# 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

**University College Dublin**
**Technical Report UCD-CSI-2011-06**
**May 2011**

## Abstract

Real-world social networks from many domains can naturally be modelled as dynamic graphs. However, approaches for detecting communities have largely focused on identifying communities in static graphs. Therefore, 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 communities which persist over time in dynamic networks, where each community is characterised by a series of evolutionary events. Based on this model, we propose a scalable community-tracking strategy for efficiently identifying dynamic communities. Evaluations on a large number of synthetic graphs containing embedded evolutionary events demonstrate that this strategy can successfully track communities over time in dynamic networks with different levels of volatility. We then describe experiments to explore the evolving community structures present in real mobile operator networks, represented by monthly call graphs for millions of subscribers.

## 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, the 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 groups in the network is lost.

Modelling structural changes in networks is important in a range of real-world social network analysis problems where the data naturally has a temporal dimension. 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 (Lin *et al.*, 2008), or the formation of circles of friends in online social networks such as Facebook and Twitter. Other application areas include the analysis of the evolution of research communities within and across academic disciplines (Palla *et al.*, 2007). A particularly relevant application is the analysis of mobile subscriber networks (Wu *et al.*, 2009), where the behaviours of groups of users over time are potential predictors of future activity 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 which disregard temporal data.

Increasingly, researchers have highlighted the importance of identifying the key events that characterise the life cycle of a community of users in dynamic social networks (Palla *et al.*, 2007). In this paper we describe a model for tracking the evolution and structure of communities in multiple snapshots of 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.* Palla *et al.* (2007)), the method is independent of the choice of underlying community finding algorithm which is applied to the individual step graphs. It can also aggregate information from either disjoint or overlapping groupings of nodes. To evaluate the method, we introduce a procedure for generating synthetic dynamic networks. These networks contain embedded communities (both disjoint and overlapping) and evolutionary events, which provide 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 24 week period, containing approximately four million unique users.

The remainder of the paper is structured as follows. In the next section we provide an overview of existing work in the area of dynamic community finding and related research areas. In Section 3 we outline the proposed model for dynamic community finding, and provide a detailed description of the associated tracking method. Evaluations of the operation of this method on both synthetic benchmark networks and large mobile call graphs are given in Section 4. The paper concludes in Section 5 with a summary and suggestions for plans for future work.

## 2 Related Work

### 2.1 Dynamic Community Finding

A large body of literature exists concerning the problem of finding communities in static graphs (Fortunato, 2010). Many different algorithms have been proposed to identify communities in a single network snapshot, based on different objective functions and search strategies. Motivated by the temporal nature of real-world social networks, some of this focus has recently shifted to the topic of mining dynamic graphs. Although previously proposed dynamic community finding models differ in the approach used to actually find communities, a commonly-employed broad strategy involves considering a dynamic network in terms of $l$ individual *time step* graphs, representing successive snapshots of the graph taken at regular intervals.

For instance, Palla *et al.* (2007) 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 by Tantipathananandh *et al.* (2007), 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.* (2007) 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 in other work, the authors placed a significant emphasis 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.

Rosvall & Bergstrom (2010) proposed a framework for identifying changes in dynamic networks. Each network time step graph is clustered. Subsequently, the network is perturbed using a bootstrap resampling process and re-clustered – this is repeated for a large number of runs. This ensemble process is used to quantify the significance of the clusters generated at each time step. Associated clusters from different time steps are visually linked using "alluvial" diagrams, which chart the progression of a small number of clusters over time. The authors applied these techniques to study changing citations patterns in dynamic bibliographic networks covering a range of fields.

### 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.* (2006) proposed an "evolutionary clustering" framework to handle this problem, where 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 (Chi *et al.*, 2007), combining current and historic data to cluster relational data in the form of a pairwise affinity matrix.

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 (Greene & Cunningham, 2009). 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. A number of such algorithms construct a consensus clustering by matching together related clusters identified across multiple runs of a standard clustering algorithm such as $k$-means (Dimitriadou *et al.*, 2002; Dudoit & Fridlyand, 2003). However, the unique temporal aspect of the data in dynamic community detection distinguishes the problem from these two tasks, where the sequence of groupings to be aggregated is not considered.

## 3 Methods

### 3.1 Model for Dynamic Community Analysis

In this section, we provide a generalization 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.

#### 3.1.1 Dynamic Timelines

Firstly, we represent a dynamic network as a set of $l$ *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 multiple 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 by Palla *et al.* (2007), these need not necessarily comprise of cliques. Rather, the observations can be taken from any disjoint or overlapping grouping that provides assignments for some or all of the nodes in the complete 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* – that is, a sequence 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:

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

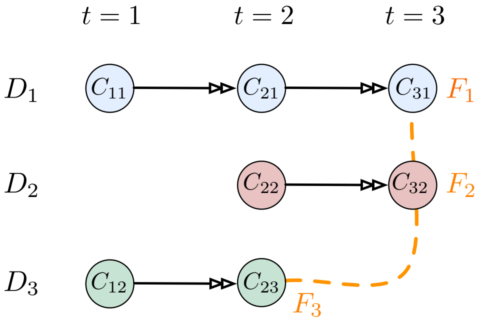


Figure 1: Example of three dynamic communities tracked over three time steps, featuring continuation, birth, and death community life-cycle events.

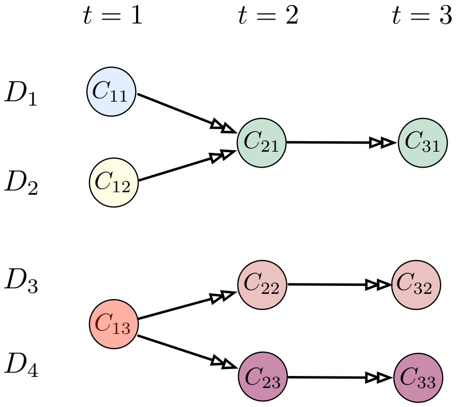


Figure 2: Example of four dynamic communities tracked over three time steps, featuring merging and splitting life-cycle events.

- $D_3$: $\{C_{12}, C_{23}\}$

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:

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

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 the previous step $t = 2$.

#### 3.1.2 Evolutionary Events

In the dynamic community finding literature there can be seen a broad consensus (*e.g.* Palla *et al.* (2007); Tantipathananandh *et al.* (2007); Asur *et al.* (2007)) on the fundamental events that can be used to characterize 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

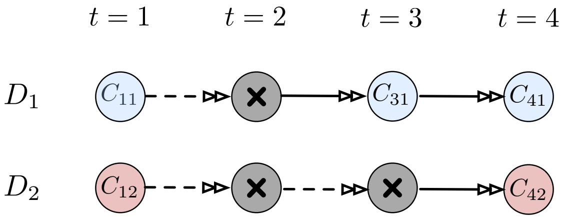


Figure 3: Example of two "intermittent" dynamic communities which are not observed at all time steps after birth. The dynamic community $D_1$ is unobserved in the graph at time $t = 2$, but continues in time $t = 3$, while $D_2$ is missing from both $t = 2$ and $t = 3$.

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, a dynamic community need not necessarily be observed at all steps after birth – it may be observed at birth time $t$ and at death time $t' > t$, but may be missing from one or more intermediate steps in between. 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 behavior of the nodes in the network and the duration or granularity of each time step. By maintaining a set of fronts for all dynamic communities (rather than simply the communities from the previous time step), we can support the identification of timelines with this kind of intermittent behavior.

## 3.2 Tracking Communities Across Time Steps

### 3.2.1 Tracking Procedure

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 the optimal weighted bipartite matching problem is the Hungarian method (Kuhn, 1955). The strategy of finding the optimal match between communities in different time steps was previously considered by Tantipathananandh *et al.* (2007). 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 (Asur *et al.*, 2007), and also in data integration (Greene & Cunningham, 2009). 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.

### 3.2.2 Matching Communities

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 (Jaccard, 1912). Given a step community $C_{ta}$ and a front $F_i$, the similarity between the pair is calculated as:

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

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 optimizations based on sorted sets (Baeza-Yates, 2004), or bit array operations (Asur *et al.*, 2007). 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.

The output of the matching process between $\mathbb{C}_t$ and $\{F_1, \ldots, F_{k'}\}$ will 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 $\theta$, a new dynamic community is created for $C_{ta}$. An outline of the entire process is provided in Figure 4.

1. Apply static community finding algorithm on $g_1$ to extract $\mathbb{C}_1$. Initialize $\mathbb{D}$ 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.

### 3.3 Post-Processing

#### 3.3.1 Generating Static Communities

At any given time step, a conventional overlapping set of communities can be derived from the current set of all dynamic community timelines 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 $\{C_{13} \cup C_{23} \cup C_{33}\}$.

#### 3.3.2 Ranking Dynamic Communities

For large networks analyzed over long time periods, the dynamic community finding process may generate a large number of timelines. For instance, for mobile call graphs such as the one described later in Section 4.6, hundreds of thousands of dynamic communities may be identified. This naturally leads to issues regarding interpretability. Therefore, it can be helpful to rank large sets of dynamic timelines based on some measure of significance. We consider two key aspects of significance: (a) the consistency or *stability* of node memberships in a dynamic community over time, (b) the *longevity* of a dynamic community. Specifically, for a dynamic community $D_i$ we compute the mean Jaccard similarity between each of its constituent step communities and the previous dynamic community front:

$$\mathrm{sig}(D_i) = \frac{1}{l-1} \sum_{C_{tj} \in D_i} \frac{|C_{tj} \cap F_t|}{|C_{tj} \cup F_t|} \tag{2}$$

Note that, to reward longevity, we normalize the sum with respect to the number of all possible pairs $l-1$, even if the dynamic community does not have an observation for every time step. This computation yields a score $\in [0, 1]$. A stable dynamic community, appearing frequently and with similar node memberships between time steps, will have a significance score close to 1. Less stable or infrequently appearing communities will have a score close to 0. The scores produced by Eqn. 2 can be used to produce a ranking of all dynamic communities identified on a given dataset.

## 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 synthesising dynamic network data have used artificial data based on simple membership switching. For instance, Tang *et al.* (2008) 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.* (2009) 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.* (2008) 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 (2009) 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.

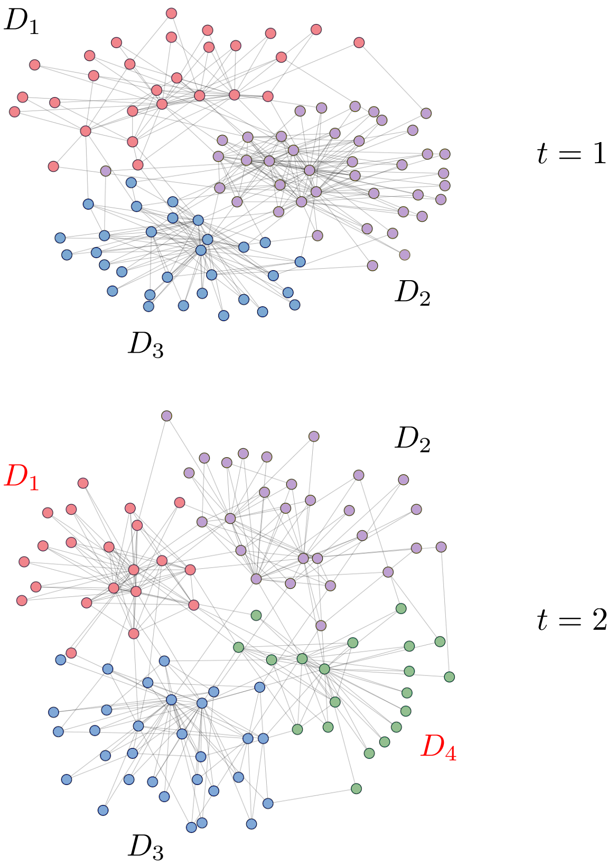


Figure 5: Simple benchmark dynamic graph representing 100 nodes over two time steps. At time $t = 2$, a *split* event occurs, where the embedded community $D_1$ divides into a pair of communities $(D_1, D_4)$.

We adapted the tool provided by Lancichinetti & Fortunato to generate sets of unweighted undirected time step graphs with embedded communities, both disjoint and overlapping. 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 types of synthetic dynamic networks covering different evolutionary behaviours, corresponding to the community evolution events outlined in Section 3.1.2. In each of the four network types, 20% of node memberships were randomly permuted at each time step to simulate the natural movement of users between communities over time. The behavior types are as follows:

1. *Intermittent communities*: Firstly, we consider the case of intermittently-appearing communities that are observed in only a subset of time steps. In addition to membership switching as described above, we also randomly hide a certain proportion of the communities from the original set $\mathbb{C}_1$ at each time step – 10% of original embedded communities are missing from the second time step onwards.
2. *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.

| **Parameter** | **Description** | **Set 1** | **Set 2** | **Set 3** |
|---|---|---|---|---|
| $k$ | average degree | 20 | 10 | 50 |
| $k_{max}$ | max degree | 40 | 40 | 120 |
| $C_{min}$ | minimum community size | 20 | 10 | 60 |
| $C_{max}$ | maximum community size | 60 | 50 | 100 |
| $\tau_1$ | degree exponent | -2 | $-2$ | $-2$ |
| $\tau_2$ | community exponent | -1 | $-1$ | $-2$ |
| $\mu$ | mixing parameter | 0.2 | 0.4 | 0.2 |
| $O_n$ | overlapping nodes | 0 | $N/2$ | $N$ |
| $O_m$ | communities per node | 1 | 2 | 3 |

Table 1: Parameter values used to generate three different sets of dynamic networks, each containing $N = 15,000$ nodes. Ten time step graphs were generated using these parameters for each of four event behaviour types.

3. *Birth and death*: To replicate the creation and destruction of communities, at each step we create 40 additional communities by removing nodes from other existing communities, and randomly remove 40 existing communities.
4. *Merging and splitting communities*: Finally, we considered the case where community merging and splitting events are embedded in the data. Based on an initial set of communities, at each subsequent time step, 40 instances of community splitting were introduced, together with 40 cases where two existing communities were merged.

The generated graphs share a number of parameters with the generation process described by Lancichinetti & Fortunato (2009). To investigate the effect of graphs with different levels of community membership overlap and inter-community connectivity, we created three different sets of dynamic networks for all of the behaviours listed above, resulting in 12 synthetic datasets in total. Each dataset consisted of 15,000 nodes represented over 10 time steps. Details for the different parameter sets are shown in Table 1. The parameters for *Set 1* correspond to those used in experiments for networks containing non-overlapping communities by Greene *et al.* (2010). The other two sets are based on parameters used by Lee *et al.* (2010) for benchmarking static community finding algorithms: the networks in *Set 2* contain some community overlap, but the level of inter-community connectivity or "mixing" is high; in *Set 3* the ground truth communities for the networks exhibit high overlap, with every node assigned to three different communities. For further discussion of the significance of the step graph generation parameters, consult Lancichinetti & Fortunato (2009).

### 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 the purpose of our experiments here, we used the MOSES overlapping community finding algorithm proposed by McDaid & Hurley (2010) to identify groups of nodes on individual time step graphs. This allowed us to test the dynamic community finding approach for both the case of disjoint and overlapping step communities.

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, we derive a static grouping from the set of active (*i.e.* non-dead) dynamic communities identified by our approach, as described in Section 3.3.1. To externally validate this grouping, we use the generalised form of Normalised Mutual Information (NMI) introduced by Lancichinetti *et al.* (2009) 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.

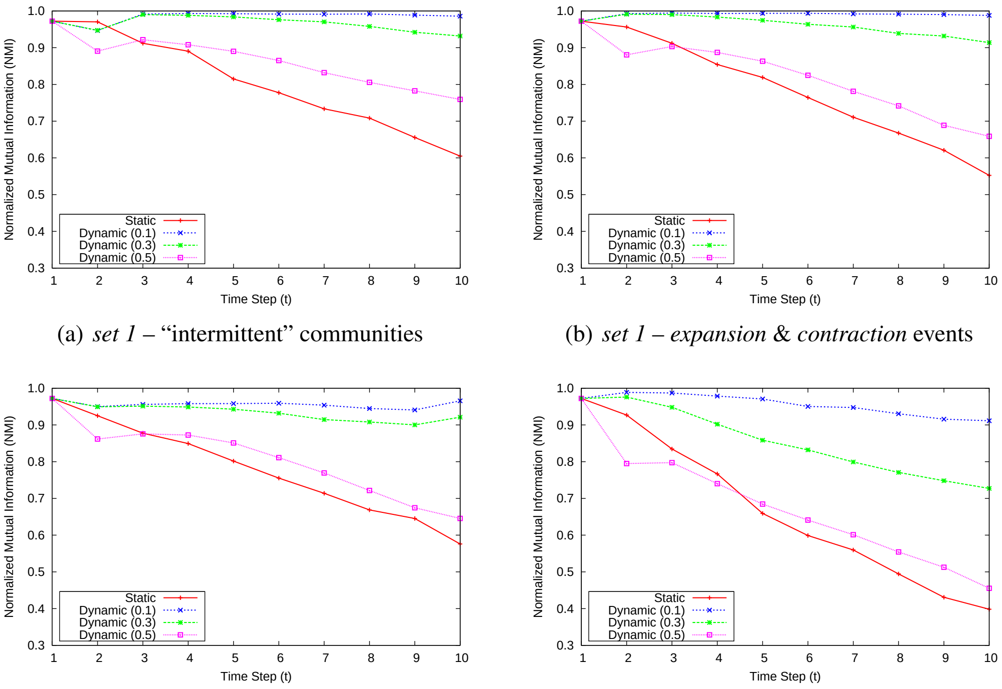
We used a C++ implementation of the dynamic community tracking approach. We ran the experiments using a single core on a Pentium Intel 2.40GHz server with 128GB RAM. An implementation of the tracker, together with the synthetic graph generator and the synthetic datasets used in our experiments, are made available online[1].

[1] See http://mlg.ucd.ie/snam

### 4.3 Evaluation: Community Identification

In our first evaluation, we examined the ability of the proposed approach to correctly identify the ground truth communities in the sets of benchmark dynamic networks described previously. The goal here was to determine whether applying a step-based dynamic community finding process could improve our ability to detect dynamic communities, when compared with traditional static community finding methods which treat dynamic networks as a single graph without regard to temporal information. In our experiments we investigated a range of matching 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 applied static community finding to the graph constructed from aggregating edges across multiple time steps. Specifically, for each step $t$, we employed the MOSES algorithm (McDaid & Hurley, 2010) to the aggregated graph constructed from the nodes and edges present in $\{g_1 \cup \cdots \cup g_t\}$. This process was repeated for all three sets of networks generated using the parameters listed in Table 1.

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 in *Set 1*. These networks contain non-overlapping ground truth communities that are well-separated in the step graphs. For all four event types we see that performance of static community detection degrades over time – the addition of more edges from successive time steps does not provide the community detection algorithm with a clearer picture of the network. On the contrary, the decreasing NMI scores show that performance degrades as contradictory edge information is added. The effect is most pronounced on the data with embedded merge and split events, where the changes in community structures over time are most volatile. In the case of dynamic community finding, relaxed matching thresholds ($\theta \leq 0.3$) yield significantly better results. In particular, the choice of $\theta = 0.1$ leads to little or no degradation in performance over time – for the intermittent and expansion/contraction cases the ground truth communities are identified with almost 100% accuracy.



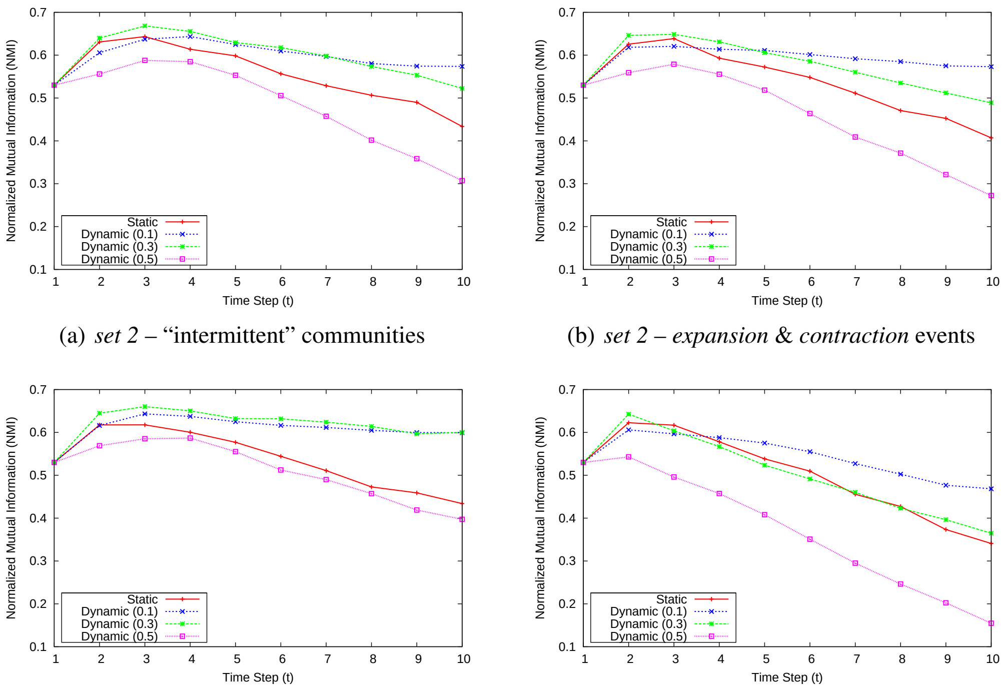
(a) *set 1* – "intermittent" communities

(b) *set 1* – *expansion* & *contraction* events

(c) *set 1* – *birth* & *death* events

(d) *set 1* – *merging* & *splitting* events

Figure 6: Comparison, in terms of Normalized Mutual Information (NMI), of static and dynamic community finding on the four synthetic networks in *Set 1*. These networks contain non-overlapping embedded communities that are well-separated from one another.



(a) *set 2* – “intermittent” communities

(b) *set 2* – *expansion* & *contraction* events

(c) *set 2* – *birth* & *death* events

(d) *set 2* – *merging* & *splitting* events

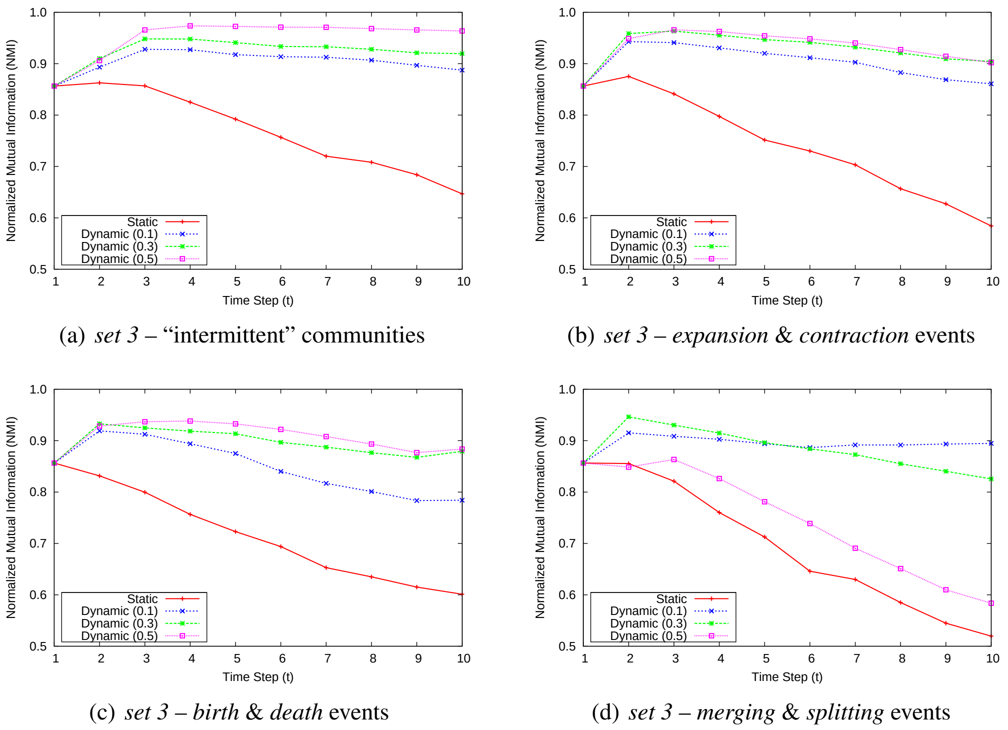
Figure 7: Comparison, in terms of NMI, of static and dynamic community finding on the four synthetic networks in *Set 2*. These networks contain some overlapping embedded communities, with relatively high inter-community edge connectivity.

Figure 7 shows the results of evaluations performed on the four networks in *Set 2* which contain embedded communities with limited overlap – 50% of the 15,000 nodes belong to more than one community, while the remainder are assigned to a single embedded community. There is also a considerable proportion of edges outside communities, making distinct communities more difficult to identify. The results indicate that dynamic community finding with a low $\theta$ parameter provides consistent accuracy over time. This is in contrast to the decreasing NMI scores for the static approach. For these four networks we also observe that a high threshold value ($\theta \geq 0.5$) is not appropriate – an overly conservative matching policy can lead to pairs of associated step communities not being matched together. This may be due to both changing memberships between time steps, or the inability of the underlying community finding algorithm to accurately identify communities on the time step graphs. The issues of volatility and the effects of different $\theta$ values are explored in more detail in Section 4.4.

Figure 8 lists results for the networks in *Set 3*, where every node is assigned to multiple communities which overlap considerably – this is designed to replicate the kind of pervasive overlap often observed in real-world networks (Ahn *et al.*, 2010). The communities are also larger than those present in *Set 1* and *Set 2*. These networks demonstrate a significant difference in performance between the static and dynamic approaches. Again the presence of contradictory information, arising from changing node memberships due to evolution events and membership switching, leads to a large decline in NMI scores for the static approach from $t = 3$ onwards. For the less volatile behaviour types (*i.e.* intermittent communities and expansion/contraction), conservative matching thresholds were effective in this case. For the merging and splitting data, lower thresholds yielded the highest accuracies.

### 4.4 Evaluation: Effects of Volatility

Following the results shown in the previous section, we now examine the degree to which community membership volatility impacts upon both the static and dynamic community finding strategies. That



(a) *set 3* – “intermittent” communities

(b) *set 3* – *expansion* & *contraction* events

(c) *set 3* – *birth* & *death* events

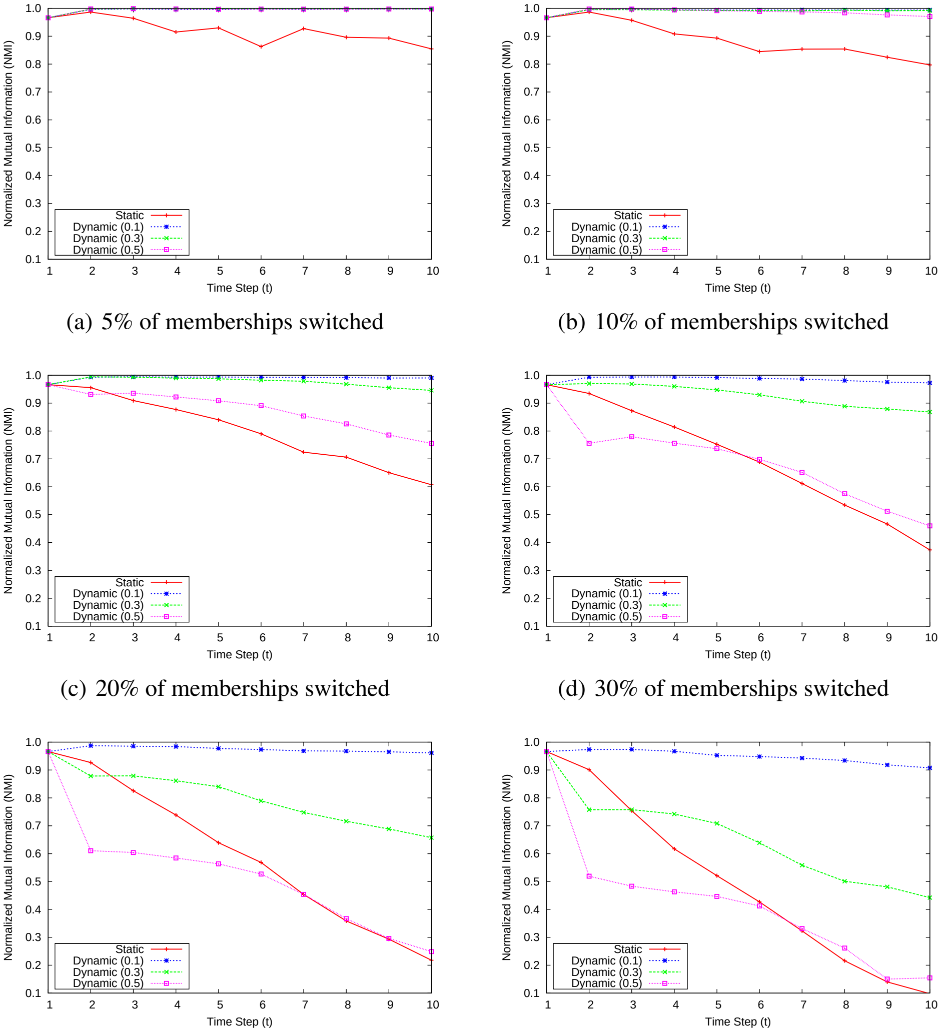
(d) *set 3* – *merging* & *splitting* events

Figure 8: Performance, in terms of NMI, of dynamic community finding on the four synthetic networks in *Set 3*. These networks contain highly overlapping embedded communities, with every node assigned to approximately three communities at each time step.

is, we evaluate the degree to which increasing rates of change in community memberships impacts on the ability of the two strategies to correctly identify communities. Based on the generation parameters given for *Set 1* in Table 1, we generated six additional dynamic synthetic networks, each containing 15,000 nodes and 10 time steps. In each dataset an increasing proportion of nodes was moved between communities after each time step – the proportion ranged from 5% to 50%.

Figure 9 shows the effect of this increasing volatility on both static and dynamic community finding, in terms of generalised NMI scores relative to the ground truth communities embedded in the data. For a switching rate of 5%-10%, static community finding using the MOSES algorithm achieves reasonable accuracy, while the dynamic community finding approach recovers the embedded communities almost perfectly. As the level of volatility increases, the ability of static community finding to correctly identify communities at later time steps decreases dramatically. This is perhaps unsurprising, as the presence of many intra-community edges from the earlier time steps could prove misleading. This strongly suggests that taking a static view of such rapidly changing networks will not be helpful in identifying meaningful communities.

The results shown in Figure 9 also provide us with interesting observations regarding the choice of threshold parameter $\theta$ for dynamic community finding. For the networks with a switching rate $\geq 30\%$, the choice of a conservative matching threshold ($\theta \geq 0.5$) leads to results which are worse than those provided by static community finding. In these cases we observe that an overly strict policy on matching will naturally lead to a failure to match changing but related pairs of communities between time steps – this is a consequence of the similarity measure defined in Eqn. 1. In contrast, a relaxed matching threshold ($\theta = 0.1$) leads to the almost perfect identification of the ground truth groupings. Only in the case of switching 50% of nodes between time steps do we notice a minor degradation in performance in the later time steps. This represents an extreme case – at this point effectively half of the nodes from each community have moved elsewhere after each step. These results suggest that the choice of a relatively relaxed matching threshold can serve to identify core



(a) 5% of memberships switched

(b) 10% of memberships switched

(c) 20% of memberships switched

(d) 30% of memberships switched

(e) 40% of memberships switched

(f) 50% of memberships switched

Figure 9: Performance, in terms of Normalised Mutual Information (NMI), of static and dynamic community finding methods on six synthetic networks as the level of volatility increases – *i.e.* an increasing proportion of node memberships are switched between consecutive time steps.

community structure, in dynamic networks where community structures are changing either slowly or very rapidly between time steps.

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.

### 4.5 Evaluation: Scalability

To examine the scalability of the dynamic community finding method proposed in Section 3, we used the synthetic graph generation process to produce dynamic networks of successively larger

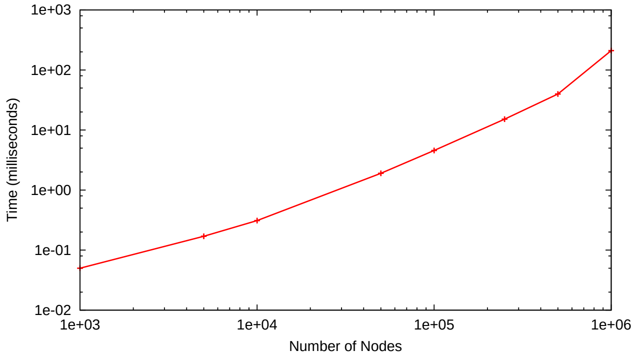


Figure 10: Plot of running time in milliseconds against number of nodes, for the proposed dynamic community finding method ($\theta = 0.3$) on synthetic dynamic graphs with 10 time steps and of increasing size from 1,000 to 1 million nodes.

sizes. We generated networks using parameter *set 1* from Table 1 to create step graphs with disjoint communities, with the membership of 5% of nodes switching from one step community to another over 10 time steps. Step communities were identified using the MOSES algorithm, and the dynamic community finding process was repeated over 10 runs.

Figure 10 shows the running times for dynamic community finding with matching threshold $\theta = 0.3$. We observe that the scaling of the dynamic community method is close to linear in the number of nodes in the graph. A dynamic network with 100k nodes and 10 time steps can be processed in 4.6 seconds – resulting in 2,739 dynamic community timelines, while a graph containing 1 million nodes can be processed in approximately 210 seconds, resulting in the discovery of over 27k community timelines. 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 threshold values $\theta \in [0.2, 0.5]$.

In these experiments, most of the computation time was due to the underlying community finding algorithm (MOSES) used to find communities in each step graph. For instance, the time taken to identify step communities on the 1 million node graph using MOSES was approximately 10 hours, in contrast to the dynamic tracking time of $\approx$ 210 seconds. Since the majority of the computation during tracking (calculating front-step community similarities and matching step communities to dynamic communities) can be performed independently, we suggest that further scope exists for improving computational performance by parallelising the matching procedure. Even with the current implementation, we can readily process dynamic graphs far larger than those that can handled by existing methods, such as those based on clique percolation (Palla *et al.*, 2007).

## 4.6 Application to Mobile Call Data

In our second evaluation, we applied the proposed method to a real mobile operator network. We analysed weekly voice call graphs over 24 consecutive weeks, each containing approximately four million unique subscribers and tens of millions of edges.

### 4.6.1 Experimental Setup

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 per week) were removed – these typically represent service numbers which can potentially obscure communities of "real" network subscribers. To improve the stability of the communities identified on individual network snapshots, we constructed longer time steps covering four weeks by aggregating the weekly graphs with no overlap. This yielded six step graphs containing approximately 3.9-4.2 million nodes and 20-26 million edges each. The mean node degree for each graph was in the range $[10.3, 12.6]$, with maximum degree range $[671, 830]$.

Since we might expect pervasive overlap in a network of this type, we again applied the MOSES algorithm (McDaid & Hurley, 2010) to each step graph to identify overlapping communities. We examined a range of relaxed matching thresholds $\theta \in [0.2, 0.5]$ for dynamic community finding, with $d = 3$ used to remove communities that were no longer active.

### 4.6.2 Discussion of Results

The running time to identify each set of step communities was approximately 30 hours using the MOSES algorithm. The number of step communities identified in each graph was relatively consistent, ranging from 502k-574k communities per step. The distribution of community sizes across time steps was similar, with the vast majority (on average $\approx 85\%$) of communities containing ten or less callers. We also observed significant overlap between step communities – on average each node was assigned to $\approx 2.5$ step communities, with a small number of high-degree nodes assigned to over 50 communities per step.

To identify meaningful dynamic communities, we naturally require that some structure persists in the network across time. In the call data, on average 86% of nodes were preserved between successive time step graphs, with an average 47% of edges being preserved. To explore this further, we examined the agreement between the sets of step communities generated by the MOSES algorithm on each step graph. This was done by calculating the generalized NMI similarity between succesive pairs of sets of step communities. The resulting NMI scores were surprisingly low, falling in the range $[0.09, 0.11]$. The low level of agreement across time steps raised the concern that coherent dynamic groups might not be identifiable using these step communities – the results described herein indicates that this is not necessarily the case.

The average running time for dynamic community finding was approximately 7–8 hours for each full set of time steps on a single core, yielding 2-3 million dynamic communities for each threshold value. As discussed in Section 4.5, the computational bottleneck for large datasets will be the time required to identify the underlying groups in the step graphs, rather than the dynamic tracking procedure itself. Also in a real-world scenario the dynamic community finding process would typically be run incrementally as new batches of data arrive, rather than repeatedly re-applying the entire process to months or years worth of data.

Table 2 provides details of the dynamic communities identified on the caller network for the range of thresholds $\theta \in [0.2, 0.5]$. Recall from Section 3.3.1 that, for a given timeline, the overall dynamic community membership is defined as the union of the set of nodes appearing in its constituent step communities. We observe that a large proportion of communities only appear in a single time step – this indicates that many communities in the call data are ephemeral and do not persist across time. However, even in the most conservative case $\approx 282k$ long-lived dynamic communities were identified (*i.e.* communities observed in two or more steps). In the case of $\theta = 0.2$ this rises to $\approx 935k$. This represents a substantial number of subscriber communities from the perspective of a mobile operator. Table 2 also shows that a large proportion of "intermittent" communities were found, which were observed in non-consecutive time step graphs.

To examine the issue of dynamic community longevity in more detail, Figure 11 illustrates the number of steps in which long-lived dynamic communities appeared. As the value of $\theta$ increases, we observe that a more conservative matching policy results in fewer matches between communities in different time steps, and consequently shorter dynamic timelines. But even in strictest case ($\theta = 0.5$), the dynamic community finding algorithm identifies $\approx 190k$ timelines observed in at least 50% of the time steps.

| $\theta$ | **Timelines** | **Long-Lived (%)** | **Intermittent (%)** |
|---|---|---|---|
| 0.2 | 2,014,651 | 46.4% | 33.8% |
| 0.3 | 2,306,976 | 27.7% | 19.8% |
| 0.4 | 2,626,672 | 17.1% | 17.4% |
| 0.5 | 2,900,921 | 9.7% | 15.7% |

Table 2: Details of all dynamic communities identified on 24 weeks of call data for matching threshold values $\theta \in [0.2, 0.5]$.

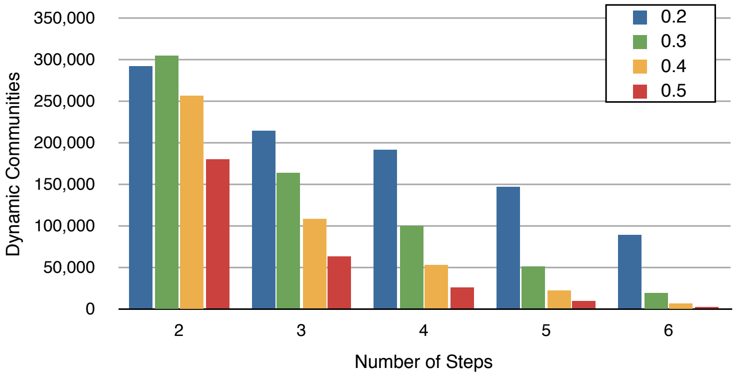


Figure 11: Number of time steps in which *long-lived* dynamic communities were observed in mobile call data covering 24 weeks, divided into six time steps, for matching thresholds $\theta \in [0.2, 0.5]$.

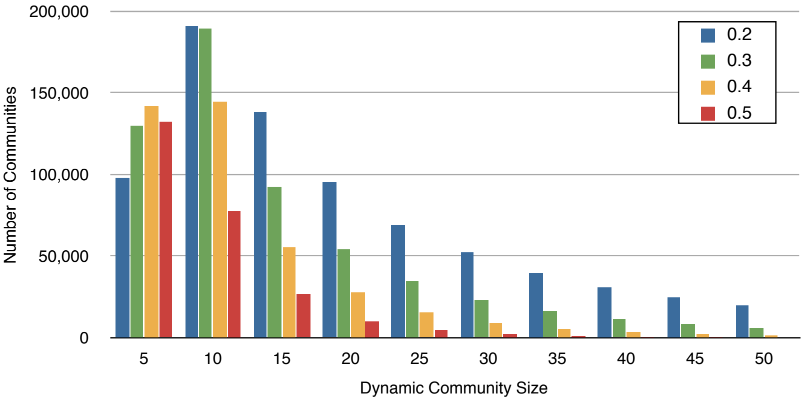


Figure 12: Distribution of long-lived dynamic community sizes for matching thresholds $\theta \in [0.2, 0.5]$.

We also examined the distribution of community sizes resulting from the dynamic analysis of the call graphs. The resulting size distributions for the dynamic communities identified by different $\theta$ values are shown in Figure 12. Note that the plot is truncated after communities of size 50 – this represents 86.7% of all dynamic communities for the most conservative case ($\theta = 0.2$) and 99.5% for the most relaxed case ($\theta = 0.5$). The majority of communities are of size 5-10 nodes, with a tendency towards slightly larger communities as $\theta$ decreases. This roughly corresponded to our prior assumptions regarding user calling patterns. These small communities tend to persist over many or all of the time steps in the data, indicating the presence of small core groups of users consistently calling one another over a six month time period.

## 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 traditional static community finding strategies which do not take temporal information into account. Additionally, we have performed an evaluation on a large real-world mobile call network over a 40 week period. On this data our proposed method uncovered a large number of dynamic communities in this network with different evolutionary characteristics, while requiring comparatively little computational overhead.

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 edges, disjoint/overlapping communities). However, an interesting avenue for further work would be to integrate the matching-based tracking procedure with a scalable overlapping community finding algorithm, so that information from dynamic community timelines can be used to seed or direct community detection in the next time step. This could potentially lead to improved temporal smoothing, yielding a higher degree of consistency in communities found across time steps.

Given the large number of communities which may be identified on large dynamic networks such as mobile operator call graphs, another important area of research concerns the development of an appropriate, scalable visualization technique for the output of the dynamic community finding process. We intend to extend initial work in this area by Rosvall & Bergstrom (2010) to deal with the case of large networks with substantial numbers of small communities.

## Acknowledgements

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

## References

Ahn, Y.Y., Bagrow, J.P. & Lehmann, S. (2010). Link communities reveal multiscale complexity in networks. *Nature*, **466**, 761–764.

Asur, S., Parthasarathy, S. & Ucar, D. (2007). 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*, 921, ACM.

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

Chakrabarti, D., Kumar, R. & Tomkins, A. (2006). Evolutionary clustering. In *Proc. 12th ACM SIGKDD International conference on Knowledge Discovery and Data mining*, 554–560, ACM.

Chi, Y., Song, X., Zhou, D., Hino, K. & Tseng, B. (2007). Evolutionary spectral clustering by incorporating temporal smoothness. In *Proc. 13th ACM SIGKDD International conference on Knowledge Discovery and Data mining*, 153–162, ACM.

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

Duan, D., Li, Y., Jin, Y. & Lu, Z. (2009). Community mining on dynamic weighted directed graphs. In *Proc. 1st ACM international workshop on Complex networks meet information & knowledge management*, 11–18, ACM, New York, NY, USA.

Dudoit, S. & Fridlyand, J. (2003). Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, **19**, 1090–1099.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, **486**, 75–174.

Greene, D. & Cunningham, P. (2009). 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)*.

Greene, D., Doyle, D. & Cunningham, P. (2010). Tracking the evolution of communities in dynamic social networks. In *Proc. International Conference on Advances in Social Networks Analysis and Mining (ASONAM'10)*, IEEE.

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

Kuhn, H.W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, **2**, 83–97.

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

Lancichinetti, A., Fortunato, S. & Kertész, J. (2009). Detecting the overlapping and hierarchical community structure of complex networks. *New J. Phys*, **11**, 033015.

Lee, C., Reid, F., McDaid, A. & Hurley, N. (2010). Detecting highly overlapping community structure by greedy clique expansion. In *Proc. 4th Workshop on Social Network Mining and Analysis*.

Lin, Y.R., Chi, Y., Zhu, S., Sundaram, H. & Tseng, B.L. (2008). Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proc. 17th International conference on World Wide Web (WWW'08)*, 685–694.

McDaid, A. & Hurley, N. (2010). Detecting highly overlapping communities with Model-based Overlapping Seed Expansion. In *Proc. International Conference on Advances in Social Networks Analysis and Mining (ASONAM'10)*, 112–119, IEEE.

Palla, G., Barabási, A. & Vicsek, T. (2007). Quantifying social group evolution. *Nature*, **446**, 664–667.

Rosvall, M. & Bergstrom, C.T. (2010). Mapping change in large networks. *PLoS ONE*, **5**, e8694.

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

Tantipathananandh, C., Berger-Wolf, T. & Kempe, D. (2007). A framework for community identification in dynamic social networks. In *Proc. 13th ACM SIGKDD International conference on Knowledge Discovery and Data mining*, 717–726, ACM.

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