Detecting the Structure of Social Networks using (α, β)-Communities***

Jing He[2], John Hopcroft[1], Hongyu Liang[2],
Supasorn Suwajanakorn[1], and Liaoruo Wang[1]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853
{jeh17, ss932, lw335}@cornell.edu
[2] Institute for Theoretical Computer Science, Tsinghua University, Beijing, China
{hejing2929, hongyuliang86}@gmail.com

Abstract. An (α, β)-*community* is a subset of vertices C with each vertex in C connected to at least β vertices of C (self-loops counted) and each vertex outside of C connected to at most α vertices of C ($\alpha < \beta$) [11]. In this paper, we present a heuristic (α, β)-Community algorithm, which in practice successfully finds (α, β)-communities of a given size. The structure of (α, β)-communities in several large-scale social graphs is explored, and a surprising core structure is discovered by taking the intersection of a group of massively overlapping (α, β)-communities. For large community size k, the (α, β)-communities are well clustered into a small number of disjoint cores, and there are no isolated (α, β)-communities scattered between these densely-clustered cores. The (α, β)-communities from the same group have significant overlap among them, and those from distinct groups have extremely small pairwise resemblance. The number of cores decreases as k increases, and there are no bridges of intermediate (α, β)-communities connecting one core to another. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Further, similar experiments on random graph models demonstrate that the core structure displayed in various social graphs is due to the underlying social structure of these real-world networks, rather than due to high-degree vertices or a particular degree distribution.

1 Introduction

Much of the early work on finding communities in social networks focused on partitioning the corresponding graph into disjoint communities [3, 6, 10, 12–16]. Algorithms often required dense graphs and conductance was taken as the measure of the goodness of a community [4, 7, 10, 17]. To identify well-defined communities in social networks, one needs to realize that an individual may belong to

* Authors are listed alphabetically.

** This research was partially supported by the U.S. Air Force Office of Scientific Research under Grant FA9550-09-1-0675, the National Natural Science Foundation of China under Grant 60553001, and the National Basic Research Program of China under Grant 2007CB807900 and 2007CB807901.

multiple communities at the same time and is likely to have more connections to individuals outside of his/her community than inside. For example, a person in the theoretical computer science community is likely to have many connections to individuals outside of the theoretical computer science community, who may be his/her family members, or enroll in his/her institution, or attend his/her religious group. One approach to finding such overlapping communities is that of Mishra et al. [11], where the concept of an (α, β)-community was introduced and several algorithms were given for finding an (α, β)-community in a dense graph provided there is an advocate for the community. An advocate for a community is an individual who is connected to a large fraction of the members of that community.

In this paper, we discuss the concept of (α, β)-community, and develop a heuristic (α, β)-Community algorithm that in practice efficiently finds (α, β)-communities of a given size. Further, we thoroughly explore the structure of (α, β)-communities in several large-scale social networks. Surprisingly, in a Twitter friendship graph with 112,957 vertices and 481,591 edges, there are 6,912 distinct (α, β)-communities of size 200 among the 45,361 (α, β)-communities returned by the algorithm. Moreover, these (α, β)-communities are neatly categorized into a small number of massively overlapping clusters. Specifically, the (α, β)-communities from the same cluster have significant overlap ($> 90\%$) among them, while the (α, β)-communities from distinct clusters have extremely small ($< 5\%$) pairwise resemblance. This leads to the notion of a core which is the intersection of a group of massively overlapping (α, β)-communities, where the core also shares a significant overlap ($> 75\%$) with every member (α, β)-community in that group.

The total number and average size of cores in the Twitter graph as functions of the community size k are given in Table 1. Interestingly, as the size k increases, some cores merge into larger ones while others simply disappear. Moreover, cores may fracture when they merge into larger ones, with a fraction of vertices disappearing from larger cores and reappearing later. Among the interesting questions we explore in this paper are why (α, β)-communities correspond to well-defined clusters and why there is no bridge of (α, β)-communities connecting one cluster to another. A bridge is a sequence of intermediate (α, β)-communities where adjacent subsets of the sequence have substantial overlap but the first and last subsets have little overlap. Other intriguing questions include whether different types of social networks incorporate fundamentally different social structures, and what it is about the structure of social networks that leads to the structure of cores as in the Twitter graph and why some networks do not display this structure as in random graph models.

By taking the intersection of a group of massively overlapping (α, β)-communities obtained from repeated experiments, we can eliminate the random factors and extract the underlying structure with multiple runs of the (α, β)-Community algorithm. In social graphs, for large community size k, the (α, β)-communities are well clustered into a small number of disjoint cores, and there are no isolated (α, β)-communities scattered between these densely-clustered cores. The

number of cores decreases as k increases and becomes relatively small for large k. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Moreover, the cores correspond to dense regions of the graph, and there are no bridges of intermediate (α, β)-communities connecting one core to another. In contrast, the cores found in several random graph models usually have significant overlap among them, and the number of cores does not necessarily decrease as k increases. Extensive experiments demonstrate that the core structure displayed in various large-scale social graphs is indeed due to the underlying social structure of the networks, rather than due to high-degree vertices or a particular degree distribution.

The rest of this paper is organized as follows. First, we introduce the definition of an (α, β)-community in Section 2 and show their frequent existence. Then, we prove the NP-hardness of finding an (α, β)-community and present the heuristic (α, β)-COMMUNITY algorithm. In Section 3, we apply the algorithm to various large-scale social graphs and random graphs to explore, analyze, and demonstrate the core structure in social networks. We conclude in Section 4 with comments on the problems considered and future work.

2 Preliminaries

The concept of (α, β)-community was proposed by Mishra et al. [11] as a powerful tool for graph clustering and community discovery. In [11], an (α, β)-community refers to a cluster of vertices with each vertex in the cluster adjacent to at least a β-fraction of the vertices in the cluster and each vertex outside of the cluster adjacent to at most an α-fraction of the vertices in the cluster. In this paper, we adopt a slightly different definition. Given a graph $G = (V, E)$ with self-loops added to all vertices, a subset $C \subseteq V$ is called an (α, β)-*community* when each vertex in C is connected to at least β vertices of C (self-loop counted) and each vertex outside of C is connected to at most α vertices of C ($\alpha < \beta$). Similarly to that of [11], this definition acknowledges the importance of self-loops: although a maximal clique should intuitively be a community, this cannot be guaranteed without self-loops. An (α, β)-community in a graph G is called *proper* if it corresponds to a non-empty proper subgraph of G.

Given a subset $S \subseteq V$, for a vertex $v \notin S$, $\alpha(v)$ is defined as the number of edges between v and vertices of S. Similarly, for a vertex $w \in S$, $\beta(w)$ is defined as the number of edges between w and vertices of S (self-loop counted). Then, $\alpha(S) = \max\{\alpha(v) | v \notin S\}$ and $\beta(S) = \min\{\beta(w) | w \in S\}$.

A maximal clique is guaranteed to be an (α, β)-community since self-loops are counted by the definition. Every graph that is not a clique must contain an (α, β)-community (or, a maximal clique) as a proper subgraph. Starting with any vertex, it is either a proper (α, β)-community (i.e. a maximal clique) or there must be another vertex connected to it (i.e. not a maximal clique). Then, a pair of two vertices connected by an edge is either a proper (α, β)-community (i.e. a maximal clique) or there must be a third vertex connected to both (i.e. not a maximal clique). Continue this argument until a proper (α, β)-community is

found or all vertices are included in a clique, contradicting the assumption that the graph is not a clique. Thus, we have the following theorem:

Theorem 1. *Every graph other than a clique contains a proper (α, β)-community.*

Proof. Given a graph $G = (V, E)$, let $C = V$ and repeatedly remove a vertex from the set C with the lowest β-value. We will show that either G is a clique, or C forms a proper (α, β)-community at some point during the recursion.

First, assume that C is not a clique. Let v_1 be the first vertex removed from C with $\beta(v_1) = \rho$. Once v_1 has been removed, $\alpha(v_1) = \rho - 1$ since its self-loop is no longer counted. Hence, $C = V \setminus \{v_1\}$ and $\alpha(C) = \alpha(v_1) = \rho - 1$. Assume that C is still not an (α, β)-community at this point, i.e. $\alpha(C) = \rho - 1 \geqslant \beta(C)$. For each vertex $v \in \{u \in C | (u, v_1) \notin E\}$, $\beta(v)$ does not change and for each vertex $v \in \{u \in C | (u, v_1) \in E\}$, $\beta(v)$ is reduced by one. Then, v_1 is connected to some vertex $v_2 \in C$, which now has the lowest β-value and will be removed from C in the next iteration. The removal of v_1 must have reduced $\beta(v_2)$ by one such that $\beta(C) = \beta(v_2) = \rho - 1$.

If the set C does not become an (α, β)-community as we recursively remove vertices in this way, then $\beta(C)$ must be reduced by one during each iteration. Further, if a vertex v_i is removed from C in the ith iteration, $\beta(v_i)$ should be equal to $\rho - (i - 1)$ upon its removal, which means v_i has an initial β-value ρ and is connected to all removed vertices $v_1, v_2, \cdots, v_{i-1}$ outside of C. Thus, if no (α, β)-community is ever found until the last vertex v_n, $n = |V|$, has been removed from C, then the graph G is simply a clique. □

Given a graph G with a self-loop added to each vertex and an integer k, define COMMUNITY as the problem of finding an (α, β)-community of size k in G. Given a graph G and an integer k, define CLIQUE as the problem of determining whether there exists a clique of size k in G.

Theorem 2. *The* COMMUNITY *problem is* NP*-hard.*

Proof. We will show that if the COMMUNITY problem is polynomial-time solvable, so is the CLIQUE problem, which is a well-known NP-hard problem.

Let $\{G = (V, E), k\}$ be an input to the CLIQUE problem, where the goal is to decide whether G contains a clique of size k. Without loss of generality, assume that G is not a clique and $k \geqslant 3$. Let $n = |V|$ and for each m such that $k \leqslant m \leqslant n - 1$, construct a graph $H_m = (V_m, E_m)$ as follows:

$$V_m = V_{m,1} \cup V_{m,2},\ V_{m,1} = \{x_i | 1 \leqslant i \leqslant n + m + 1\},\ V_{m,2} = \{y_j | 1 \leqslant j \leqslant m + 1\};$$
$$E_m = \{(x_{i_1}, x_{i_2}) | 1 \leqslant i_1 < i_2 \leqslant n + m + 1\} \cup \{(y_{i_1}, y_{i_2}) | 1 \leqslant i_1 < i_2 \leqslant m + 1\} \cup \{(y_j, x_i) | 1 \leqslant j \leqslant m + 1, 1 \leqslant i \leqslant m - 1\}.$$

H_m contains a clique of size $n + m + 1$ and a clique of size $m + 1$, where each vertex of the second clique is connected to a fixed subset of $m - 1$ vertices of the first clique. Let $G_m = G^* \cup H_m^*$, where G^* and H_m^* are obtained by adding self-loops to all vertices of G and H_m, respectively. Note that G^* and H_m^* are disjoint.

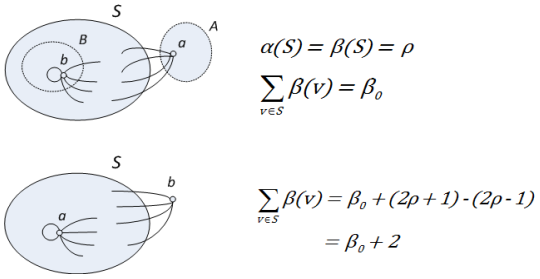


Fig. 1. The (α, β)-COMMUNITY algorithm.

The graph G has a clique of size k if and only if it has a maximal clique of size m, $k \leqslant m \leqslant n-1$. Then, we proceed to prove that G has a maximal clique of size m if and only if G_m contains an (α, β)-community of size $n+2m+1$. First, assume that G contains a maximal clique on $V' \subseteq V$ with $|V'| = m$. Consider the set $S = V' \cup V_{m,1}$ with $\beta(S) = m$. By the maximality of the clique V', every vertex in $V \setminus V'$ is adjacent to at most $m-1$ vertices in V'. Further, by the construction of the graph H_m, every vertex in $V_{m,2}$ is adjacent to $m-1$ vertices in $V_{m,1}$. Hence, S is an (α, β)-community of size $n+2m+1$ since $\alpha(S) = m-1 < \beta(S)$.

Now, assume that G_m has an (α, β)-community S of size $n+2m+1$. Since the set S contains at least $(n+2m+1) - (n+m+1) = m$ vertices from $V_{m,1}$, there exists at least one vertex v of $S \cap V_{m,1}$ that is not connected to any vertex of $V_{m,2}$. In general, assume that S contains k vertices of $V_{m,1}$, $m \leqslant k \leqslant n+m+1$, and thus $\beta(S) \leqslant \beta(v) = k$. If $k < |V_{m,1}|$, there exists at least one vertex outside of S which is adjacent to k vertices in S, leading to $\alpha(S) \geqslant k \geqslant \beta(S)$ that contradicts the definition of (α, β)-community. Hence, S contains all vertices of $V_{m,1}$.

Then, assume that there exists some vertex $y_j \in V_{m,2}$ in the set S, i.e. $|S \cap V_{m,2}| \geqslant 1$. Since $|S| - |V_{m,1}| = m < |V_{m,2}|$, at least one vertex of $V_{m,2}$ is outside of S. Note that $V_{m,2}$ is a clique and every vertex of $V_{m,2}$ is connected to $m-1$ vertices of $V_{m,1}$. Hence, $\beta(S) \leqslant (m-1) + |S \cap V_{m,2}|$ and $\alpha(S) \geqslant (m-1) + |S \cap V_{m,2}|$ that contradict the fact that S is an (α, β)-community. Thus, the remaining m vertices of S are all from V. Recall that $\alpha(S) \geqslant m-1$ and there are no edges between V and $V_{m,1}$. If $S \setminus V_{m,1}$ is not a clique, then $\beta(S) \leqslant m-1 \leqslant \alpha(S)$, again leading to a contradiction. Hence, $\beta(S) = m$ since $S \setminus V_{m,1}$ is a clique, and $S \setminus V_{m,1}$ is also a maximal clique of size m since $\alpha(S) < \beta(S) = m$. Therefore, we have completed the proof by constructing a correspondence between the COMMUNITY problem and the CLIQUE problem. □

Next, we give a heuristic algorithm for finding an (α, β)-community of size k in a graph $G = (V, E)$. Starting with a random subset $S \subseteq V$ of k vertices, the algorithm proceeds as follows. As long as $\alpha(S) > \beta(S)$, replace the vertex in S having the lowest β-value with the vertex outside of S having the highest α-value. Each such swap will increase the value of $\sum_{v \in S} \beta(v)$ by $-(2\beta-1)+2\alpha+1 = 2(\alpha-$

$\beta)+2$ if there is no edge between the two vertices, or $-(2\beta-1)+2\alpha-1 = 2(\alpha-\beta)$ if there is an edge between the two vertices. Since $\sum_{v\in S}\beta(v)$ cannot increase infinitely, the algorithm either returns an (α,β)-community S or reaches a state in which $\alpha(S) = \beta(S)$.

If $\alpha(S) = \beta(S)$, let $A = \{v \in V \setminus S \mid \alpha(v) = \alpha(S)\}$ and $B = \{w \in S \mid \beta(w) = \beta(S)\}$ denote the two subsets of vertices with the highest α-value and the lowest β-value, respectively. The algorithm finds a pair of vertices $a \in A$ and $b \in B$ that are not connected, if such a pair exists, and swaps a and b by adding a to S and removing b from S. Since self-loops are counted, the sum $\sum_{v\in S}\beta(v)$ is increased by two, as illustrated in Fig. 1.

Algorithm 1 (α,β)-COMMUNITY$(G = (V,E), k)$

```
S ← a random subset of V of k vertices
while β(S) ⩽ α(S) do
    S ← SWAPPING(G, S)
    A ← {v ∉ S | α(v) = α(S)}
    B ← {v ∈ S | β(v) = β(S)}
    if {(a_i, b_j) ∉ E | a_i ∈ A, b_j ∈ B} ≠ ∅ then
        pick such a pair of vertices (a_i, b_j)
        S ← (S − {b_j}) ∪ {a_i}
    else if {a_i ∈ A | (a_i, a_k) ∉ E, ∀a_k ∈ A, k ≠ i} ≠ ∅ then
        pick such a vertex a_i
        S ← S ∪ {a_i}
    else if {b_j ∈ B | (b_j, b_k) ∉ E, ∀b_k ∈ B, k ≠ j} ≠ ∅ then
        pick such a vertex b_j
        S ← S − {b_j}
    else
        S ← S ∪ A
    end if
end while
return S
```

Then, the condition $\alpha(S) = \beta(S)$ may cease to hold and the algorithm returns to replacing the vertex in S having the lowest β-value with the vertex outside of S having the highest α-value. Since $\sum_{v\in S}\beta(v)$ cannot increase infinitely, repeatedly performing the above steps will either find an (α,β)-community S or lead to the case where $\alpha(S) = \beta(S)$ and the sets A and B form a bi-clique. In the latter situation, if a vertex $v \in A$ is not connected to any other vertex of A, adding v to S will increase $\beta(S)$ by one but not increase $\alpha(S)$, resulting in an (α,β)-community. Similarly, removing some $w \in B$ that is not connected to any other vertex of B will also produce an (α,β)-community.

Thus, when the algorithm terminates, it either finds an (α,β)-community or gives us a set S for which $\alpha(S) = \beta(S)$ and the sets A and B form a bi-clique, where neither A nor B has an isolated vertex in the subgraphs induced by the respective sets. When this situation is reached, we simply add all vertices of A

to S and start a new round of the algorithm. Although we cannot guarantee to find an (α, β)-community due to this latter case, in practice when k is not too small (e.g. smaller than 20), we never run into the bi-clique situation and thus always find an (α, β)-community.

Algorithm 2 SWAPPING$(G = (V, E), S)$

while $\beta(S) < \alpha(S)$ **do**
  $A \leftarrow \{v \notin S \mid \alpha(v) = \alpha(S)\}$
  $B \leftarrow \{v \in S \mid \beta(v) = \beta(S)\}$
  pick a vertex $a \in A$ and a vertex $b \in B$
  $S \leftarrow (S - \{b\}) \cup \{a\}$
end while
return S

A mathematical description of this (α, β)-COMMUNITY algorithm, along with a subroutine called SWAPPING, is given above. Three corollaries are also given to demonstrate the correctness and proper termination of the SWAPPING algorithm. Their proofs are straightforward and thus omitted from this paper for the sake of conciseness.

Corollary 1. $\sum_{v \in S} \beta(v)$ *is strictly increased during each iteration of the* SWAPPING *algorithm.*

Corollary 2. *The* SWAPPING *algorithm always terminates. When it terminates, swapping any pair of vertices in* A *and* B *will not increase* $\sum_{v \in S} \beta(v)$.

Corollary 3. *The* SWAPPING *algorithm returns a subset* S *with* $\beta(S) \geqslant \alpha(S)$.

3 Experimental Results

3.1 Social Graphs

Twitter The Twitter dataset [1, 2] corresponds to a directed friendship graph among a subset of Twitter user accounts. Each vertex represents an individual Twitter user account, and each edge represents a following relation from one user to another. For simplicity, we convert this directed graph into an undirected graph, ignoring the direction of the edges and combining multiple edges with the same pair of endpoints. Further, we iteratively remove from the graph the isolated and degree-one vertices in order to get rid of the insignificant outliers. This effectively reduces the number of vertices and edges, resulting in a smaller graph with 112,957 vertices and 481,591 edges. Then, the average degree of the Twitter graph is 8.52. Finally, self-loops are added to this graph in accordance with the definition of (α, β)-community.

For a given size k, the heuristic (α, β)-COMMUNITY algorithm is applied to the Twitter graph for finding (α, β)-communities starting with a number of (e.g.

500) random subsets of size k. Theoretically, the algorithm is not guaranteed to terminate within a reasonable period of running time, thus we specify an upper bound (e.g. 1,000) on the number of iterations the algorithm can execute. However, from what we have observed in the experiments, the case of not finding any (α, β)-community within 1,000 iterations is extremely rare. In other words, 500 (α, β)-communities are obtained most of the time with 500 runs of the algorithm.

(a) Core (b) Chain

Fig. 2. The overlapping structure.

To shed a light on how many (α, β)-communities there are in the Twitter graph, 45,361 runs of the algorithm are performed for $k = 200$ and 6,912 distinct (α, β)-communities are obtained. Surprisingly, many (α, β)-communities are observed to massively overlap with each other and differ only by a few vertices. Moreover, such a great number of (α, β)-communities are all clustered into a small number of disjoint groups. Specifically, every pair of (α, β)-communities from the same group shares a resemblance higher than 0.9, while every pair of (α, β)-communities from distinct groups shares a resemblance lower than 0.06. Here, the pairwise resemblance $r(A, B)$ between two sets A and B is defined as:

$$r(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

The overlapping (α, β)-communities form a "core" structure rather than a "chain" structure, as illustrated in Fig. 2. The intersection of all (α, β)-communities in each group bears an over 75% resemblance with every single (α, β)-community in that group. For $k = 200$, all 6,912 (α, β)-communities found by the 45,361 runs of the algorithm cluster into four "cores". The "cores" correspond to dense regions of the graph while being exclusively disjoint, and in contrast to what we would have expected, there are no isolated (α, β)-communities scattered between these densely-clustered "cores".

For a group of pairwise similar (α, β)-communities, we formally define the *core* to be the intersection of those (α, β)-communities. The number of cores can be determined by computing the resemblance matrix of all obtained (α, β)-communities. Intuitively, (α, β)-communities can be categorized according to the resemblance matrix in a way that every pair of (α, β)-communities in the same category is similar to each other, i.e. the pairwise resemblance is large. A pairwise resemblance is considered to be sufficiently large if it is greater than 0.6,

k	25	50	100	150	200	250	300	350	400	450	500
number of cores	221	94	19	9	4	4	4	3	3	3	3
average core size	23	45	73	112	151	216	276	332	364	402	440

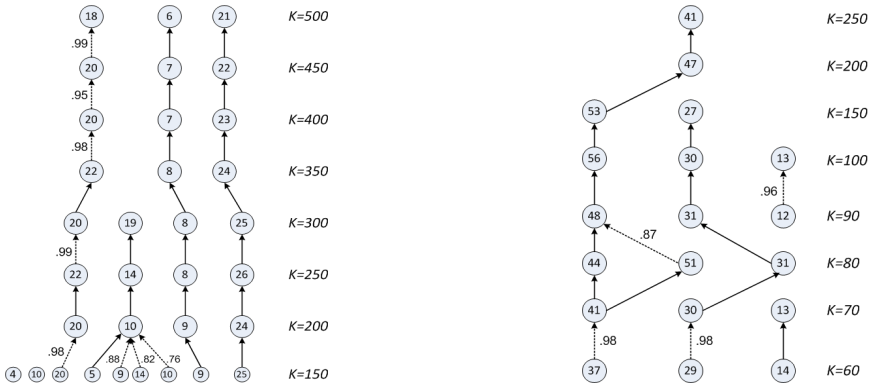
Table 1. Cores of the Twitter graph

while in practice we frequently observe resemblance greater than 0.9. Based on each category, a core is formed by taking the intersection of all member (α, β)-communities. Therefore, the number of cores is equal to that of such intersections, i.e. the number of blocks along the diagonal of the resemblance matrix. The number and average size of cores in the Twitter graph as functions of the community size k are given in Table 1.

The number of cores decreases as the size k increases. This number is relatively small when k becomes large and will eventually decrease to one as k further increases, indicating that (α, β)-communities are well clustered into a small number of cores before gradually merging into one large core. For example, the (α, β)-communities are clustered into 9 cores for $k = 150$ and 4 cores for $k = 200$, where in both cases the cores are disjoint from each other. As the size k increases, the cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. A layered tree diagram is constructed to illustrate this phenomenon in Fig. 3(a).

Each level of the diagram, indexed by the size k, consists of cores extracted from collections of pairwise similar (α, β)-communities by taking their respective intersections. For each pair of cores in adjacent levels, a directed edge is added from the lower level to the upper level if the fraction of overlap is significant, i.e. a substantial fraction (e.g. 60%) of vertices in the core of the lower level is contained in the core of the upper level. If the fraction of overlap is smaller than one, a dotted arrow with this fraction labeled is added to represent a partial merge. Otherwise, a solid arrow with the label "1" omitted is added to represent a full merge. As shown in Fig. 3(a), the fraction of overlap is close to one as we move up the levels, that is, a core of some lower level is (almost) entirely merged into a core of the next higher level.

The definition of (α, β)-community does not prevent a community from having more edges connecting it to the rest of the graph than those connecting within the community itself. Empirically, there are many more vertices outside of an (α, β)-community, and the edges connecting the community to the rest of the graph are almost always more than those connecting within itself. This definition gives an intuitive criterion as to whether to classify a subset of vertices as a community, i.e. the number of edges connecting each vertex in the community to vertices of the community should be strictly greater than that connecting any vertex outside of the community to vertices of the community. Moreover, by taking the intersection of a number of massively overlapping (α, β)-communities, the set of (α, β)-communities which differ only by a few vertices is reduced to an underlying core. Thus, each (α, β)-community consists of one of a small number



(a) Twitter (b) Slashdot

Fig. 3. Tree diagrams indexed by the size k. (Each circle represents a core obtained for a given size, in which the integer denotes the β-value of the core. Each dotted arrow represents a partial merge with the fraction of overlap labeled, and each solid arrow represents a full merge.)

of cores and a few random peripheral vertices, and these peripheral vertices are what gives rise to such a large number of (α, β)-communities.

Before proceeding to our experiments on other social networks, we provide a detailed discussion on the core structure. There might be a generic bias in the (α, β)-COMMUNITY algorithm which is attracted to dense regions of the graph, and thus it is possible that (α, β)-communities located in sparse regions of the graph are never found by the algorithm.

A natural question is what causes the Twitter graph to display this core structure, and further, why the graph shows only a small number of disjoint cores for a large size k. As we will show later, this is due to the fact that a definite social structure, as opposed to randomness, exists in the Twitter network. To take a closer look into this, we simplify the Twitter graph by removing low-degree vertices, i.e. vertices of degree lower than 19, and then obtain a smaller graph with 4,144 vertices and 99,345 edges. The smallest β-value for most (α, β)-communities is given by 19, thus removing vertices of degree lower than 19 will get rid of insignificant low-degree vertices without destroying the fundamental structure of the graph. Again, the (α, β)-COMMUNITY algorithm is applied to this graph with minimum degree 19 for $k = 200, 250, 300, 350, 400$, and exactly two disjoint cores are obtained in each case. Between any two adjacent levels in the corresponding tree diagram, the two cores of the lower level are completely contained in those of the upper level. One possible reason for such a small number of cores could be that the vertices of the cores are more "powerful" in pulling other vertices toward them. If we remove the two cores from the graph and repeat the experiment for $k = 200$, the returned (α, β)-communities are no longer clustered and form a large number of scattered communities.

Another question is why there are exactly two distinct cores in the simplified Twitter graph. For instance, define C_1 and C_2 as the two cores obtained for $k = 200$. C_1 corresponds to a fairly dense subgraph with 156 vertices and 3,029 edges, where the minimum degree is 23 and the average degree is 38.8. C_2 has 159 vertices and 2,577 edges, where the minimum degree is 19 and the average degree is 32.4. Consider the bipartite graph with the two sets of vertices being the vertices of C_1 and the vertices of C_2. Surprisingly, there are only 105 cross edges between C_1 and C_2, where 110 (70%) vertices of C_1 and 100 (63%) vertices of C_2 are not associated with any cross edges. Thus, the cores C_1 and C_2 correspond to two subsets of vertices that are densely connected internally but sparsely connected with each other. As a result, they are returned by the algorithm as the cores of two groups of massively overlapping (α, β)-communities.

It is observed that, in addition to the merging of cores, some cores existing for a smaller k simply disappear from the tree diagram as k increases. In other words, few vertices of these disappearing cores are contained in the cores of the next higher level. The cores take on more vertices as the community size k increases, and there may be two cores taking on the same set of vertices. Thus, the SWAPPING algorithm should run into one of the following two situations: either 1) most vertices of the two cores merge into a new core with some peripheral vertices discarded, or 2) most vertices of one core plus a small fraction of the other form a new core with the latter one disappearing. To verify that one of the above two cases happens, consider the two cores obtained for $k = 150$ that later disappear for $k = 200$, as shown in Fig. 3(a). Let C be one of the two disappearing cores, and recursively perform the following process: enlarge C by adding a random vertex $v \notin C$, run the (α, β)-COMMUNITY algorithm on this enlarged C to find an (α, β)-community of one size larger, and update C to be this obtained (α, β)-community. This process is repeated a number of times until the size of C is increased to 200. Empirically, any obtained (α, β)-community of size 200 contains only a small fraction of vertices of the initial core, while the initial core was completely contained in the (α, β)-communities of size up to about 170. A core may fracture when merging into some other core of a larger size. What happens is that, as vertices are added to one core A, they are also well connected to another core B. As k further increases, these vertices of A will be included in a larger core C that completely contains B, leading to the disappearance of A. If we continue to increase k, the vertices that have disappeared may reappear in a larger core that completely contains C, since they are well connected to the rest of that core.

A *bridge* between two (α, β)-communities or two cores S_1 and S_m is a sequence of intermediate (α, β)-communities $S_2, \cdots, S_{m-1}$, where the pairwise resemblance is large between adjacent subsets but small between the first and last subsets, i.e. $r(S_1, S_m) < 0.3$ and $r(S_i, S_{i+1}) > 0.6$ for all $i \in \{1, 2, \cdots, m-1\}$. The *length* of the bridge is thus given by $m - 1$. Recall that for $k = 200$, (α, β)-communities are all clustered into four disjoint cores, and there is little overlap between any two (α, β)-communities from distinct cores. It is possible that there exists a bridge in the Twitter graph, but the bias of our algorithm may prevent

k	30	40	50	60	70	80	90	100	150	200	250
number of cores	29	10	3	3	3	3	3	3	2	1	1
average core size	25	33	41	53	62	72	85	97	148	197	244

Table 2. Cores of the Slashdot graph

k	30	40	50	60	70	80	90	100	150	200	250
number of cores	64	49	41	45	32	31	25	30	32	20	18
average core size	36	45	52	63	73	81	88	101	146	182	223

Table 3. Cores of the Coauthor graph

it from being found. Thus, although no bridge is detected in this experiment, a subsequent question is whether the graph contains any bridge between two cores at all.

Next, the following experiment is designed to determine whether there exists a bridge between two cores. Pick any two cores obtained for $k = 200$ and recursively perform the following steps: randomly choose r vertices from one core and $200 - r$ vertices from the other to form an initial subset of size 200, and apply the (α, β)-Community algorithm to this subset. If every iteration returns an (α, β)-community that substantially overlaps with one core but is disjoint from the other, then it implies that there does not exist any bridge between the two cores. During 100 runs of the algorithm, 99 of them return such an (α, β)-community that significantly overlaps with one core but is disjoint from the other. Only one trial returns an (α, β)-community C that contains 95.54% of one core A and 26.22% of the other core B. However, no other intermediate (α, β)-communities can be found between B and C using the above method, which demonstrates the non-existence of a bridge.

Another approach to finding a bridge is to search for (α, β)-communities that fall between cores. Generate random subsets of size 200 and run the (α, β)-Community algorithm recursively. As we have seen before, four disjoint cores are obtained with 500 runs of the algorithm, and for another 45,361 runs, the (α, β)-community obtained at the end of each iteration is compared with the four cores to check whether it is an intermediate (α, β)-community. This approach is also useful for estimating the total number of (α, β)-communities of a given size. Among the 45,361 runs, no intermediate (α, β)-communities are found, however, only 6,912 distinct (α, β)-communities are returned, which indicates a relatively small number of (α, β)-communities of size 200 and/or a generic bias of our algorithm towards some particular communities over others.

Overall, the above experiments have suggested that there is no bridge between cores, that is, there is not likely to exist a sequence of intermediate (α, β)-communities that connects two cores with substantial overlap between adjacent pairs. The non-existence of such a bridge demonstrates the underlying social structure of the Twitter network with (α, β)-communities neatly categorized into a few densely-clustered disjoint cores.

Slashdot Slashdot is a technology-related news website known for its professional user community. The website features contemporary technology-oriented news submitted by users and evaluated by editors. In 2002, Slashdot introduced the Slashdot Zoo feature, which allows users to tag each other as friends or foes. The social network based on the common interest shared among Slashdot users was obtained and released by Leskovec et al. [10] in February 2009.

The Slashdot graph contains 82,168 vertices and 504,230 edges, with an average degree of 12.3. Similarly, the (α, β)-Community algorithm is applied to this dataset and the statistics are given in Table 2. The number of cores decreases as the community size k increases and becomes relatively small for large k, behaving the same as it did in the Twitter graph. The cores returned by the algorithm are almost disjoint from each other and correspond to dense regions of the graph, with few edges connecting the bipartite graph induced by the vertices of each pair of cores. This indicates that (α, β)-communities are well clustered into a small number of cores for large k, which correspond to dense regions of the graph and share little overlap among them. For example, the (α, β)-communities are clustered into three nearly disjoint cores for $k = 100$, where only 171 edges connect the two cores of size 93 and 100 that have 2,142 and 1,105 internal edges, respectively. Before eventually merging into one large core as k further increases, these densely-clustered cores emerge as the underlying social structure displayed by the Slashdot network. It is also observed that, as k increases, the cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. A layered tree diagram is constructed to illustrate this phenomenon in the Slashdot graph, as shown in Fig. 3(b).

arXiv hep-ph Coauthor arXiv hep-ph (High Energy Physics – Phenomenology) Coauthor dataset was crawled from the e-print arXiv that covers scientific coauthorship between authors of papers submitted to the hep-ph archive [9]. If author i coauthors a paper with author j, there is an undirected edge between vertex i and vertex j in the corresponding graph. If a paper has k authors, then there is a clique of size k in the graph. The dataset contains papers published from January 1993 to April 2003 (124 months), starting within a few months of the inception of arXiv, and thus it represents essentially the complete history of the hep-ph archive.

The arXiv hep-ph Coauthor graph contains 12,006 vertices and 118,489 edges, with an average degree of 19.7. Since there exists a clique of size 239 in this graph, the (α, β)-Community algorithm returns this clique or a substantial piece of it as a core for $k \geqslant 200$. After removing this clique from the graph, we apply the algorithm again and obtain the statistics as shown in Table 3.

arXiv hep-ph Citation arXiv hep-ph Citation dataset was crawled from the arXiv that covers citations among a collection of 34,546 papers in the hep-ph archive with a total of 421,578 citation links [5, 8]. If paper i cites paper j or vice versa, then there is an undirected edge between vertex i and vertex j in

k	30	40	50	60	70	80	90	100	150	200	250
number of cores	168	123	90	76	64	57	47	43	33	35	28
average core size	28	38	47	55	65	75	84	93	139	182	223

Table 4. Cores of the Citation graph

the corresponding graph. This dataset was originally released in the KDD Cup 2003 [5], and represents essentially the complete history of the hep-ph archive.

The arXiv hep-ph Citation graph contains 34,546 vertices and 420,877 edges, with an average degree of 24.4. Again, the (α, β)-Community algorithm is applied to this graph and the statistics are shown in Table 4. While maintaining a similar tendency, the Citation graph displays more cores than other social graphs for the same value of k. Further, there are four disjoint cores for $k = 900$, and as k continues to increase, the number of cores will eventually decrease to one as in other social graphs.

3.2 Random Graphs

To demonstrate that the structure we have found in social graphs is not merely a random artifact, a similar set of experiments is carried out for random graphs. The comparison between the results from social graphs and random graphs again verifies the existence of community structure in various large-scale social networks.

First, we generate a random graph according to the $G(n, p)$ model, with $n = 112,957$ (same as the number of vertices of the Twitter graph) and $p = 8.52$ (same as the average degree of the Twitter graph). With self-loops added to all the vertices, the graph contains 112,957 vertices and 597,674 edges, which are also similar to those of the Twitter graph. However, conducting the same experiment on this graph reveals a completely different structure from what we have seen in social graphs. The (α, β)-Community algorithm is employed to find 500 (α, β)-communities for each size k from 30 to 300. For each value of k, the 500 obtained (α, β)-communities have little overlap among them (less than 5% in most cases), and are scattered all over the graph where no massively overlapping (α, β)-communities are found. Interestingly, $\alpha = 1$ and $\beta = 2$ hold for each (α, β)-community in this random graph, as opposed to the values as large as 20 in the Twitter graph. Hence, this suggests that random subsets are extracted from $G(n, p)$ which are not even connected, implying the absence of an underlying social structure.

One question is whether the massively overlapping (α, β)-communities in the Twitter graph are due to the high-degree vertices. To resolve this question, we then generate random d-regular graphs for a wide range of values of d with 4,144 vertices (same as the number of vertices of the Twitter graph with low-degree vertices removed). Recall that the lowest β-value for most large (α, β)-communities in the Twitter graph is 19, thus removing vertices of degree lower than 19 will not destroy the fundamental structure of the graph. For each value of

d, the (α, β)-COMMUNITY algorithm returns scattered (α, β)-communities with little overlap among them. This suggests that high-degree vertices are not the primary reason for such few number of cores in the Twitter graph.

Another question that comes up is whether it is the degree distribution of the Twitter graph that causes the massively overlapping (α, β)-communities. To resolve this question, we conduct similar experiments on randomly generated graphs of size 4,144 with power-law degree distribution or the same degree distribution as the Twitter graph. There are several ways to generate random graphs with a given degree distribution, two of which produce the same degree distribution as the Twitter graph while the third one gives a power-law distribution that is different from the Twitter graph.

(1) Uniform model:
Given the degree distribution, we put in edges by selecting vertices uniformly at random. As a result, high-degree vertices are not as densely connected as in the Twitter graph, and this uniform model behaves the same as the $G(n, p)$ model for (α, β)-communities of small size. As the size k increases, (α, β)-communities will gradually overlap with each other, and cores can be extracted from the graph with significant overlap among them.
Moreover, most high-degree vertices are contained in the cores as expected. For example, consider the two cores obtained for $k = 450$. One core is of size 172, containing 93% of the vertices of degree higher than 200 and 63% of the vertices of degree higher than 150. The other core is of size 351, containing 100% of the vertices of degree higher than 200 and 84% of the vertices of degree higher than 150.

(2) Proportional model:
Given the degree distribution, we put in edges by selecting vertices with probability proportional to their degrees. As a result, high-degree vertices are densely connected, and for $k \geqslant 150$, the graph displays only one core with 200 (α, β)-communities returned by the algorithm. Moreover, almost all high-degree vertices are contained in the core. For example, the core is of size 125 for $k = 200$, containing 94% of the vertices of degree higher than 200 and 73% of the vertices of degree higher than 150. The core corresponds to the dense region of the graph due to the way the graph was generated, where high-degree vertices are more likely to be selected.

(3) Preferential attachment growth model:
We first create a clique of size five, then recursively add a new vertex and randomly pick five of the existing vertices to be its neighbors with probability proportional to their degrees. Thus, the resulting graph displays a power-law degree distribution which is different from the Twitter graph. For each size k from 50 to 300, the (α, β)-COMMUNITY algorithm returns a small number of cores with substantial overlap among them. In contrast to what we have observed in the Twitter graph, the number of cores steadily increases with the size k, e.g. 7 cores for $k = 90$ and 11 cores for $k = 250$.

According to the extensive experiments described above, random models do not produce clusters as social graphs do. The cores obtained by the (α, β)-

COMMUNITY algorithm usually have significant overlap among them, and correspond to dense regions due to the way the graph was generated. This demonstrates that the core structure displayed by various large-scale social graphs is indeed due to the existence of underlying structure of the social networks.

4 Conclusion

In social networks, the (α, β)-communities returned by the (α, β)-COMMUNITY algorithm for a given size k are well clustered into a small number of disjoint cores, each of which is the intersection of a group of massively overlapping (α, β)-communities. Two (α, β)-communities from the same group share a significant overlap and differ by only a few vertices, while the pairwise resemblance of two (α, β)-communities from different groups is extremely small. The number of cores decreases as k increases and becomes relatively small for large k. The cores obtained for a smaller k either disappear or merge into the cores obtained for a larger k. Further, the cores correspond to dense regions of the graph, and there are no isolated (α, β)-communities scattered between these densely-clustered cores. In addition, there are no bridges of (α, β)-communities connecting one core to another. Various large-scale social graphs have been explored thoroughly, all of which display the core structure rather than the chain structure.

By constructing random graphs with a power-law degree distribution or the same degree distribution as the social graphs, it is demonstrated that neither high-degree vertices nor a particular degree distribution can result in the core structure displayed in large-scale social graphs. The cores found by the (α, β)-COMMUNITY algorithm in random graphs usually have significant overlap among them and are increasingly scattered across the graph as the size k increases, which implies the non-existence of well-defined clusters in random graphs and verifies the existence of underlying structure in various social networks.

Our work opens several new questions about the structure of large-scale social networks, and it demonstrates the successful use of the (α, β)-COMMUNITY algorithm on real-world networks for identifying their underlying social structure. Further, our work inspires an effective way of finding overlapping communities and discovering the underlying core structure from random perturbations. We conjecture that, in social graphs, the vertices inside an (α, β)-community but outside of the corresponding core are actually located in the overlapping regions of multiple communities. Other open questions include whether different types of social networks display fundamentally different social structures, how the core structure will evolve over time, whether the cores correspond to the stable backbones of the network, and whether the vertices that belong to multiple communities at the same time constitute the unstable regions of the network.

References

1. M. D. Choudhury, Y.-R. Lin, H. Sundaram, K. Candan, L. Xie, and A. Kelliher. How does the sampling strategy impact the discovery of information diffusion in

social media? In *Proc. 4th Int'l AAAI Conf. Weblogs and Social Media (ICWSM)*, 2010.
2. M. D. Choudhury, H. Sundaram, A. John, D. D. Seligmann, and A. Kelliher. Birds of a feather: does attribute homophily impact information diffusion on social media? (under review).
3. A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70(06111), 2004.
4. M. Gaertler. Clustering. *Network Analysis: Methodological Foundations*, 3418:178–215, 2005.
5. J. Gehrke, P. Ginsparg, and J. Kleinberg. Overview of the 2003 KDD cup. *SIGKDD Explorations*, 5(2):149–151, 2003.
6. M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99(12):7821–7826, 2002.
7. K. Lang and S. Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *Proc. 10th Int'l Conf. Integer Programming and Combinatorial Optimization (IPCO)*, 2004.
8. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD)*, 2005.
9. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: densification and shrinking diameters. *ACM Trans. Knowledge Discovery from Data (TKDD)*, 1(1), 2007.
10. J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. In *Proc. 18th Int'l World Wide Web Conf. (WWW)*, 2008.
11. N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Finding strongly-knit clusters in social networks. *Internet Mathematics*, 5(1–2):155–174, 2009.
12. M. E. J. Newman. Detecting community structure in networks. *The European Physical J. B*, 38:321–330, 2004.
13. M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69(066133), 2004.
14. M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(036104), 2006.
15. M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103(23):8577–8582, 2006.
16. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(026113), 2004.
17. S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.