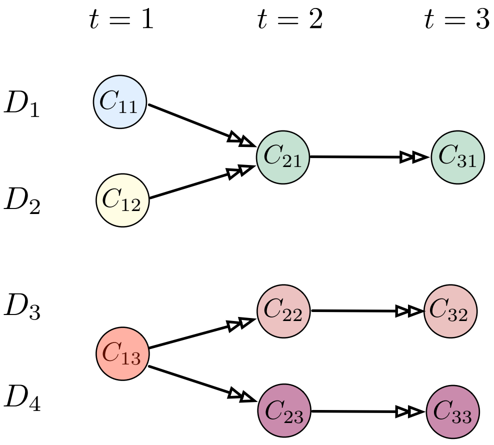


Figure 2: Example of four dynamic communities tracked over three time steps, featuring merging and splitting life-cycle events.

The most recent observation in a timeline is referred to as the *front* of the dynamic community – the front for D_i is denoted F_i. The fronts for the three dynamic communities are highlighted in Figure 1. Note that the dynamic community D_3 does not have a corresponding observation at time $t = 3$ – its front is the step community C_{23} from time $t = 2$.

In the dynamic community finding literature there can be seen a broad consensus (*e.g.* [14, 16, 1]) on the fundamental events that can be used to characterise the evolution of dynamic communities. Given the notation above, we can formulate these key events in terms of a set of rules covering step and dynamic communities:

- *Birth:* The emergence of a step community C_{tj} observed at time t for which there is no corresponding dynamic community in $\mathbb{D}$. A new dynamic community D_i containing C_{tj} is created and added to $\mathbb{D}$. An example in Figure 1 is the community D_2 born in the second time step.
- *Death:* The dissolution of a dynamic community D_i occurs when it has not been observed (*i.e.* there has been no corresponding step community) for at least d consecutive time steps. D_i is subsequently removed from the set $\mathbb{D}$. An example in Figure 1 is D_3, assuming that no further step communities are subsequently assigned to its timeline.
- *Merging:* A merge occurs if two distinct dynamic communities (D_i, D_j) observed at time $t - 1$ match to a single step community C_{ta} at time t. The pair subsequently share a common timeline starting from C_{ta}. In Figure 2 the dynamic communities D_1 and D_2 are both matched to C_{21} in the second step.
- *Splitting:* It may occur that a single dynamic community D_i present at time $t-1$ is matched to two distinct step communities (C_{ta}, C_{tb}) at time t. A branching occurs with the creation of an additional dynamic community D_j that shares the timeline of D_i up to time $t-1$, but has a distinct timeline from time t onwards. In Figure 2 an existing dynamic community D_3 is matched to both C_{22} and C_{23} in the second step, resulting in the creation of an additional dynamic community D_4.
- *Expansion:* The expansion or growth of a dynamic community D_i occurs when its corresponding step community at time t is significantly larger than the previous front associated with D_i (*e.g.* a growth of $> 10\%$).
- *Contraction:* The contraction or reduction of a dynamic community D_i occurs when its corresponding step community at time t is significantly smaller than the previous front associated with D_i (*e.g.* a reduction of $> 10\%$).

We may also have trivial one-to-one matching or *continuation* events where a dynamic community observed at time t also has an observation at time $t + 1$. However, note that a dynamic community may not necessarily be observed at all time steps after birth – it may be observed at birth time t and

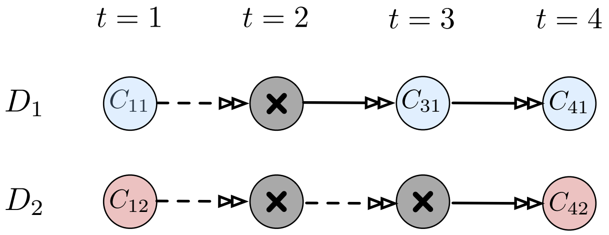


Figure 3: Example of two "intermittent" dynamic communities which are not observed at all time steps after birth. The dynamic community D_1 is unobserved in the graph at time $t = 2$, but continues in time $t = 3$, while D_2 is missing from both $t = 2$ and $t = 3$.

at death time $t' > t$, but may be missing from one or more intermediate steps (but less than d steps) between t and t'. Two examples are shown in Figure 3. This reflects the notion that temporally "intermittent" structure may exist in a network, which is dependent on the behaviour of the nodes in the network and the duration or granularity of each time step.

3.2 Tracking Communities Across Time Steps

In the context of the model described above, a key question concerns how best to map step communities at each time t to the existing set of dynamic communities $\mathbb{D}$. Further questions may arise regarding the feasibility of performing this correspondence process in an efficient manner for graphs containing a large number of nodes and communities.

One approach is to formulate this problem as a weighted bipartite matching task, which involves finding the optimal correspondence between the dynamic community fronts and the step communities. A common solution to weighted bipartite matching is the Hungarian method [10]. The strategy of finding the optimal match between communities in different time steps was previously considered in [16]. However, in general, bipartite matching approaches will assume a zero-to-one or one-to-one mapping between nodes in the two sets – which will not readily support the identification of dynamic events such as community merging and splitting. Rather, we propose a heuristic threshold-based method, which allows for many-to-many mappings between communities across different time steps. Threshold-based cluster aggregation techniques have previously been employed in dynamic community finding [1], and also in data integration [8]. This strategy is independent of the choice of the underlying static community finding algorithm applied to the individual step graphs.

The strategy proceeds as follows. The first step grouping $\mathbb{C}_1$ is generated by applying a chosen static community finding algorithm to the graph g_1 – we use this graph to bootstrap the process. A distinct dynamic community is created for each step community. The next grouping $\mathbb{C}_2$ is generated on the graph g_2. An attempt is made to match these step communities with the fronts $\{F_1, \ldots, F_{k'}\}$ (*i.e.* the step communities from $\mathbb{C}_1$). All pairs (C_{2a}, F_i) are compared, and the dynamic community timelines and fronts are updated based on the event rules described previously in Section 3.1. The process continues until all l step graphs have been processed.

To perform the actual matching between $\mathbb{C}_t$ and the fronts $\{F_1, \ldots, F_{k'}\}$, we employ the widely-adopted Jaccard coefficient for binary sets [9]. Given a step community C_{ta} and a front F_i, the similarity between the pair is calculated as:

$$sim(C_{ta}, F_i) = \frac{|C_{ta} \cap F_i|}{|C_{ta} \cup F_i|} \tag{1}$$

If the similarity exceeds a matching threshold $\theta \in [0, 1]$, the pair are matched and C_{ta} is added to the timeline for the dynamic community D_i.

For practical purposes, the intersection calculations required for Eqn. 1 can be performed efficiently using a number of strategies, including optimisations based on sorted sets [2], or bit array operations [1]. In the implementation used in this paper, we represent dynamic communities in terms of a node-community map against which incoming step communities are compared. This change leads to substantial performance improvements when compared to a naïve implementation based on pairs of set structures – this is reflected in the running times presented later in Section 4.4.

1. Apply static community finding algorithm on g_1 to extract $\mathbb{C}_1$. Initialise $\mathbb{D}$ by creating a new dynamic community for each step cluster $C_{1i} \in \mathbb{C}_1$.
2. For each subsequent step $t > 1$, extract $\mathbb{C}_t$ from g_t.
3. Process every $C_{ta} \in \mathbb{C}_t$ as follows:
    1. Match all dynamic communities D_i for which $sim(C_{ta}, F_i) > \theta$.
    2. If there are no matches, create new dynamic community containing C_{ta}.
    3. Otherwise, add C_{ta} to each matching dynamic community.
4. Update the set of fronts for each dynamic community to be the latest matched step community. For each case where one existing dynamic community has been matched to 2 or more step communities, create a split dynamic community.
5. Repeat from #2 until all time step graphs have been processed.

Figure 4: Summary of the proposed dynamic community finding method.

The output of the matching process between $\mathbb{C}_t$ and $\{F_1, \ldots, F_{k'}\}$ will naturally reveal series of community evolution events. A step community C_{ta} matching to a single dynamic community indicates a "continuation", while the case where C_{ta} matches multiple dynamic communities results in a merge event. If no suitable match is found for C_{ta} above the threshold θ, a new dynamic community is created for C_{ta}. An overview of the entire process is provided in Figure 4.

Note that at any given time step, we can derive a conventional overlapping set of groups from the current dynamic communities in $\mathbb{D}$. Specifically, for each active (*i.e.* non-dead) dynamic community D_i with timeline $\{C_{1a}, \ldots, C_{tz}\}$, we construct a corresponding group with the nodes from the union of the step communities in its timeline $\{C_{1a} \cup \cdots \cup C_{tz}\}$. Any exact duplicate groups are removed. We will generally be interested in those "long-lived" dynamic communities that persist over more than one time step, rather than potentially noisy "short-lived" communities that only appear once and are never observed again. Therefore, groups corresponding to short-lived communities are also removed. As an example, for D_4 in Figure 2, the corresponding overlapping group will consist of the nodes contained in $\{C_{13} \cup C_{23} \cup C_{33}\}$.

4 Evaluation

4.1 Benchmark Network Generation

We wanted to examine the behaviour of the proposed approach on dynamic networks in the presence of the evolution events described previously using a form of ground truth. However, to the best of our knowledge no comprehensive benchmarks have been proposed for this purpose. Previously, attempts at synthesizing dynamic network data have used artificial data based on simple membership switching. For instance, Tang *et al.* [15] described an approach for generating small-scale synthetic multi-mode dynamic network data, by generating a set of latent communities, and randomly changing a proportion of community memberships at each stage. Similarly, Duan *et al.* [7] generated synthetic streams of random weighted directed graphs with embedded community structure, where the community structure changes between four different time slices. Lin *et al.* [13] adapted two well-known toy networks to produce additional time step networks using membership switching and corresponding changes to node edges.

To produce more realistic benchmark data, we developed an alternative set of benchmarks based on the embedding of events in synthetic graphs. Lancichinetti & Fortunato [11] proposed techniques for generating static networks with embedded ground truth communities, which can be used for benchmarking community finding techniques. A network is generated based on a user-specified set of parameters related to network size, node degree range, and community size. The generator uses these parameters to construct a suitable set of embedded communities around which the network is constructed.

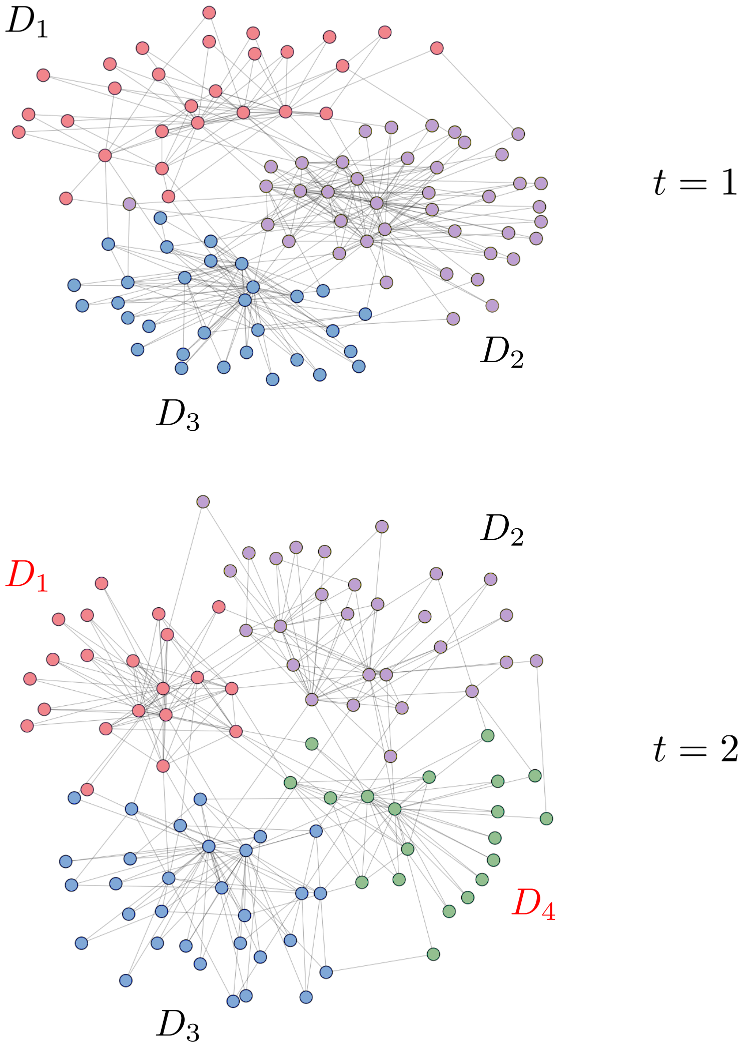


Figure 5: Simple benchmark dynamic graph representing 100 nodes over two time steps. At time $t = 2$, a *split* event occurs, where the embedded community D_1 divides into the pair of communities (D_1, D_4).

We adapted the tool provided by Lancichinetti & Fortunato to generate sets of unweighted undirected time step graphs. These graphs share similar characteristics, but each has community memberships (and edges) that have been permuted in a particular way. The change is controlled through the injection of a user-specified number of community events of a specific type. In this way the generator produces a ground truth for quantitative evaluation, in the form of a set of dynamic community timelines. A small example dynamic graph produced by the generator is shown in Figure 5, involving 100 nodes, four embedded dynamic communities, and a single merge event.

For the evaluations described here, we constructed four different synthetic networks for four different event types, covering 15,000 nodes over 5 time steps. The time step graphs share a number of parameters with the generation process described by Lancichinetti & Fortunato: nodes have mean degree of 20, maximum degree of 40, and a mixing parameter value of $\mu = 0.2$ which controls the level of edges between communities. The networks began at $t = 1$ with $\approx$ 400 embedded communities, which were constrained to have sizes in the range $[20, 60]$. In each of the four synthetic datasets, 20% of node memberships were randomly permuted at each step to simulate the natural movement of users between communities over time. Subsequently, embedded events were added by the generator as follows:

- *Intermittent communities*: In the first dataset, we consider the case of "intermittent" communities. We randomly hide a certain proportion of the communities from the original set $\mathbb{C}_1$ at each time step – 10% of communities are unobserved from time $t = 2$ onwards.
- *Expansion and contraction*: To examine the effect of rapid community expansion and contraction, we created graphs where 40 randomly selected communities expand or contract by 25% of their previous size. In the case of expansion, the new community members are chosen at random from other communities.
- *Birth and death*: To replicate the creation and destruction of communities, we create 40 additional communities by removing nodes from other existing communities, and randomly remove 40 existing communities.
- *Merging and splitting*: Finally, we considered the case where community merging and splitting events are embedded. Based on an initial set of communities, at each subsequent time step, 40 instances of community splitting was introduced, together with 40 cases where two existing communities were merged.

4.2 Experimental Setup

As noted previously, our dynamic community finding approach is independent of the choice of the underlying static community finding algorithm applied to the individual time steps. For our experiments we use the modularity optimisation algorithm introduced by Blondel *et al.* [3]. Since this algorithm tends to produce a hierarchical structure with a limited number of levels, in our benchmark experiments we derive a disjoint partition of the nodes by selecting the level where the number of communities is closest to the number of communities in the ground truth at time $t = 1$.

The validation of dynamic community finding techniques is not straight-forward. Based on the ground-truth available in the synthetic networks, we make use of conventional cluster validation techniques as follows. After each time step t, we derive an overlapping grouping from the set of active dynamic communities identified by our approach, as described in Section 3.1. We can use the generalised form of Normalised Mutual Information (NMI) introduced in [12] to compare the memberships of *nodes* in this grouping relative to those in the grouping derived in the same manner on the ground truth dynamic communities at the same time step.

We used a C++ implementation of the proposed dynamic community finding method. We ran the experiments using a single core on a standard Pentium Intel Quad 2.40GHz machine with 6GB RAM. The implementation, together with the synthetic graphs used in our experiments, are available online[1].

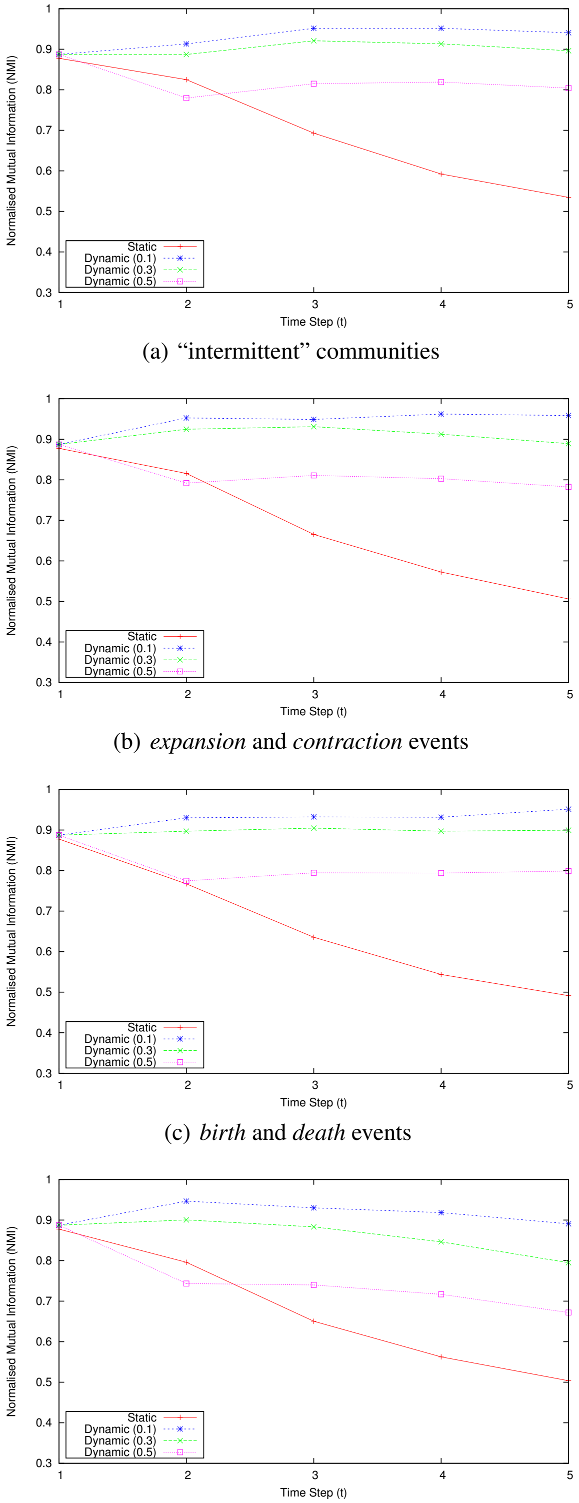
4.3 Discussion of Benchmark Results

In our experiments we investigated a range of threshold parameters $\theta \in [0.1, 0.5]$, with a fixed maximum age $d = 3$ for determining dead communities. As a baseline for comparison, we consider the traditional "static" approach of applying community finding to the graph constructed from aggregating edges across multiple time steps. Specifically, for each step t we apply the Blondel algorithm to the aggregated graph constructed from the edges in $\{g_1 \cup \cdots \cup g_t\}$. For comparison purposes we also select the level in the resulting hierarchy closest to the "correct" number of embedded communities.

Figure 6 shows a comparison of the output of the two strategies in terms of NMI accuracy, relative to the ground truth communities, as calculated after the addition of each time step graph from the four benchmark networks. These networks exhibit considerable volatility in terms of community structure and associated internal/external community edges between time steps. It is interesting to note that the static approach performs poorly under these conditions – the addition of additional edges from extra time steps does not provide the community detection algorithm with a clearer picture of the network. On the contrary, the decreasing NMI scores with time show that performance degrades as contradictory edge information is added.

We see from Figure 6 that the choice of the parameter θ has a noticeable effect on the performance of the dynamic strategy. Low values of θ lead to the most consistent accuracy scores across the five time steps. This observation is unsurprising, as the changing community memberships and sizes between

[1] See `http://mlg.ucd.ie/dynamic`



(a) “intermittent” communities

(b) *expansion* and *contraction* events

(c) *birth* and *death* events

(d) *merging* and *splitting* events

Figure 6: Performance, in terms of Normalised Mutual Information (NMI), of the proposed dynamic community finding method on four synthetic networks containing different types of embedded community events.

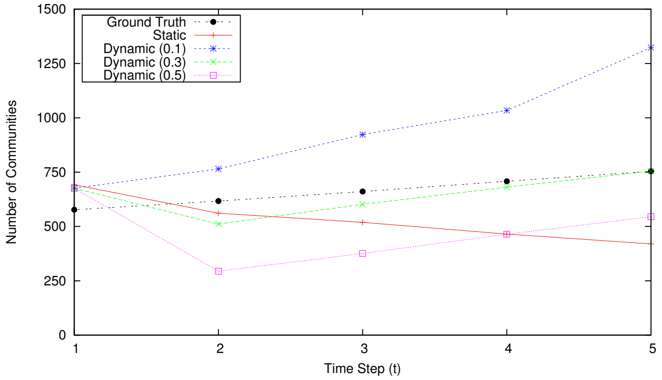


Figure 7: Number of overlapping communities found on a synthetic dynamic network containing embedded *merging* and *splitting* events, relative to the number of communities in the ground truth.

time steps make higher values for Eqn. 1 unlikely. However, for all parameter values consider, the dynamic method was superior after more than two time steps had been added. In terms of the number of distinct overlapping groups derived from the dynamic timelines, Figure 7 shows a representative example of the effect of changing the value of θ on the merge/split dataset. For a low value ($\theta = 0.1$), the number of groups is considerably higher than in the ground truth, where many of these groups are near duplicates. Whereas for a relatively high value ($\theta = 0.5$), the number of groups is lower. We observe that, since higher θ values naturally enforce a more conservative matching behaviour, this leads to more once-off short-lived communities which are subsequently discarded. In general, across all networks, a moderate value of $\theta = 0.3$ provided a reasonable compromise between node assignment accuracy and the identification of the optimal number of communities. Interestingly, the static strategy consistently under-estimated the true number of communities, particularly as the number of steps increased. Taken in conjunction with the results from Figure 6, we surmise that the merging of time steps prior to community identification does indeed partially or fully obscure a portion of the community structures in the network.

In general, when examining various network generation parameters, we observed that the static approach proved effective in cases where there was relatively little volatility between time steps, even when the community structures were relatively poorly defined *within* time steps (*i.e.* high inter-community connectivity). In this case the simple aggregation of the persistent edges was sufficient to uncover community structure. In contrast, the proposed dynamic strategy was most successful in cases where communities were evolving rapidly across time steps. Our experiences with mobile call data, described later, suggest that the latter scenario is more likely to occur in many real-world dynamic networks.

4.4 Evaluation of Scalability

To examine the scalability of the method described in Section 3, we used the synthetic graph generation process to produce dynamic networks of successively larger sizes, with the membership 5% of nodes switching from one step community to another. In each case, 5 time steps were produced using the same parameters described previously, and the experiments were repeated over 10 iterations.

Figure 8 shows that the scaling of the dynamic community method is close to linear in the number of nodes in the graph. For $\theta = 0.5$, a dynamic network with 100k nodes can be processed in under 2 seconds – resulting in 7035 dynamic communities. A graph containing 1 million nodes can be processed in approximately 85 seconds, resulting in the discovery of over 70k dynamic communities. Similarly, the number of maintained dynamic communities does not significantly impact on running times. We observed almost identical patterns on the same synthetic graphs for other matching parameter values $\theta \in [0.2, 0.7]$. The time to run the underlying Blondel algorithm ranged from a few milliseconds up to ≈ 40 seconds for the million node graph.

In these experiments, the primary constraint preventing us from applying the proposed method to networks beyond this point was the prohibitive memory requirements for generating synthetic graphs

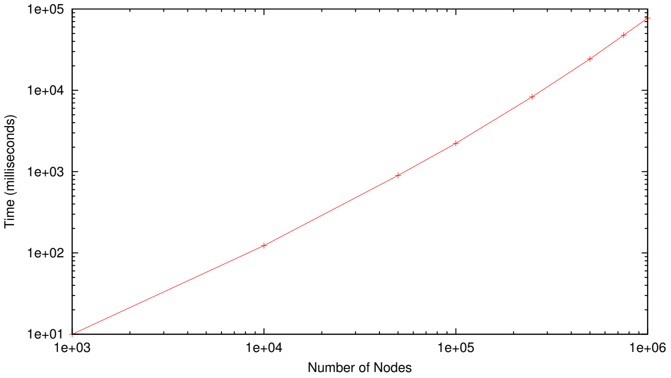


Figure 8: Plot of running time in milliseconds against number of nodes, for the proposed dynamic community finding method ($\theta = 0.5$) on synthetic graphs of increasing size from 1,000 to 1 million nodes.

with more than 10^6 nodes. Since the majority of the computation (*i.e.* calculating front-step community similarities) can be performed independently, we suggest that significant scope exists for parallelising the procedure to further improve computational performance. Even with the current implementation, we can readily process dynamic graphs far larger than those that can handled by methods such as those based on clique percolation [14].

4.5 Application to Real-World Data

In our second evaluation, we applied the proposed method to a real mobile operator network. Specifically we examined weekly voice call graphs over eight consecutive weeks, each containing approximately four million unique subscribers and tens of millions of edges. We preprocessed the data to produce unweighted, undirected graphs. A small number of nodes with unusually high degree in a given weekly call graph (calls to > 250 other nodes) were removed – these typically represent service numbers which can potentially obscure genuine user communities. Initial experiments on single week time steps yielded unstable results, suggesting that there may be insufficient structure in a week-long interval. To address this issue, we constructed longer time slices covering a two week period (with no overlap), yielding four separate time step graphs. Here we focus on the application of the proposed dynamic community finding method to these fortnightly graphs.

Due to its computational efficiency, we again applied the Blondel modularity optimisation algorithm [3] to produce sets of disjoint step communities for each step graph – this process took approximately 15 minutes in each case. We truncated the resulting hierarchy of communities at the first level. This yielded over 200,000 distinct step communities for each fortnightly interval.

Based on the analysis of benchmark data described in Section 4.3, we selected a matching threshold value of $\theta = 0.3$. The full procedure ran in just over 19 minutes, producing $\approx 150k$ long-lived dynamic communities present in at least two time steps. Of these, $\approx 27\%$ communities were visible in three time steps, and $\approx 9.6\%$ were observed across the full two month period. When overlapping groups were derived from the long-lived community timelines, about 80% of the communities contained between 5 and 15 users, which roughly corresponded to our prior assumptions regarding user calling patterns.

In addition to the implicit birth and continuation events, a range of other community timeline events and behaviours were detected in the data. For instance, we identified 82k long-lived communities exhibiting intermittent behaviour – these were born, unobserved for 1-2 steps, and then reappeared at a later time. Approximately $\approx 24k$ community merge events were detected in the four intervals. In many cases multiple inter-related merge events occurred at successive steps. A representative example is shown in Figure 9, where we see a number of dynamic communities merged at consecutive times periods. A smaller number of $\approx 4k$ community split events was also detected in the network. Further work on this data will focus on the identification of clusters of dynamic communities sharing characteristic timeline signatures, such as the one shown in Figure 9.

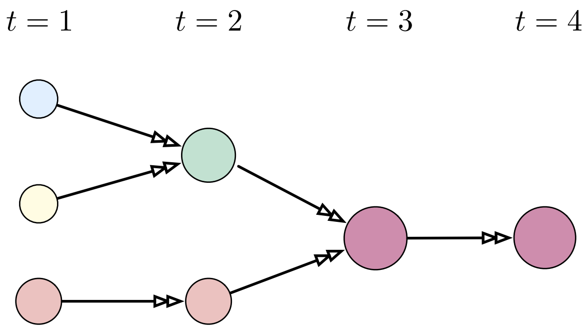


Figure 9: Example of dynamic community merge events identified in the dynamic analysis of a mobile subscriber network, over an 8 week period divided into 4 time steps.

5 Conclusions

In this paper, we have described both a general model for tracking communities in dynamic networks, and a fast, effective method based on that model which readily scales to graphs with $\approx 10^6$ nodes and 10^7 edges. We have described an approach for benchmarking dynamic community finding using synthetic graphs with embedded community events. Evaluations on these synthetic networks show that the proposed method performs at least as well if not better than static community finding.

Additionally we have performed a preliminary evaluation on a real-world mobile call network. On this data our method uncovered a large number of dynamic communities in this network with different evolutionary characteristics, while requiring relatively little computational overhead. Our experiments on this network suggest that the choice of the time step window size is important – this is an issue common to many dynamic data analysis procedures [4]. It will be interesting to study in detail the impact that this choice has on the quality and stability of dynamic communities.

The fact that our proposed method is independent of the choice of underlying algorithm is advantageous from one point of view – a suitable algorithm can be selected depending on the characteristics of the network (*e.g.* weighted/unweighted, directed/undirected, disjoint/overlapping communities). However, an interesting avenue for further work would be to integrate the proposed model with a scalable overlapping community detection algorithm, so that information from dynamic community timelines can be used to seed or direct the community finding algorithm in the next time step.

Acknowledgments

This research was supported by Science Foundation Ireland (SFI) Grant No. 08/SRC/I1407 (*Clique: Graph & Network Analysis Cluster*). The authors thank Idiro Technologies for their participation in the analysis of the mobile operator network.

References

[1] S. Asur, S. Parthasarathy, and D. Ucar. An event-based framework for characterizing the evolutionary behavior of interaction graphs. In *Proc. 13th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, page 921. ACM, 2007.

[2] R. Baeza-Yates. A fast set intersection algorithm for sorted sequences. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM 2004)*, volume 3109, pages 400–408. Springer, 2004.

[3] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech*, 10008, 2008.

[4] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 560. ACM, 2006.

[5] Y. Chi, X. Song, D. Zhou, K. Hino, and B. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proc. 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 162, 2007.

[6] E. Dimitriadou, A. Weingessel, and K. Hornik. A combination scheme for fuzzy clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(7):901–912, 2002.

[7] D. Duan, Y. Li, Y. Jin, and Z. Lu. Community mining on dynamic weighted directed graphs. In *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*, pages 11–18, New York, NY, USA, 2009. ACM.

[8] D. Greene and P. Cunningham. Multi-view clustering for mining heterogeneous social network data. In *Workshop on Information Retrieval over Social Networks, 31st European Conference on Information Retrieval (ECIR'09)*, 2009.

[9] P. Jaccard. The distribution of flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.

[10] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.

[11] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *eprint arXiv: 0904.3940*, 2009.

[12] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure of complex networks. *New J. Phys*, 11:033015, 2009.

[13] Y.-R. Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceeding of the 17th international conference on World Wide Web (WWW'08)*, pages 685–694, 2008.

[14] G. Palla, A. Barabási, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664, 2007.

[15] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. In *Proc. 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 677–685. ACM, 2008.

[16] C. Tantipathananandh, T. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07)*, pages 717–726, 2007.

[17] B. Wu, Q. Ye, and S. Yang. Group CRM: a new telecom CRM framework from social network perspective. In *Proceedings of the 1st ACM International Workshop on Complex Networks in Information and Knowledge Management (CNIKM)*, Hong Kong, China, 2009.