Membership Detection Using Cooperative Mining


Collection Permanent Link: http://hdl.handle.net/10822/761530

© SIAM

This material is made available online with the permission of the author, and in accordance with publisher policies. No further reproduction or distribution of this copy is permitted by electronic transmission or any other means.
Membership Detection Using Cooperative Data Mining Algorithms

Calvin Newport † Lisa Singh † Yiqing Ren †

Abstract
More and more companies are providing data mining and analytics solutions to customers using social media data. The general approach taken by these companies is to continually collect data from social media sites and then use the collected snapshot of the content for a data mining or analytics task. Unfortunately, given the exponential increase in the volume of social media data, building local database snapshots and running computationally expensive algorithms is not always plausible. As an alternative to the centralized approach, in this paper, we study the feasibility of cooperative algorithms where data never leaves the mined social media network, and instead the network users themselves work together, using only the communication primitives provided by the social media site, to solve data mining problems. While cooperative algorithms can be built for many different data mining tasks, to show the viability of this approach, we focus on a task fundamental to many different social mining applications - membership detection (an individual using the social media site wants to efficiently get a request to a member of a known group with unknown membership). Using Twitter as our specific social graph, we seek cooperative algorithms that solve this problem with high probability even when we assume only a small fraction of the Twitter network participates and we enforce a bound on the number of tweets generated. After validating the potential of cooperative solutions on Twitter, we empirically evaluate a collection of cooperative strategies on a snapshot of the Twitter network containing over 50 million users. Our best solution, which we call brokered token passing, can reliably and efficiently detect group membership while requiring only a small number of tweets be sent and a small percentage of users participate.

1 Introduction
The growth of social media content has spawned a new generation of data mining algorithms and services. Many such services require continual collection of data from different social media sites—creating the centralized snapshots needed to run data mining analysis. Though this approach works well for many scenarios, in a future that features an increasing number of consumer-facing data mining services, and increasing dataset sizes, the need to constantly collect snapshots might form a hidden bottleneck. It is important, therefore, to explore alternatives to this centralized strategy.

One alternative suggested in the existing literature is to adopt a distributed data mining (e.g., [25]) strategy which performs some processing at the different source locations, instead of completing all the processing at a centralized location. In this paper, we push this idea to its logical extreme by distributing the data mining responsibilities to the users of the network itself—having these users cooperate to solve data mining problems locally, using only the communication primitives already offered by the social media service. In practice, we assume it is not the human users of these networks directly cooperating, but instead small bits of code running at each user’s machine—e.g., agents—that have access to the user’s account and can perform the actual cooperate needed to solve the problem. To differentiate our fully-distributed approach from previous work, we use the term cooperative data mining algorithms to describe the types of solutions we explore in this paper.

With the right cooperative algorithms, for example, you might be able to identify trending hashtags in Twitter, or detect communities within your Facebook friends, using only local communication and avoiding the need to run centralized algorithms on centralized snapshots of the full network. The promise of a cooperative approach to social network data mining is that it avoids the bottleneck problem of the centralized approach and it might allow for more rapid deployment of certain data mining applications and services.

There are, of course, key questions that must be answered to establish the viability of cooperative data mining, notably: (1) Are useful data mining problems efficiently solvable in a cooperative manner?; (2) How many users must participate to guarantee a high probability of success?; and (3) How can you recruit the needed participant numbers and support them with the needed infrastructure for disseminating and executing agent code?

The majority of this paper tackles the first two questions, though we discuss the third in Section 7. (In other
words, we focus on the effectiveness of cooperative algorithms, not the feasibility/details of their real world deployment). In the following, we adopt a case study approach: identifying a fundamental data mining task in the social media setting and then studying cooperative solutions to the problem, via simulation, in a real social network. More specifically, we study the membership detection problem, which assumes a querier, using a particular social networking site (Twitter in this paper) and a known group with unknown members, wants to detect if there are members of this group nearby in the social network graph and if so send a request to at least one of these members. For example, the querier might have a question about Star Wars that he wants to send to a nearby member of the Star Wars fan group (See Figure 1.)

![Figure 1](Image 1)

**Figure 1:** A scenario on Twitter where two users Ann and Joe (yellow nodes) want a request delivered to nearby member of a given group (red nodes), but do not know in advance where the group members are located in the network. In Ann’s case, she has no group members in her local neighborhood and in Joe’s case, he has two group members in his local neighborhood.

1.1 Centralized vs. Cooperative Solutions to Membership Detection In studying the membership detection problem, we treat Twitter as a graph with nodes corresponding to users and edges capturing the followee/follower relationships. A possible centralized solution would begin by extracting the neighbors of the querier from a network snapshot to see if any of them are members of the group of interest. If not, then the 2-hop network of the querier would be extracted and checked for group members. This process would continue until a group member is detected or the search reaches a pre-specified distance limit. While this solution is straightforward, it requires time to collect the centralized snapshots, and the computational cost will grow with the number of queriers. A cooperative solution, by contrast, would have the Twitter users themselves search their local neighborhood for members using only the communication primitives provided by Twitter (e.g., tweets, mentions)—avoiding the need for centralized snapshots or for a single service to handle all queries over the entire network. In this paper, we seek cooperative algorithms for the membership detection problem that work even if we assume only a small fraction of Twitter users participate in the algorithm, the groups are small, and the total number of messages generated is bounded.

1.2 Cooperative Membership Detection We begin by showing that groups based on hashtags are well distributed across the Twitter social network, with a significant fraction (≈ 25%) of randomly chosen Twitter users ending up within 2-hops of a group member for most groups studied. Drawing on techniques from the distributed algorithm community, we then explore several strategies for solving this cooperative version of our problem, including those based on constant-expansion of the request, token-passing, and a hybrid strategy called brokered token passing. In this strategy, group members recruit a small percentage of their participating followers to act as brokers on their behalf, passing along any member requests they receive using direct messaging. The result is a strategy that maintains the small message complexity of token passing, but also achieves the high success rate of constant expansion (succeeding over 80 - 90% of the time, even when group members recruit only 10% of their followers to act as brokers).

1.3 Contributions This paper offers the following contributions to both our understanding of social networks and the strategies available to detect and identify information they contain: (1) a new approach to data mining social networks that allows the users themselves to cooperate to solve the problem; (2) a scalable algorithm for membership detection in directed social networks; (3) An extensive empirical analysis that uses a Twitter network containing over 50 million nodes for the membership detection problem; (4) An improved understanding of the structure of affinity groups in Twitter.

2 Related Literature

Much research on social data mining has emerged, including work that focuses on characterizing the structure of social networks, e.g. Twitter [17, 23], identifying groups and communities (we refer you to Tang and Huag’s [27] survey for an overview of the area), modeling influence [5], understanding and modeling how information flows and spreads through a network [8, 11–14, 21], modeling the evolution of graphs, communities, and groups [2, 3, 10, 15, 19], and using social networks as tools, e.g. for event detection [6, 20].

Another relevant thread of research attempts to understand different behaviors of users, user influence, and the dynamics of information flow on Twitter. Weng et al. [28] propose a page-rank type measure to quantify influence. Cha et al. [4] compare different measure of influence, while Leavitt and associates [18] conclude...
that media organizations are the most influential since they are better at spreading content. Wu, et. al. [29] characterize and model content and temporal patterns on Twitter. Romero and associates [26] characterize information spread in the context of hashtag topics and map the spread to the idea of “complex contagion”. Kupavskii et al. [16] design models to predict retweet cascades over time.

Other research investigates search strategies for social networks. Adamic and Adar [1] attempt to find short paths in large social networks using local contact information. Dodds, Muhamad, and Watts [7] conduct a social search experiment using 60,000 emails and 18 targets, and find that successful searches take advantage of weak ties that exist in organizational hierarchies.

While all this research is broadly relevant to our work, its focus is on characterizing existing social networks and modeling processes or structures based on this characterization. Also, all of the mentioned research deploys a centralized strategy of collecting a snapshot of the network to be studied, and then analyzing this snapshot. Our focus, by contrast, is on solving data mining problems within the network itself—having the users cooperate to identify the needed information and execute algorithms. This approach represents an extreme realization of the distributed data mining concept of moving processing closer to the data source (see Park et al. [25] for a good summary). Our analysis of our cooperative strategies draws more from the field of discrete distributed algorithm theory (e.g., [22]) than it does from the related field of multi-agent systems (e.g., [9, 24]), which typically model the agents as more continuous processes.

3 Twitter as a Cooperative System
Our approach to the membership detection problem is to recruit Twitter users to participate in a cooperative algorithm designed to efficiently get requests to relevant group members if any such members are nearby in the social graph. To design and describe such algorithms, we propose the following communication model and formal problem definition.

3.1 Twitter as a Communication Network
We model the Twitter social graph as a directed graph $G = (V, E)$, where the nodes in $V$ correspond to Twitter users, and $(u, v) \in E$ iff the user corresponding to $v \in V$ follows the user corresponding to $u \in V$. We use the notation $\text{id}_u$, for a given node $u \in V$, to refer to $u$’s unique id (i.e., the user’s Twitter handle). There are two ways to communicate in this model. First, a user $u$ can broadcast a message to all of its outgoing neighbors in $G$ (indicated by $\text{bcast}$ in our algorithm pseudocode in Section 5). This corresponds to the real world behavior of $u$ sending a tweet that is received by $u$’s followers. In addition, $u$ can send a message directly to a node $v \in V$ (indicated by $\text{send}$ in our pseudocode), if it has previously learned that node’s id. This corresponds to real world behavior of the mention feature of Twitter (i.e., using $@\text{username}$ to direct a tweet to a specific user, regardless of whether or not user $v$ follows $u$).

3.2 The Membership Detection Problem
We now formalize the membership detection problem with respect to our proposed Twitter model. We define a source $s \in V$ as a user in $G$ with a request $r_s$ that can be answered by a group of target members, $T \subseteq V \setminus \{s\}$. The algorithm treats the contents of $r_s$ as a black box, and consider $T$ to be those group members who, on receiving $r_s$, have an answer they are willing to share with $s$. We also define a group of users $P \subseteq V$, where $s \in P$ and $T \subseteq P$, to be the set of participants—the Twitter users willing to follow the rules of our cooperative algorithms. For a given user $u \in V$, we use the notation $P(u)$ to describe the subset of $P$ that are also outgoing neighbors of $u$ in $G$ (i.e., the followers of $u$ that are also participants).

We assume each node $u \in P$ knows $P(u)$, but has no other a priori knowledge of $P$. Similarly, nodes know $\text{id}_v$ for each neighbor $v$ in $G$, but do not have a priori knowledge of the ids of the other nodes in the network. Finally, we assume nodes in $T$ know they are in $T$ (i.e., they know they are in a group willing to respond to requests), but have no a priori knowledge of the identities of the other group members.

More formally, given a graph $G$, a user $s$ and a group $T$, the objective of the membership detection problem is for $s$ to efficiently find a member of $T$ using paths in $G$ even though $s$ does not know the members of $T$. A given instance of the membership detection problem is solved when a message containing $r_s$ arrives at a node in $T$. In this paper, we seek solutions to the problem that are efficient, meaning that the total number of messages sent (i.e., tweets generated) are bounded. Choosing this bound is subjective. However, we generally consider a solution that generates messages on the scale of hundreds or thousands in a graph containing over a million nodes to be efficient, while one that requires tens or hundreds of thousands of messages to be inefficient.

A simple strategy for solving our problem would be for the source to flood $r_s$ throughout the subgraph of $G$ induced by $P$. If any node from $T$ is included in this subgraph, it will eventually receive $r_s$. This strategy,
however, will almost certainly violate our requirement that the solution be efficient: the total number of messages sent (i.e., tweets generated) must be bounded to a reasonable quantity. Another temptation might be to use the send communication abstraction to directly send a request from the source to a group member. Strategies of this type are foiled by our assumptions on a priori knowledge: the source does not know in advance the ids of the group members. As will be detailed in Sections 5 and 6, our best performing solution is one we refer to as brokered token-passing.

4 Affinity Groups in the Twitter-sphere
We study the distribution of a particular group membership, affinity groups, across the Twitter social graph to better understand the feasibility of cooperative strategies. We define an affinity group as one that has a shared interest or affinity and use a Twitter hashtag as a proxy for an affinity group. That is, users who all recently tweeted the same hashtag are members of a particular affinity group. The advantage of this approach is two-fold. First, by definition Twitter users who include a particular hashtag in a message are showing an interest in a specific topic. Second, as we will show, it is not always the case that individuals interested in a particular topic form topological clusters in $G$. Therefore, using community detection to identify affinity groups may be misleading. Our approach allows us to use real group structures that already exist within Twitter and to better understand their distribution throughout $G$.

As we detail below, once we identify an affinity group using this method, we then select sources at random and determine the fraction of such sources that have an member from the group nearby—for our purposes, within 2-hops (given the degree expansion of Twitter, moving beyond 2 hops yields neighborhood sizes that are too large for our purposes). If affinity group members are well-distributed throughout the Twitter social graph, we would expect that on average a source would have at least 1 group member nearby. In this case, distributed membership detection solutions (which propagate search requests locally around the source) are feasible. On the other hand, if affinity group members are tightly clustered, we would expect this average to be much lower than 1—leaving distributed membership detection strategies less feasible. As will be shown, our results support the first scenario.

In more detail, for this analysis, we randomly select 1000 source users, and then build out their 2-hop follower networks using the Twitter API. Crucially, we ignore users that have an unusually large number of followers (i.e., more than 10,000). Such users tend to be celebrities or news services, which might provide artificially positive results, as they shorten the effective diameter between nodes and affinity groups, but are unlikely to participate in a distributed search for such group members. We then identified 100 different affinity groups, each describing Twitter users that all tweeted the same hashtag within a constrained time period. The size of these groups ranged from 96 to 9988 members with a median size of around 2450 (see Figure 2). Notice, in all cases these affinity groups are orders of magnitude smaller than the approximately 500 million users (http://www.statisticbrain.com/twitter-statistics/ (5-7-2013)) that make up the full Twitter network. For a source to identify a member of a small group within such a massive network can seem, at first glance, equivalent to finding a needle in a haystack.

We say that a source detects an affinity group member if there is a group member within its 2-hop network. Similarly, we say a given affinity group member is found if it is reached by at least one source. Figure 3 captures the number of sources that detect at least one affinity group member across the different groups. Over the 100 affinity groups studied, the median percentage of sources that reach a a group member is 24%, with the vast majority of found affinity group members located within the second (as oppose to first) hop of the source that detected it. On average, each source detects 1.6 affinity group members. (See Table 1 for relevant statistics.) In other words, affinity group members are often nearby, but there are rarely lots of members nearby—motivating the need for non-trivial distributed strategies for detecting these members.

We note that this distribution of affinity group members roughly matches what we would expect (given the average size of affinity groups and source follower counts) if affinity group members were randomly distributed throughout the Twitter social graph. Figure 4, however, adds nuance to this observed distribution by showing that the median percentage of a source detecting an affinity group member within a particular affinity group is approximately 11%. This result indicates that instead of an even distribution, most affinity groups have a well-connected core, close to most Twitter users, complemented by a larger and harder to reach periphery. A distributed algorithm that can efficiently reach the periphery will have the potential to access a larger number of affinity group members.

5 Distributed Algorithm for Finding Affinity Group Members
In this section, we describe three different distributed strategies for solving the membership detection problem formally defined in Section 3: constant-expansion,
token passing, and brokered token passing. We will evaluate these strategies in Section 6 for affinity group membership using simulation on a snapshot of the Twitter social graph. For the sake of concision, we only present a detailed algorithm (i.e., pseudocode) for the brokered token passing strategy, as this strategy is shown in Section 6 to decisively outperform the others.

### 5.1 Constant Expansion

The constant expansion membership detection strategy generates a controlled flood. It is parameterized by an expansion factor $c$ (assumed to be constant) and hop count $h$. The strategy begins with the source passing its request to $c$ of its participating followers, chosen at random, along with a hop counter initialized to $h$. (In this strategy, as in the other two strategies described below, when we say a user passes a message to a specific group of its followers, we assume it does so by broadcasting the message, i.e., tweeting it, along with a list of the group of followers for which it is intended. Followers not in that group, or not participating, can ignore the message or at a minimum, not worry about retweeting it.)

On being passed the message, a non-source node decrements the hop counter. If the hop counter is still at least 1, it passes along the request and the hop counter initialized to $h$. (In this strategy, as in the other two strategies described below, when we say a user passes a message to a specific group of its followers, we assume it does so by broadcasting the message, i.e., tweeting it, along with a list of the group of followers for which it is intended. Followers not in that group, or not participating, can ignore the message or at a minimum, not worry about retweeting it.)

The total number of messages generated by this strategy is bounded as $O(c^{h-1})$—a value that clearly grows quickly as these parameters increase beyond modest values. We expect this solution to be effective in detecting affinity group members, but inefficient in its message complexity.

### 5.2 Basic Token Passing

The token passing strategy is parameterized by a token count $t$ and hop count $h$. The source creates $t$ tokens, each containing its request, its id, and the current hop count (initialized to $h$). As with constant expansion search, the source then chooses, for each token, a participating follower at random to which to pass the token. On being passed a token, a user decrements the hop count. If this count is still at least 1, it chooses one of its participating followers at random and passes on the token. If the token arrives at a group member, the member can directly respond to the source’s request using the send abstraction and the source id included in the token.

Notice, unlike for constant expansion search, the number of copies of requests in a token-passing system at any one time remains upper-bounded by $t$. It follows that the total number of messages generated in an execution of this protocol is bounded by $O(t \cdot h)$, which is significantly more efficient than constant expansion search for similar parameters. At the same time, however, we expect the limited number of request copies to reduce the probability that the problem is solved (an affinity group member is detected).

### 5.3 Brokered Token Passing

The brokered token passing strategy is an optimization of basic token passing that uses a small number of extra messages to greatly increase the probability that a token containing an affinity group member request finds its way to a member. In more detail, the optimization has each affinity member recruit its participating followers to act as brokers on its behalf. This can be achieved with a single broadcast message per group member that will be received by all its participating followers. (Unlike the token passing messages, this recruitment is not

<table>
<thead>
<tr>
<th>Twitter-sphere Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nbr of affinity groups</td>
<td>100</td>
</tr>
<tr>
<td>Avg nbr of members (mbrs) per expert group</td>
<td>2,145</td>
</tr>
<tr>
<td>Avg nbr of distinct mbrs found per affinity group</td>
<td>290</td>
</tr>
<tr>
<td>Avg nbr of sources that reach a group mbr</td>
<td>239</td>
</tr>
<tr>
<td>Avg nbr of 1-hop neighbors of $s$</td>
<td>115</td>
</tr>
<tr>
<td>Avg nbr of unique 2-hop neighbors of $s$</td>
<td>27,148</td>
</tr>
<tr>
<td>percentage of 1-hop neighbors participating in path to mbr</td>
<td>0.6%</td>
</tr>
<tr>
<td>percentage of 2-hop neighbors participating in path to mbr</td>
<td>0.003%</td>
</tr>
<tr>
<td>Avg nbr of mbrs in 2-hop neighborhood of $s$</td>
<td>1.6</td>
</tr>
</tbody>
</table>
specific to a given instance of membership detection, but can instead be treated as an initial start-up cost.) The source then initiates the token passing strategy described above. If a token arrives at a broker, the broker passes it along to its affinity group member (using the send abstraction). A detailed description of this distributed algorithm is presented in pseudocode format in three parts: Algorithm 1 (the code for the source), Algorithm 2 (the code for affinity group members), and Algorithm 3 (the code for other participants).

6 Membership Detection Experiments

In this section, we analyze the performance of our proposed distributed solutions to the membership detection problem. We simulate our algorithms on a snapshot of the Twitter network from August 2009 that contains approximately 51 million users (nodes) and 2 billion follower/followee relationships (directed edges). (Refer to [4] for a more detailed discussion of the data set.)

For these detection experiments, we need to define a set of target affinity group members ($T$). We begin with the affinity group set described in Section 4 based on 100 Twitter hashtags. We then identify the subset of these members that exist in our Twitter snapshot, resulting in 9433 common affinity group users. We argue that this is a reasonable size affinity group set for our experiments as it is small compared to the size of the data set and a reasonable size affinity group set for our experiments.

Notice that further increases to $c$ fail to increase the success rate. The final 8% of sources are likely sources that are essentially isolated from the largest component. This same upper bound of $\approx 90\%$ appears consistently across our experiments.
6.2 Basic Token Passing

In an attempt to increase efficiency, we consider basic token passing. Figure 6 shows the detection success rate, \( \rho \), for various combinations of hop counts, \( h \), and token counts, \( t \), each evaluated over 1000 random sources. For clarity, the chart is shown using a log-scale. The efficiency of this approach spans from 50 messages sent when \( h = 50 \) and \( t = 1 \) (\( \eta = 0.02 \)), which is significantly more efficient than constant expansion, to a maximum of 50,000 messages when \( h = 5000 \) and \( t = 10 \) (\( \eta = 2 \times 10^{-5} \)), which is worse than constant expansion and studied for the sake of completeness. Notice, the percentage of members found in all cases is less than 1%. In other words, the efficiency we gained over constant expansion was gained at the expense of detection success.

6.3 Brokered Token Passing

We now consider brokered token passing, an algorithm that attempts to combine the high efficiency of token passing with the high success rate of constant expansion. A key parameter in simulating brokered token passing is the percentage of a member’s followers that will agree to participate as brokers. We begin by evaluating this strategy when all followers of members agree to be brokers. Such a high rate of participation is unlikely in practice, but we begin our study here for the sake of providing an upper bound on possible performance.

Figure 7 shows the detection success rate, \( \rho \), for various combinations of token count, \( t \), and hop count, \( h \), each evaluated for a 1000 random sources. For 50 hops and 1 token, 76% percent of sources succeeded (\( \rho = 0.76 \)). If we increase to 5 tokens, this success rate increases to 93% (\( \rho = 0.93 \)). Even with this high success rate the efficiency remains high, with the strategy never requiring more than 250 messages (\( \eta = 0.004 \)) for token passing, plus only an additional single broadcast message for each member to recruit its followers to become brokers.

As previously mentioned, it is unrealistic to assume that every follower of every member will participate as a broker. With this in mind, we also evaluate brokered token passing under successively smaller participation percentages. (For a given participation percentage \( p \), we selected at random, for each member, \( p \) percent of its followers to act as brokers.) Figure 8 shows the detection success rate for \( h = 50 \), over various combinations of \( t \) and participation percentages, each evaluated for 1000 random sources. The results are positive. Notice, for example, that even if we assume only 10% of members’ followers agree to be brokers, 10 tokens is enough to achieve a success rate of 92% (\( \rho = 0.92 \)). The message efficiency here remains quite competitive, with at most 500 messages generated by token passing in the \( t = 10 \) case (\( \eta = 0.002 \)).

To better understand how different participation percentages translate into total number of available brokers, we summarize, in Table 2, the relationship between these two values. Even though the average number of followers for this data set is 41, the average number of followers for the affinity group member set is approximately 32. Therefore, 10% of each member’s follower network is only 3.2 users - a trivial number of brokers when considering the gain in both detection success rate and detection efficiency.

Finally, we analyze the effect of varying the number of hops, \( h \), for our brokered token passing strategy. Figure 9 shows the effect on detection success rate when varying the number of hops and the number of tokens. For this analysis, we assume 20% of a member’s followers are brokers. We find that the search success
rate is fairly constant for a particular number of tokens across different numbers of hops. For this data set, for example, using 5 tokens for 30 hops, leads to a search success percentage of 90% ($\rho = 0.9$). That is, we can reduce hops (and therefore increase efficiency) without suffering a major sacrifice in success percentage.

6.4 Analysis of Target Affinity Group Size

The above results all assume a single affinity group size of 9433. In order to better understand the relationship between detection success and the number of target affinity group members, we conclude by investigating the relationship between affinity group size and detection success when using brokered token passing. We consider three different affinity group sizes, 9433 (the size used in the experiments presented above), 2450 (the average group size from the affinity group distribution analysis in Section 4), and 500 (to provide a performance lower bound). Figure 10 shows the detection success rate for these three different affinity group sizes and different participation percentages for broker recruitment (x-axis). For this experiment, the number of hops is fixed at 50 ($h = 50$) and the number of tokens is fixed at five ($t = 5$). Each combination of parameters was tested for 1000 random sources.

The figure highlights the significant decline in detection success when the number of affinity group members falls to 500 (at least, for reasonable broker participation percentages). An interesting observation, however, is that even though there is a significant decline in detection success rate, it still remains usefully large with a value of $\approx 25\%$. For an affinity group size of 2450, and a participation percentage of only 10%, the success percentage jumps to over 60%, a value that continues to grow as we increase participation.

6.5 Discussion

Our analysis of distributed algorithms for membership detection resulted in a number of interesting findings. First, constant expansion leads to high search success rates for relatively small expansion factors (e.g., we get to $\rho = 0.92$ for $c = 10$ and $h = 5$). While efficiency of this strategy improves over basic flooding, it is still low. Basic token passing, by contrast, boasts high efficiency, but low search success rates. Our brokered token passing strategy, however, manages to achieve both high success rates and high efficiency, even when we assume only a small percentage of Twitter users participate. For example, if we assume 5 tokens, a limit of 50 hops per token, and that only 10% of each affinity group member’s followers participate as brokers, we still succeed more than 80% of the time ($\rho > 0.8$) while sending no more than 250 messages ($\eta = .004$) per execution. Brokered token passing, in other words, satisfies our goal of identifying a realistic distributed strategy that can reliably deliver requests to group members without generating an unreasonably large number of messages or assuming an unreasonably large percentage of Twitter users participating in the membership detection.

7 Conclusions and Future Directions

We return in this final section to the big picture idea of cooperative data mining. In the introduction, we

<table>
<thead>
<tr>
<th>Percentage of Member</th>
<th>Avg Nbr of Brokers Per Member</th>
</tr>
</thead>
<tbody>
<tr>
<td>10%</td>
<td>3.15</td>
</tr>
<tr>
<td>20%</td>
<td>6.32</td>
</tr>
<tr>
<td>50%</td>
<td>15.8</td>
</tr>
<tr>
<td>100%</td>
<td>31.6</td>
</tr>
</tbody>
</table>
specified three questions relevant to the viability of a cooperative approach. This paper provides insight for the first two questions. We show evidence that the useful membership detection data mining problem can be solved by cooperative algorithms in a real social network (Twitter) with reasonable efficiency (a few hundred tweets) and a high success rate (over 80%) using only a small constant fraction of network participants. Such results emphasize the potential of cooperative approaches, but much research remains to explore more complex data mining problems, in a wider variety of social networks, and under a wider variety of conditions. Detection of group membership in graphs is a fundamental social mining problem. It is also a component of other social mining problems, including information diffusion algorithms and online clustering methods. Consequently, membership detection results can be used to guide cooperative algorithms for those data mining problems as well.

This work leaves un-addressed the practical details of how one would induce users to follow the rules of our distributed strategy. We envision several possible implementation strategies. A simple approach could embed the forwarding instructions in the text of the tweet itself, or have participants download and read the rules separately. A more transparent approach is to augment Twitter users accounts with an application that can monitor their Twitter feed and help them implement the relevant distributed algorithm, perhaps using an agent-based framework.

Finally, we reemphasize the perhaps obvious disclaimer that cooperative data mining in social network is by no means proposed as a replacement for centralized data mining in these settings. The centralized strategy has been working well for many years and will continue to return valuable insight. The introduction of a cooperative alternative simply expands the toolbox available as the demand for social data mining continues to grow.

References


