Towards Social-aware Routing in Dynamic Communication Networks

Thang N. Dinh, Ying Xuan, My T. Thai
Computer & Information Science & Engineering Department,
University of Florida, Gainesville, FL 32611.
Email:{tdinh, yxuan, mythai@cise.ufl.edu}

***Abstract*—Many communication networks such as Mobile Ad Hoc Networks (MANETs) involve in human interactions and exhibit properties of social networks. Hence, it is interesting to see how knowledge from social networks can be used to enhance the communication processes. We focus on the use of identifying modular structure in social networks to improve the efficiency of routing strategies. Since nodes mobility in a network often alters its modular structure and requires recomputing of modules from scratch, updating the modules is the main bottleneck in current social-aware routing strategies where nodes often have limited processing speed. Towards real-time routing strategies, we develop an adaptive method to efficiently update modules in a dynamic network in which a novel compact representation of the network is used to significantly reduces the network size while preserving essential network structure.**

***Index Terms*—Social Networks, Routing, Modular Structure, Dynamic Networks, Adaptive Algorithm**

I. Introduction

Mobile Ad hoc Networks (MANETs), dynamic and self-configuring networks of mobile devices, have been widely used in practice and become a popular research topic. Due to mobile nodes, unstable links and time-dependent topology of a MANET, designing an efficient routing scheme on MANETs has emerged as the most challenging but profitable problem.

Existing routing protocols can be classified into two categories: proactive *Table-Driven* and reactive *On-Demand*. Table-driven protocols [1] [2] employ routing tales, which contain the best approximated shortest paths between nodes, and direct messages accordingly. Whenever mobility occurs the routing tables are required to be updated and propagated throughout the network to maintain a consistent network view. This results in a huge amount of communication overhead and bandwidth consumption, especially when the network topology changes rapidly.

By contrast, reactive *On-Demand* schemes, which only initiate the path discovery process on receiving message delivery requests (Figure 1), obtain better performance and thus seize plenty of attraction since the last decade [3] [4]. Naive on-demand forwarding schemes in which nodes flood duplicated copies of messages to neighbors, can provide the highest delivery ratio. However, they inevitably raise the problem of buffer overflow and bandwidth exhaustion. Therefore, to devise a routing scheme with the small message redundancy, small delivery latency, as well as reasonable delivery ratio, is the main objective of the recent researches toward this problem.

Among various path discovery strategies proposed in this scope, *Social Patten* based algorithms have shown their great potentials recently. The reason is that while MANETs serve as communication networks among mobile devices, they often exhibit properties of social networks when the devices (e.g., mobile phones) are carried by human beings, and then the communications indicate human interactions. In contrast with the rapid changes of nodes' locations in MANETs, the changes of the social relationships among these device carriers are much less frequent. Moreover, recent real traces for MANETs [5] [6] reveal that there exist groups of individuals, where a majority of individual interactions take place within the same group. This resembles the *Community Structure* (also known as modular structure) in social network, which are grouping of nodes into modules with dense connections inside each module but fewer connections crossing modules.

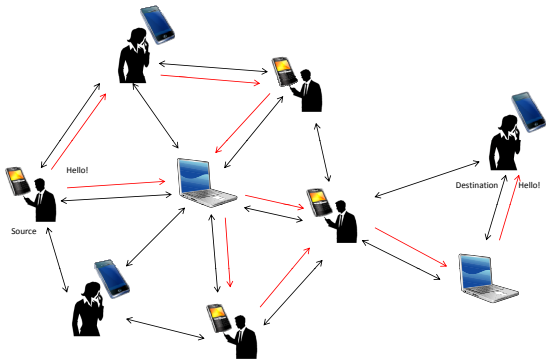


Fig. 1. Message forwarding in a MANET

Multiple routing strategies [7] [8] [9] [10] based on the discovery of modular structure have provided significant performance enhancement compared to traditional schemes. Specifically, with the modular structure of the network detected, only the nodes within the same module will be selected as intermediate nodes [7], which avoids the unnecessary message forwarding through nodes belonging to other modules. In this line of works, the forwarding strategy is determined as soon as the modular structure of the network is identified. For a complete view of algorithms to identify modular structure in networks, we refer to the comprehensive survey in [11].

Together with the network topology, the modular structure

also changes through time, although with a slower speed. Hence, the routing performance would also degrade if routing are performed using a fixed and obsolete modular structure. Therefore, *how to accurately and promptly identify modular structures in dynamic network topologies* is the kernel problem to be solved in this paper.

In spite of the fruitful existing modular structure detection schemes, to enhance both the detection accuracy and efficiency in dynamic networks is non-trivial. In previous approaches, [12]–[14] network modules are identified separately at each time point by running modular structure identification algorithms many times. The main disadvantage of this approach is that recomputing modular structure of the network from scratch may result in *prohibitive computational costs*, particularly in case of nodes with limited processing speed. In addition, it may be *infeasible* since it requires global information. Moreover, recomputing modular structure independently may lead to substantial and unexpected variation in comparison to the previous modular structure which would cause mass propagation overhead from these updates.

To overcome the above limitations, we propose a graph-based approach in which we use modular structure found in previous states of the network as a guide to adaptively identify modules in the next state. Adaptively updating modules in dynamic networks not only avoids recomputing them from scratch for each time point but also produces consistent modular structures with minimum changes in the routing tables. The key in our approach to reduce the running time and computational cost is a compact network representation with significantly smaller size such that the modular structure of compact networks is as same as that of the original networks. The structure information is embedded in this compact representation to allow detection of modules at next time point. Several experiments with real data sets show that our approach performs extremely well, especially in term of the running time. The updating process only need the information about the changing part of the network that can be accumulated whenever mobility occurs. In other words, our approach can be implemented in a distributed manner which we will investigate in further work.

The rest of the paper is organized as follows. Section II presents the preliminaries and the problem definition. In section III, we discuss the construction of compact network representation together with detailed analytical results. Section IV analyzes how network structure affected when elemental changes such as adding, removing new nodes, links happen and presents an adaptively updating algorithm. Section V shows the experimental evaluation. Finally, the conclusion and future work are discussed in Section VI.

II. Preliminaries and Definitions

In our network devices/nodes are carried by people and relationships among people are reflected via communications links between nodes. We model the network using a weighted and undirected graph $G = (V, E)$. Each vertex $v \in V$ represents a mobile node (or a person who carries the device) in the network. Each edge $(u, v) \in E$ is associated with a weight $w(u, v)$ that measures how often two nodes u, v in the each others' communication ranges.

As people form communities, nodes in the network can be partitioned into modules. A modular structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ is a division of network into disjoint sets of nodes C_i. To quantify the strength of $\mathcal{C}$, we use the most widely accepted quantitative measurement [15] *modularity* $\mathcal{Q}$ which is defined as follows:

$$\mathcal{Q}(\mathcal{C}) = \sum_{i=1}^{k} \left(\frac{\|C_i, C_i\|}{m} - \frac{\text{vol}^2(C_i)}{4m^2} \right) \quad (1)$$

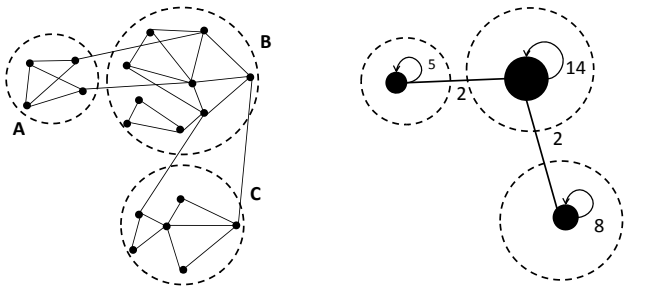
where $\|X, Y\|$ is the total weights of links whose one end belongs to X and the other belongs to Y; $\text{vol}(C_i)$ is the total weight of links that have at least one end-point in C_i; m is the total weights of links in the network. The modularity measure the difference between the fraction of links that fall within modules and the expected fraction of links that fall within modules in a random network with same nodes' degrees. The modularity can be applied for both unweighted and weighted networks which is a plus comparing to other approaches. Due to the modularity definition, if we scale all the edge weights by a same factor, the modular structure of the network and corresponding modularity value will not be affected. This is because the weights of edges (u, v) at a given point can be simply set to the sum of all durations that two nodes u, v encounter instead of theirs frequency of meeting.

Assume that the state of the network is given by $G^0 = (V^0, E^0)$ at the beginning time t_0 and it changes into $G^i = (V^i, E^i)$ at time t_i. We say the network evolves from G^{i-1} to G^i during the $[t_{i-1}, t_i]$ interval. Denote ΔV^i the set of added or removed nodes when the network evolves evolves from G^{i-1} to G^i i.e. the symmetric difference $V^i \ominus V^{i-1}$. Similarly, we denote $\Delta E^i = E^i \ominus E^{i-1}$ the set of added or removed links when the network evolves from G^{i-1} to G^i. Given the network $G^{i-1} = (V^{i-1}, E^{i-1})$ at time t_{i-1} and the changes $(\Delta V^i, \Delta E^i)$ in the network between t_{i-1} and t_i, one can easily deduce $G^i(V^i, E^i)$ from the formulas $V^i = V^{i-1} \ominus \Delta V^i$ and $E^i = E^{i-1} \ominus \Delta E^i$.

Given a network and its sequences of changes over time: $G^0 = (V^0, E^0), (\Delta V^1, \Delta E^1), \ldots, (\Delta E^i, \Delta E^i), \ldots$, our contribution is to devise an online algorithm to efficiently detect the modules in the network at every time point without recomputing them from scratch, i.e. adaptively update modules over time. Our algorithm consists of two main steps: (1) compress the network into a compact representation and (2) run module identification algorithm on the compact representation to update the modular structure. We next present our compact representation method.

III. Compact Representation of a Network

Given a network $G = (V, E)$ and its modular structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ where each disjoint subset of nodes C_i is called a module. Our goal is to construct a new network $G' = (V', E')$ with equivalent structure to G but contains a significantly smaller number of nodes and links. Each node



(a) Original network (b) Compact representation of the network

Fig. 2. Compact representation of a network

$x_i \in V'$ corresponds to each module $C_i \in G$. We add an edge (x_i, x_j) into E' for every two adjacent modules C_i, C_j i.e. there is at least one edge $(u, v) \in E$ such that $u \in C_i, v \in C_j$. The weight of (x_i, x_j) is set to the sum of weights of all edges crossing C_i, C_j. Moreover, for each module C_i, we add a self-loop (x_i, x_i) into E' with an weight equals to the sum of all edges with both ends belong to C_i. Note that under this construction, each node x_i is a module in G', i.e. the corresponding modular structure of G' will be $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_k\}$ where $C'_i = \{x_i\}$. Figure 2 gives an example of compact representation of a network with 3 modules.

The algorithm to construct the compact representation is presented in the Algorithm 1. The notation $\text{mb}(u)$ in the line 11 gives the index of the module that contains u as a member. Note that we will use this compact representation to adaptively update the modular structure. Thus the compact representation must preserve the modular structure of the original network G which we prove next.

Algorithm 1 Compact representation of a network

1: **Input**: $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.
2: **Output**: Compact representation $G' = (V', E')$.
3: Initialize $G' = (V', E') = (\phi, \phi)$, $\mathcal{C}' = \phi$
4: **for** each module C_i in $\mathcal{C}$ **do**
5: $V' \leftarrow x_i$
6: $C'_i = \{x_i\}$, $\mathcal{C}' \leftarrow C'_i$
7: $E' \leftarrow (x_i, x_i), w(x_i, x_i) = \|C_i, C_i\|$
8: **end for**
9: $\forall i, j : w(x_i, x_j) \leftarrow 0$
10: **for all** $(u, v) \in E$ **do**
11: $i = mb(u)$, $j = mb(v)$
12: $w(x_i, x_j) = w(x_i, x_j) + w(u, v)$
13: **end for**
14: **return** $G' = (V', E')$

A. Structure Preservation

Lemma 1: The compact representation $G' = (V', E')$ of $G = (V, E)$ preserves the modularity, i.e $\mathcal{Q}(\mathcal{C}') = \mathcal{Q}(\mathcal{C})$.

Proof: From the definition (1), we have:

$$\mathcal{Q}(\mathcal{C}') = \sum_i \left(\frac{\|C'_i, C'_i\|}{m'} - \frac{\text{vol}^2(C'_i)}{4m'^2} \right) \quad (2)$$

By the construction of G', we have :

$$\begin{aligned}
\|C'_i, C'_i\| &= w(x_i, x_i) = \|C_i, C_i\| \quad (3)\\
\text{vol}(C'_i) &= 2\|C'_i, C'_i\| + \sum_{j \neq i} \|C'_i, C'_j\| \\
&= 2\|C_i, C_i\| + \sum_{j \neq i} \sum_{u \in C'_i, v \in C'_j} w(u, v) \\
&= 2\|C_i, C_i\| + \sum_{j \neq i} \|C_i, C_j\| \\
&= \text{vol}(C_i) \quad (4)\\
m' &= \frac{1}{2} \sum_i \text{vol}(C'_i) = \frac{1}{2} \sum_i \text{vol}(C_i) = m \quad (5)
\end{aligned}$$

From (2), (3), (4), and (5), it follows that $\mathcal{Q}(\mathcal{C}') = \mathcal{Q}(\mathcal{C})$ ■

Theorem 1: If modular structure $\mathcal{C}$ maximizes the modularity in $G = (V, E)$, then $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_k\}$ obtained in the construction of compact representation $G' = (V', E')$ using Algorithm 1 will be the modular structure with maximum modularity in G'.

Proof: Let $\mathcal{C}^* = \{C^*_1, C^*_2, \ldots, C^*_t\}$ be the modular structure with maximum modularity in G'. We will prove that $\mathcal{Q}(\mathcal{C}^*) = \mathcal{Q}(\mathcal{C}')$. Since each node in G' corresponds to a module in G, each $C^*_j \in \mathcal{C}^*$ can be decomposed into a union of modules in $\mathcal{C}'$. There exists a partition $I = \{I_1, I_2, \ldots, I_t\}$ of $\{1, 2, \ldots, k\}$ such that $C^*_j = \bigcup_{i \in I_j} C'_i$, $\forall j = 1..t$.

Construct a modular structure $\mathcal{C}'' = \{C''_1, C''_2, \ldots, C''_t\}$ of $G = (V, E)$ corresponding to $\mathcal{C}^*$ of G' by assigning $C''_j = \bigcup_{i \in I_j} C_i$, $\forall j = 1...t$. Using similar proof showed in Lemma 1, we can easily obtain that $\mathcal{Q}(\mathcal{C}'') = \mathcal{Q}(\mathcal{C}^*)$.

Since $\mathcal{Q}(\mathcal{C}') = \mathcal{Q}(\mathcal{C}) \geq \mathcal{Q}(\mathcal{C}'')$ (Lemma 1) and $\mathcal{Q}(\mathcal{C}'') = \mathcal{Q}(\mathcal{C}^*) \geq \mathcal{Q}(\mathcal{C}')$. We have $\mathcal{Q}(\mathcal{C}') = \mathcal{Q}(\mathcal{C}^*)$. ■

B. Size of the Compact Representation

Now, we show that the size of the compact representation G' is significantly smaller than that of the original network G. Note that we can measure not only the strength of the modular structure with modularity but also the strength of each module in the network. The modularity of a module C is defined as $\mathcal{Q}(C) = \frac{\|C,C\|}{m} - \frac{\text{vol}^2(C)}{4m^2}$.

Lemma 2: If there is a module with negative modularity in a modular structure of a network $G = (V, E)$, we can construct a new modular structure with higher modularity in which all modules have non-negative modularity.

Proof: Assume that there exists a module C in the modular structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ with $\mathcal{Q}(C) = \frac{\|C,C\|}{m} - \frac{\text{vol}^2(C)}{4m^2} < 0$. For each module $C \in \mathcal{C}$, denote $\mathcal{N}(C)$ the set of modules adjacent to the module C.

We prove that there is an module C_i adjacent to C so that merging C and C_i will result in a new modular structure with higher modularity.

We prove by contradiction. Assume that merging C with any of its adjacent modules will not increase the modularity. We calculate the change in modularity if we move a group of nodes S from its module $C_a \supseteq S$ to a new module C_b ($C_a \cap$

$C_b = \phi)$:

$$\Delta\mathcal{Q}^{\mathcal{S}}_{C_a,C_b} = \frac{1}{m}\left(\|C_b, S\| - \|C_a\backslash S, S\|\right) - \frac{1}{2m^2}\mathrm{vol}(S)\left(\mathrm{vol}(C_b) - \mathrm{vol}(C_a\backslash S)\right) \quad (6)$$

Following (6) we have for all $C_i \in \mathcal{N}(C)$:

$$\Delta\mathcal{Q}^{C}_{C,C'} = \frac{1}{m}\|C', C\| - \frac{1}{2m^2}\mathrm{vol}(C')\mathrm{vol}(C) \leq 0$$

Take the sum over all $C' \in \mathcal{N}$, we obtain:

$$\sum_{C_i\in\mathcal{N}} \frac{1}{m}\|C', C\| - \sum_{C_i\in\mathcal{N}} \frac{1}{2m^2}\mathrm{vol}(C)\mathrm{vol}(C_i) \leq 0$$

$$\Leftrightarrow \frac{1}{m}\Phi\Big(\bigcup_{C'\in\mathcal{N}}(C', C) - \frac{1}{2m^2}\mathrm{vol}(C)\mathrm{vol}\Big(\bigcup_{C'\in\mathcal{N}}(C')\Big) \leq 0$$

$$\Rightarrow \frac{1}{m}(\mathrm{vol}(C) - \|C, C\|) - \frac{1}{2m^2}\mathrm{vol}(C)\mathrm{vol}(V\backslash C) \leq 0$$

$$\Rightarrow 2\mathcal{Q}(C) \geq 0$$

Thus we obtain a contradiction.

Hence, for any module in the modular structure with negative modularity, we can merge it with one of its neighbors to have a new modular structure with higher modularity. Repeat the merging until there are no modules with negative modularity or there is only one module left in the modular structure noting that after merging, the number of modules decrease by one and we have at most $|V|$ modules at the beginning. In the first case, all modules will have non-negative modularity. In the second case, it is easy to check that the modularity of a modular structure with one module is zero. This yields the lemma. ■

Theorem 2: The size of a compact representation constructed by the Algorithm 1 is at most half of that in the original network.

Proof: Remove all isolated nodes from the network as they have no contribution to the modularity. We prove that each module in the network should contain at least two nodes. Assume that there exists a module C_i that contains only a single node. Because $\|C_i, C_i\| = 0$ and $\mathrm{vol}(C_i) > 0$ the modularity $\mathcal{Q}(C_i) = \frac{\|C_i,C_i\|}{m} - \frac{\mathrm{vol}^2(C_i)}{4m^2} < 0$ which is contradicted to the Lemma 2.

Since, the number of nodes in the compact representation equals the number of modules in the original network. This yields the proof. ■

In some social networks [16]–[18], the distribution of the module size s follows the power-law form $P(s) = c.s^{-\alpha}$ for some constant $\alpha \approx 2$ at least for a wide range of s. If the network has n nodes then the average number of modules $\overline{k}$ will be

$$\overline{k} = \frac{n}{\overline{s}} = \frac{n \times \sum_{s=2}^{n} c.s^{-\alpha}}{\sum_{s=2}^{n} c.s^{-\alpha}.s} = \frac{n \times \sum_{s=2}^{n} s^{-\alpha}}{\sum_{s=2}^{n} s^{1-\alpha}} \quad (7)$$

where $\overline{s}$ is the average module size of the network.

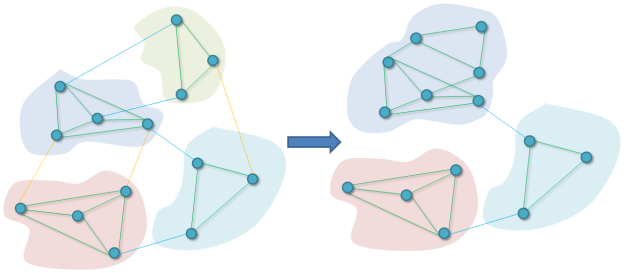


Fig. 4. Removing inter-modules links usually results in more separated modules. However, in this case removing yellow inter-modules links make modules merged.

When $\alpha = 2$ the denominator in the Equation 7 can be approximated by $\ln n$ and the sum $\sum_{s=2}^{n} c.s^{-\alpha}$ in the numerator is bounded by a constant. Hence, the average size of the compact representation is upper-bounded by $O(\frac{n}{\log n})$ which guarantee a speed up factor of at least $O(\log n)$ for our adaptive algorithm presented in next section.

IV. Adaptively Update Evolving Modules

We first discuss about how changing in network topology caused by nodes' mobility affects the modular structure of the network. Assume that we have a network $G = (V, E)$ and its modular structure $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$. The connection inside a module is denser than connection between modules. We use the terms intra-module links to refer to links with both ends belong to a module and inter-modules links to refer to ones crossing some modules. Intuitively adding intra-module links or removing inter-modules links will make modules "stronger" and the modular structure more "clear'. Vice versa, removing intra-module links or adding inter-modules links will make the existing module structure less clear.

In fact, removing inter-modules links may cause merging of modules as shown in Figure 4. When two modules have less "distraction" caused by other modules, they become more attractive to each other and can be combined as one. Other

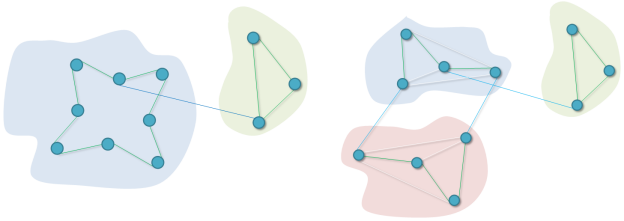
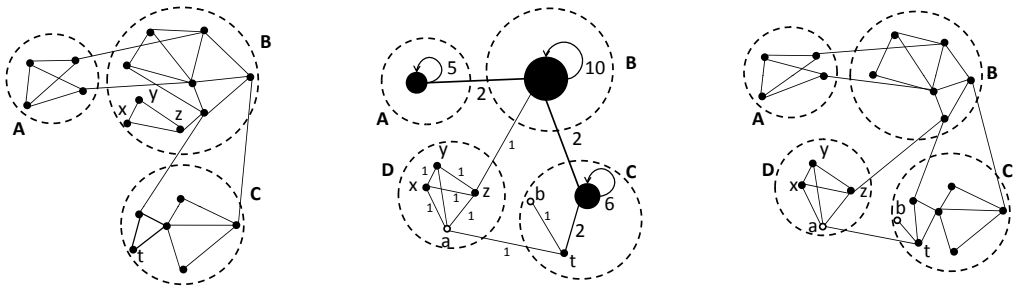


Fig. 5. Adding intra-module links usually strengthen the module. However, in this case adding 6 white links into the light blue module make it split into two.

possible striking fact is that splitting of the module when adding intra-links. Figure 5 show an example in that adding 6 more white links make the module split into two smaller ones. From the quantitative aspect, one can verify that adding an intra-module link will not always increase the modularity of the module. Assume that we have a module C_i that cannot



(a) Original modular structure

(b) Compact representation after new nodes join in

(c) New modular structure

Fig. 3. Illustration for Algorithm 2: (a) The original modular structure consists of 3 modules (b) The compact representation of the network after adding new nodes a, b and connecting a to x, y, z, t and b to t. Then, x, y, z, t are moved out of their modules. (c) The new modular structure derived from the modular structure of the compact representation

be divided further in order to increase the modularity. Among all possible bisections of C_i, consider $\{X, Y = C_i \backslash X\}$ that X, Y are the most loosely connected e.g. X, Y are obtained through the sparsest cut in the subgraph induced by C_i. In the case of new links crossing X and Y, they will enhance the local structure of C_i. Otherwise, adding new links that both ends belong to X or Y will make the connection inside either subcomponent X or Y stronger but weaken the structure of C_i, thus leading to the splitting of X and Y. Similar observations can be seen for adding and removing vertices.

The behavior of splitting modules makes it extremely challenging to adaptively update evolving modules, especially on the compact representation where each module is represented by only one node. The main difficulty is that the compact representation itself cannot reflect all the phenomena in the original network. For example, we cannot represent splitting of modules or moving of nodes into other modules using the compact representation. To overcome this limitation, we exclude nodes that are suspected to change their memberships from their modules and assign them to new singleton modules so that they can freely join other modules in the compact representation.

A. Algorithm

The algorithm for *Modules Identification in Evolving Networks* (MIEN) is presented in Algorithm 2. The presented algorithm is a general approach such that it allows the use of any existing algorithm $\mathcal{A}$ on modules identification. The selection of algorithm $\mathcal{A}$ is a trade off between running time and quality of detected modular structure. For example, if we use extremal optimization methods in the place of $\mathcal{A}$, the obtained modular structure will be nearly optimal, however, the running time will be hardly tolerable. Towards routing purpose, we sacrifice a little in the quality to guarantee a fast response and low-computing solution. We select $\mathcal{A}$ as the well-known CNM algorithm [17] which is capable of detecting modular structure of the network in $O(n \log^2 n)$ where n is the network size.

The sketch of the algorithm is as follows. After identifying the initial modular structure of $G^0 = (V^0, E^0)$ with algorithm $\mathcal{A}$, our algorithm involves in three main steps at each state t of the network: (1) Combine changes in the network at time t into the previous compact representation to produce a temporary compact representation; (2) Apply algorithm $\mathcal{A}$ on that temporary representation to obtain its modular structure; (3) Interpret the obtained modular structure of the compact representation to get the actual modular structure of the network.

We update the compact representation according to the given change $(\Delta V^t, \Delta E^t)$ as in lines 8 to 19. We need to handle four type of changes: removing of nodes ($V^{t-1} \cap \Delta V^t$), adding new nodes ($\Delta V^t \backslash V^{t-1}$), removing links ($E^{t-1} \cap \Delta E^t$) and adding new links ($\Delta E^t \backslash E^{t-1}$). Let S denote the set of nodes that are likely changing their modules. S consists of newly removed or added nodes or the ones incident to recently added or removed links. We move all nodes in S out of theirs module, assign each of them to a new singleton module in the compact representation, and update the weights of links incident to those new nodes as shown in lines 12 to 15.

After the changes in links and nodes are fully incorporated into the compact representation, we run the algorithm $\mathcal{A}$ again on the modified compact representation $G' = (V', E')$ to obtain the module structure of G'.

Finally, we use the obtained modular structure in $G' = (V', E')$ to get the modular structure in $G^t = (V^t, E^t)$. Since a node in the compact representation G' represents a module in the network, a module in G' can be seen as an union of modules in the original network. For example, if a module in G' consists of two nodes x_i and x_j, then the corresponding module in G will be the union of $C_i^{t-1} \setminus S$ and $C_j^{t-1} \setminus S$ where S is the set of newly added/removed nodes or incident to changed links defined at line 7. In the case x_i , x_j represent single nodes in G, we simply merge those nodes into a module.

Figure 3 shows an example of updating the compact representation when changes occur. At the beginning, the static algorithm $\mathcal{A}$ detects 3 modules in the network as shown in Figure 3(a). In Figure 3(b) the compact representation changes accordingly when the network has two new nodes a, b, marked with white circles, and new links $(a, x); (a, y); (a, z); (a, t); (b, t)$. For links with unit weights

we ignore the links' labels for simplicity. Note that x, y, z, t are moved out of their original modules as they are incident to new links so that they can join other modules. Running $\mathcal{A}$ on the compact representation gives us a new 4 modules as shown in Figure 3(b). In Figure 3(c), we run algorithm $\mathcal{A}$ directly on the new network instance (recomputing modules from scratch) and obtain 4 modules which are exactly the same with the one obtained on the compact representation.

B. Complexity

The Algorithm 2 includes two parts: maintaining the compact representation and running the algorithm $\mathcal{A}$ on the compact representation. Since constructing the compact representation of the network at time t^0 using Algorithm 1 takes $O(|V^0| + |E^0|)$ and updating the compact representation at time t also takes linear time in term of the changes in the network $(\Delta V^t, \Delta E^t)$, the first part can be done in linear time. Hence, the first part can be done in linear time

The running time of the second part depends on the selection of the algorithm $\mathcal{A}$ and the size of the compact network representation. The algorithm $\mathcal{A}$ can be chosen as fast as $O(n \log^2 n)$ in the case of CNM. The size of the compact representation is at most the number of modules in the original network that is small in comparison to the original network size. Moreover, updating the compact representation can introduce at most vol(S) new nodes and links into the compact representation. Hence, at each time t we only need to run the algorithm $\mathcal{A}$ on a compact network that size is proportional to the amount of changes $|\Delta V^t| + |\Delta E^t|$ instead of running $\mathcal{A}$ on the complete network $G^t = (V^t, E^t)$.

In the worst case, when all nodes are excluded from their modules, i.e. $S = V^t$, the compact representation will be exactly the original network and the complexity will equal that of the algorithm $\mathcal{A}$ adding the overhead $O(|V^t| + |E^t|)$ for updating the compact representation. Hence, if the network changes too much since the last time point, we should divide the long evolving duration into smaller ones, but not too small due to the overhead of updating the compact representation and running $\mathcal{A}$.

V. Experiments

We conducted several experiments to evaluate the performance of routing based on dynamic modular structure, compared to that using a fixed modular structure. In addition, we show that our algorithm to identify dynamic modules are much more efficient than re-computing modular structure from scratch.

A. Performance of Forwarding Strategies Using Dynamic Modular Structure

We use a synthetic scenario to measure the performance of different forwarding strategies. In our scenario, we assume that 47 students in a class A and 90 students in a class B carry mobile devices and form a MANET for sending messages. Students in the same class meet each other more often than that from different classes. Specifically, a student meet another student in the same class with probability 0.8 and a student in the other class with probability 0.2. After the first half of the experiment duration, the students in class B are split into two smaller classes due to its large size. New classes called C and D have 44 and 46 students respectively. To replicate the properties of social networks, we

Algorithm 2 MIEN - Modules Identification in Evolving Networks

Input: $G^0 = (V^0, E^0), (\Delta V^1, \Delta E^1), \ldots, (\Delta V^s, \Delta E^s)$
Output: Modular structure $\mathcal{C}^1, \mathcal{C}^2, \ldots, \mathcal{C}^s$
Find the modular structure $\mathcal{C}^0 = \{C_1, C_2, \ldots, C_{l_0}\}$ of $G^0 = (V^0, E^0)$ using the selected algorithm $\mathcal{A}$.
$G' = (V', E') \leftarrow$ Compact representation of $(G^0, \mathcal{C}^0)$ using Algorithm 1.
for $t \leftarrow 1$ to s **do**
  Let $\mathcal{C}^{t-1}$ be the modular structure of the network at time $t-1$ and $\mathcal{C}'$ be the modular structure of the compact representation $G' = (V', E')$
  Let $S \leftarrow (\Delta V^t \cup \{u |\ \exists (u, v) \in \Delta E^t\}) \cap V^t$
  for all $C_i \in \mathcal{C}^{t-1}$ **do**
    $C_i = C_i \setminus S$
    Update weight $w(x_i, x_i)$ in G'
  end for
  for all $u \in S$ **do**
    Create a new module $C'_{mb(u)} = \{x_{mb(u)}\}$ in G'
    Compute weights of links from $x_{mb(u)}$ to its neighbors in G'
  end for
  for all $(u, v) \in \Delta E^t$ **do**
    $\Delta_{u,v} = \begin{cases} +w(u,v) & (u,v) \in \Delta E^t \setminus E^{t-1} \\ -w(u,v) & (u,v) \in E^{t-1} \setminus \Delta E^t \end{cases}$
    $i \leftarrow mb(u),\ j \leftarrow mb(v)$
    $w(x_i, x_j) = w(x_i, x_j) + \Delta_{u,v}$
  end for
  Run $\mathcal{A}$ on $\mathfrak{C} \leftarrow \mathcal{A}(G') = \{C'_1, C'_2, \ldots, C'_{l_t}\}$
  Get modular structure $\mathcal{C}^t = \{C^t_1, C^t_2, \ldots, C^t_{l_t}\}$ from $\mathfrak{C}$.
  $G' = (V', E') \leftarrow$ Compact representation of $(G^t, \mathcal{C}^t)$
end for

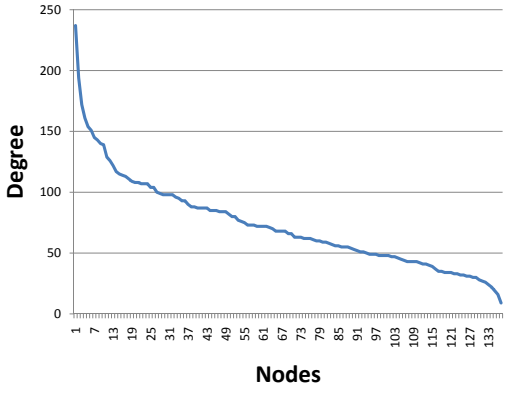


Fig. 6. Distribution of nodes' degrees

generate meeting between students following the *preferential attachment principle* [19]. The distribution of node degrees is shown in Figure 6 where the degree of a node equals to the times that it meets other nodes in the network. We evaluate

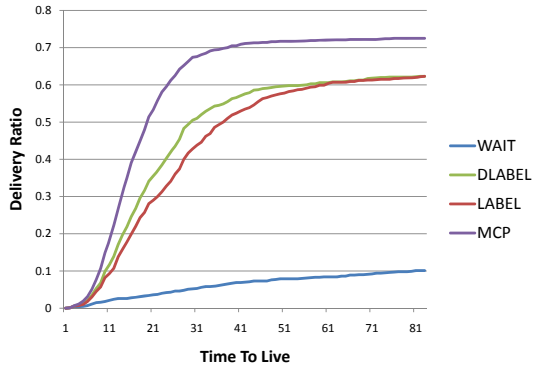


Fig. 7. Message Delivery Ratio

the following four forwarding algorithms:

- WAIT: The source node waits until it meets the destination and sends the message.
- FLOODING: A node keeps forwarding the message to every node it meets until the maximum number of hops reached.
- LABEL: A node forwards the message to nodes in the same module with the destination [7]
- DLABEL: The version of the label algorithm utilizing dynamic modular structure identified by our algorithm. Our algorithm correctly identifies modular structure of the network before and after the splitting of students in the class B.

For each algorithm, we measure:

- Delivery Ratio: The proportion of messages that have been delivered successfully over the total number of created messages.
- Average-Delivery-Time: Average time needed for a message to be delivered. Undelivered messages will not be considered.
- Redundancy: The average number of duplicated messages for each originally created message.

Although WAIT and FLOODING are naive forwarding strategies, they provide lower bound and upper bound for delivery-ratio, delivery-time and message redundancy as we will discuss in next paragraphs.

We generate 1000 messages sending requests uniformly distributed throughout the experiment duration. To avoid the circulating of duplicate messages in the network, we control forwarding algorithms by three main parameters: *hop-limit*, *time-to-live*, and *max-copies*. A message in the network cannot travel more than *hop-limit* hops or exist in the network longer than *time-to-live*, otherwise it will be discarded. A node can not forward more than *max-copies* of a same message to other nodes. Through experiments we set both *hop-limit* and *max-copies* to 5 since it is the smallest value to make the FLOODING algorithm achieve the maximum possible delivery ratio.

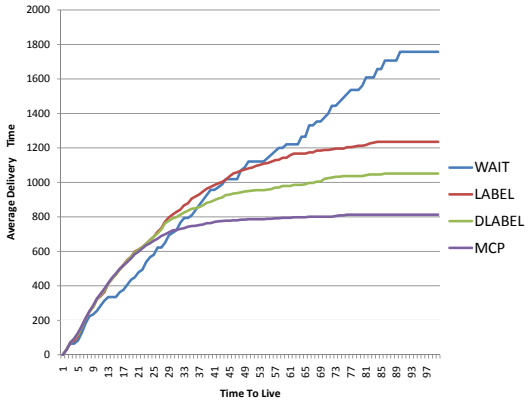


Fig. 8. Average Delivery Time

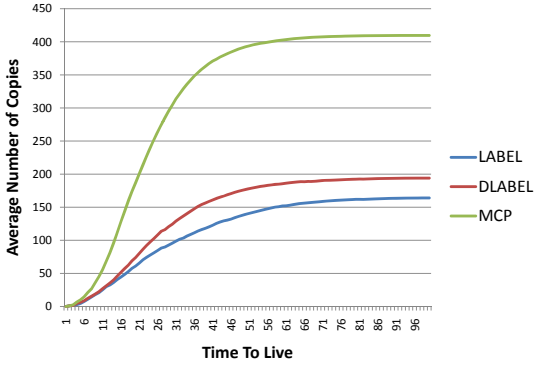


Fig. 9. Average Number of Duplicate Messages

Figure 7 shows the delivery-ratio at different values of *time-to-live*. The value of *time-to-live* is measured in percentages of the experiment duration. DLABEL has a better delivery-ratio than that of LABEL and both of them approach the best delivery-ratio of FLOODING when *time-to-live* increases.

DLABEL also gets messages delivered faster than LABEL as shown in Figure 8. The reason is that the fixed modular structure used in LABEL can get messages forwarded to wrong modules if the destinations change their modules within the experiment duration. Thanks to more accurate modular structure, the gap between DLABEL and FLOODING is only half of that in the case of LABEL.

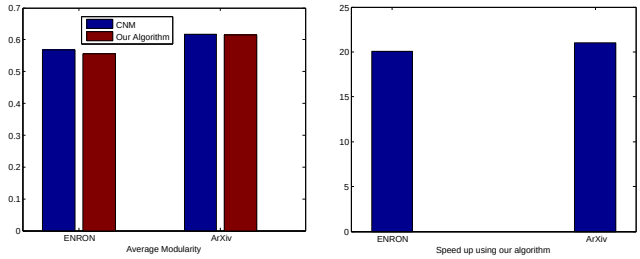
Despite of the worst delivery-ratio, the WAIT strategy has the lowest number of duplicate messages (at most one for every message created) . Since the number of duplicated messages for WAIT is too small we remove it from the graph of duplicated messages in the Figure 9. The LABEL and DLABEL show a better compromise between the delivery ratio and redundancy of sent messages, compared to the aggressive forwarder FLOODING. The number duplicate messages generated by DLABEL is slightly larger than LABEL. This is

reasonable with the better delivery-ratio and delivery-time of DLABEL.

Among four algorithms, DLABEL is the best choice as its delivery-ratio, delivery-time are only after those of the FLOODING while the duplicate message is much more reasonable. DLABEL also shows the improvement in performance over the LABEL strategy which uses a fixed modular structure proving the efficiency of our dynamic modular structure algorithm.

B. Performance of Dynamic Modular Structure Identification Algorithm

Again, note that modular structure identification is the fundamental step for social-aware routing algorithms [7] [8] [9] [10]. Improving the identification algorithms to allow online updating modules will enhance routing quality while decrease the computational overhead for node to handle. We will show in the experiments (1) the quality of detected modular structure and (2) the running time of our approach in comparison with computing modular structure from scratch.

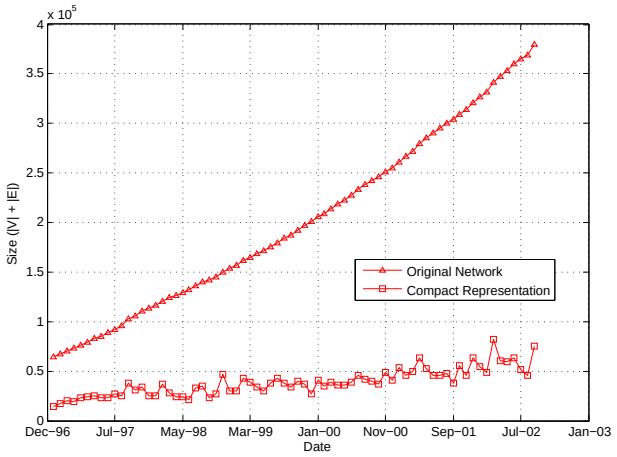


(a) Partition Quality (b) Speed up factor

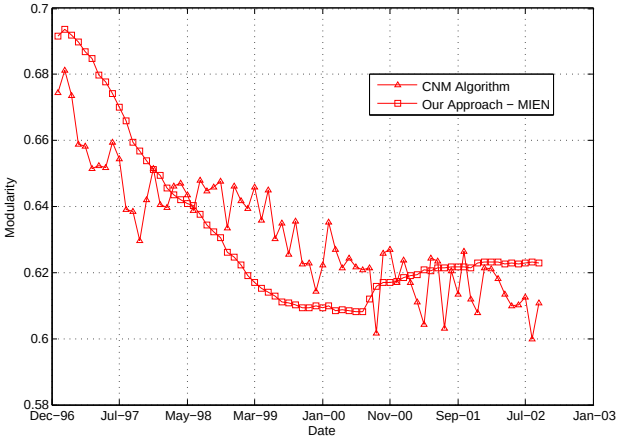
Fig. 10. Comparing partition quality and running time of CNM and our improved algorithm

We test the performance of our approach with CNM algorithm [17] which is one of fastest modular structure identification algorithm. Such a fast algorithm is more suitable for routing purpose as other more exact algorithms usually require extensive computing power. We show that our approach is faster in significant magnitude in the case of identifying modules in dynamic networks.

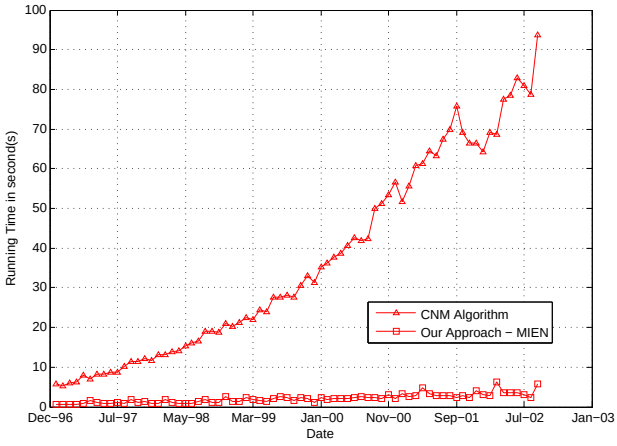
To illustrate the power of our approach, we choose two social networks namely ENRON email network [20] and the citation network of arXiv e-print provided in the KDD Cup 2003 [21]. Both networks exhibit modular structure property due to high values of modularity (Figure 10(a)) notes that networks with modularity higher than 0.3 are considered to have modular structure [15].) Time information are extracted in different ways for each network. In the email network, a link between the sender and the receiver is timed to the point the email sent. Based on timing information, we construct two snapshots of the email network at two time points; the later is one week after the former. Two snapshots have approximately 10^3 links. Two snapshots of the arXiv networks are also constructed in a same way. Each snapshot includes around 3×10^4 nodes and 3×10^5 links.



(a) Size of the Network and its Compact Representation



(b) Modularity at each time point



(c) Running Time

Fig. 11. Modules Identification of the ArXiv Network Through Time. Module structure of the network is identified every month between Jan 1997 and Jan 2003 using CNM and MIEN algorithms. The quantitative measurement modularity and running time is measured at every time point.

For each data set, modules in the first snapshot of the network are identified using CNM algorithm [17]. Then we run our approach on the second snapshot using the information about the modules identified by CNM. To demonstrate the

effectiveness of our approach, we compare it with the results obtained by running CNM directly on the second snapshot.

We observe that the modules identified by our approach in the second snapshots show high similarity in the structure with the results of running CNM directly, meanwhile the computation cost is significantly reduced as the size of compact representation is much smaller than that of the original network.

We show the quality of the detected modular structure and the speed-up achieved by our approach in Figure 10. The difference in modularity is small, which suggests that our approach results are consistent with the results of the selected algorithm CNM, while with much smaller time cost.

We further perform an intensive experiment on the Arxiv network. We identify modules in the network every month in the duration between Jan 1997 and Jan 2003. The number of nodes and links in the network are shown in Figure 11(a). The size of the network increases linearly and the number of links steps up slightly faster. The compact representation shows its advantages with substantial reduction in size. The modularity and running time at each time point are presented in Figures 11(b) and 11(c). The running time of the CNM algorithm increases almost linearly in terms of the size of the network that confirms to the known average $O(n \log^2 n)$ complexity of CNM. Because of running on the small size compact representation, the MIEN algorithm is dramatically faster as shown in Figure 11(c).

The graph for the modularity (Figure 11(b)) is a rough line that shows the significant changes in memberships at each time point. The CNM produces inconsistent modular structure between consecutive time points. Only small changes in the network can result in large changes in assigning nodes to modules. In the other hand, the MIEN algorithm results in a smoother graph showing the stability of module detection algorithm. Moreover, predicting the trend of module structure in the network is also easier with MIEN algorithm than with using CNM algorithm for many separate times.

VI. Conclusion

In this paper, we have proposed a general approach for adaptively updating modules in dynamic networks which is the key in social-aware routing strategies. We show that using modules information of the network at a given time can help 'predicting' effectively the modules of network later. Our algorithm is especially designed for very large and rapidly changing networks. The approach shows enormous improvement in term of running time compared to running one of the fastest module identification method CNM and promises tremendous application as it can be integrated to many different modules identification algorithms. Still the approach needs to be further explored to develop a distributed solution which can be deployed in real MANETs.

References

[1] S. Sesay, Z. Yang, and J. He. A survey on mobile ad hoc wireless network. *Information Technology Journal*, 3(2), 2004.

[2] L. Qin and T. Kunz. Survey on mobile ad hoc network routing protocols and cross-layer design. August 2004.

[3] S. Lee, M. Gerla, and C. Toh. A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks. *IEEE Network*, 1999.

[4] Samir R. Das and Charles E. Perkins. Performance comparison of two on-demand routing protocols for ad hoc networks. pages 3–12, 2000.

[5] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD data set cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle, May 2009.

[6] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[7] Pan Hui and Jon Crowcroft. How small labels create big improvements. In *PERCOMW '07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 65–70, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, New York, NY, USA, 2007. ACM.

[9] Augustin Chaintreau, Pan Hui, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007. Fellow-Crowcroft, Jon.

[10] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 241–250, New York, NY, USA, 2008. ACM.

[11] Santo Fortunato and Claudio Castellano. Community structure in graphs, 2007.

[12] J. Hopcroft, O. Khan, B. Kulis, and B. Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5249–5253, 2004.

[13] G. Palla, A. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, April 2007.

[14] P Pollner, G. Palla, and T. Vicsek. Preferential attachment of communities: the same principle, but a higher level. *Europhysics Letters*, 73:478, 2006.

[15] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(2), 2004.

[16] M. E. J. Newman. Fast algorithm for detecting community structure in networks, September 2003.

[17] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(6):066111, Dec 2004.

[18] A. Arenas, L. Danon, A. Diaz-Guilera, P. M. Gleiser, and R. Guimera. Community analysis in social networks, 2003.

[19] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[20] J. Sun, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Graphscope: Parameter-free mining of large time-evolving graphs. *KDD*, August 2007.

[21] Arxiv citation datasets. KDD Cup 2003, in Conjunction with the Ninth Annual ACM SIGKDD http://www.cs.cornell.edu/projects/kddcup/datasets.html, 2003.