

CS599: Structure and Dynamics of Networked Information (Spring 2005)

02/02/2005: α -communities

Scribes: Yuriy Brun and Dustin Reishus

Last time, we looked at a notion of community where we required high overall edge density among groups of nodes. This made no requirement on individual nodes. In particular, the definition favors the inclusion of high-degree nodes (in WWW terms: popular sites, e.g., yahoo.com, google.com, cnn.com). One may argue that communities should consist of nodes which predominantly belong to that community, i.e., have most of their links within the community.

Definition 1 Let $\alpha \in [0, 1]$ and G be a graph (e.g., the web graph). A set $S \subseteq G$ is called an α -community in G iff $d_S(v) \geq \alpha$ for all nodes $v \in S$, where $d_S(v)$ is the degree of v in S .

If we omit the exact value of α , we assume $\alpha = \frac{1}{2}$ [1]. This definition captures an individual's belonging to the group. If every page in a set of pages S has most of its links to other pages in S , then S is a $\frac{1}{2}$ -community.

The larger α , the more "tightly knit" the community is. So the best communities are the ones with large α . However, this leads to the problem that the whole graph is the "best" community, because all of its edges are within the selected set (itself), making it a 1-community.

The entire graph is not a very interesting community. We are more interested in discovering significantly smaller communities which nevertheless have many edges inside. As an extension, we may also wish to force the inclusion of one or more nodes, and find the best community containing them.

Unfortunately, finding the smallest (or even approximately smallest to within a factor of $c \log(n)$ for some constant c) community is NP-hard, with or without the inclusion of specific sets.

So instead, we will look at heuristics: approaches that may often work in practice, even if they give us no guarantees. Let (S, \bar{S}) be a minimum s - t cut in G . Then, S is almost a $\frac{1}{2}$ -community because each $v \in S \setminus \{s, t\}$ has at least as many edges inside S as crossing (S, \bar{S}) : otherwise moving v to the other side of the cut would make the cut cheaper. If this also held for s and t , then (S, \bar{S}) would be a $\frac{1}{2}$ -community.

If we are looking for communities including a given node s , we can use the above heuristic approach to compute the minimum s - t cut for all t in n min-cut computations. Then, we simply take the best cut found this way.

If we are looking for just communities, without specifying a node s , then we can try all (s, t) pairs ($\Theta(n^2)$ min-cut computations). We can reduce that number of computations to $O(n)$ using Gomory/Hu trees [3]. The idea is that all min-cuts can be "encoded" in a tree.

Theorem 1 Let $G = (V, E)$ be a graph (e.g., the web graph). For all nodes $i, j \in V$, let f_{ij} be the maximum flow (min-cut) between i and j . Let G' be the complete graph on V with edge costs f_{ij} . Let T be a maximum spanning tree of G' .

For each $i, j \in V$, let P_{ij} denote the (unique) i - j path in T . Then, the tree T has the property that $f_{ij} = \min_{e \in P_{ij}} f_e$ for all i, j .

Furthermore, if e is the edge attaining the minimum above, then the two connected components of $T \setminus \{e\}$ define a minimum i - j cut in the original graph G .

Proof. We will prove that $f_{ij} = \min_{e \in P_{ij}} f_e$ by contradiction, ruling out inequality in both directions.

If $f_{ij} > \min_{e \in P_{ij}} f_e$, then inserting (i, j) into T and removing the edge $e \in P_{ij}$ minimizing f_e would create a more expensive tree T' . This contradicts the assumption that T was a maximum spanning tree of G' . So $f_{ij} \leq \min_{e \in P_{ij}} f_e$.

For the other direction, let (S, \bar{S}) be an i - j cut of capacity f_{ij} . Because P_{ij} is an i - j path, it must cross this cut, i.e., there is an edge $e = (u, v) \in P_{ij}$ with $u \in S, v \in \bar{S}$. So (S, \bar{S}) is also a u - v cut, and thus, $f_{uv} \leq f_{ij}$. But then, $\min_{e \in P_{ij}} f_e \leq f_{uv} \leq f_{ij}$, completing the proof. ■

Therefore, the s - t cuts for all $s, t \in V$ can be compactly represented. An interesting additional consequence is that there are only $n - 1$ different min-cuts (and associated min-cut values) for the $n(n - 1)$ different pairs of nodes. Gomory and Hu also show how to compute T from G using only $n - 1$ min-cut computations. Using the approach of first computing T , we can thus find the communities for all nodes with only $O(n)$ min-cut computations.

If we want to find α -communities for $\alpha \neq \frac{1}{2}$, we will need to adapt the approach. The idea is to adapt some degrees so that $d_S(v) \geq \alpha d_G(v)$ holds in the original graph if and only if $d'_S(v) \geq \frac{1}{2} d'_G(v)$ holds in the new graph. We do this by adding a source and sink, and connecting them to each node v in an effort to “balance” the sides of the inequality.

Specifically, we let $d(v) = \sum_{u \neq v} c_{u,v}$. If $\alpha < \frac{1}{2}$, each node v has an edge to the source with capacity $(1 - 2\alpha)d(v)$; otherwise, each node v is connected to the sink with capacity $(2\alpha - 1)d(v)$. Nodes x whose inclusion (or exclusion) is required are connected to s (resp., t) with infinite capacity. (Notice that if no nodes are connected with infinite capacity, then the minimum s - t cut will always just separate s or t from the rest of the graph, corresponding to our above intuition that the entire graph is the best community.) The algorithm then just looks for the minimum s - t cut in the resulting graph G' .

Claim 2 *This approach will produce “almost α -communities,” in the sense that all nodes except those whose inclusion was forced will meet the community constraint.*

Proof. Here, we give the proof for the case that $\alpha \geq \frac{1}{2}$ — the proof for $\alpha < \frac{1}{2}$ is symmetric.

As in the argument for plain min-cuts in a graph, the fact that a node u (without infinite-capacity edges) is on the s -side as opposed to the t -side implies that $\sum_{v \in S, v \neq u} c_{u,v} \geq \sum_{v \in \bar{S}} c_{u,v} + (2\alpha - 1)d(u)$, or $\sum_{v \in S, v \neq u} c_{u,v} + (d(u) - \sum_{v \in \bar{S}} c_{u,v}) \geq 2\alpha d(u)$. Now, because $d(u) = \sum_{v \neq u} c_{u,v}$, this becomes $2 \sum_{v \in S, v \neq u} c_{u,v} \geq 2\alpha d(u)$, or $\frac{d_S(u)}{d(u)} \geq \alpha$. ■

The sets found by this approach can violate the community constraint at the nodes whose inclusion was forced. Notice that this may happen even when there are communities including/excluding specific nodes, i.e., the fact that the algorithm did not find a community does not mean that none exists.

Flake at al. [1] propose another min-cut based heuristic, which makes more of an effort to avoid the “entire graph” problem described above. For a parameter δ that will be varied, all nodes other than a specified node s to be included are connected to a new sink t with capacity δ . We then vary the parameter δ , and look for the minimum s - t cut. The cuts found this way can then be inspected manually, and interesting ones extracted. (Notice that they can be found with one parametric max-flow computation, as discussed in the previous lecture [2].)

For $\delta = 0$, the minimum cut is $(V, \{t\})$. On the other hand, for very large δ , the minimum cut is $(\{s\}, V \cup \{t\} \setminus \{s\})$. If along the way, some δ yields a non-trivial solution, that is an “almost community,” in that it violates the constraint only at the node s .

References

- [1] Gary Flake, Steve Lawrence, C. Lee Giles, Frans Coetzee. “Self-Organization and Identification of Web Communities”. IEEE Computer, 35:3, March 2002.
- [2] G. Gallo, M. Grigoriadis, R. Tarjan, *A fast parametric maximum flow algorithm and applications*. SIAM Journal on Computing 18/1, 2/1989, pp. 30-55.
- [3] R. E. Gomory and T. C. Hu, “Multi-terminal network flows”. J. SIAM, vol. 9, no. 4, pp. 551-570, 1961.