L28-SNA

- Walk
- Random walk
- Lazy walk
- Transition matrix
- In flow, out flow
- Stationary distribution

The idea is to run a random walk started from i and look at the distribution of the walk. We choose a community around i by selecting the vertices with high probability in the random walk distribution. Let us briefly give the intuition behind this idea. Consider a set S of vertices it turns out that

$$\mathbb{P}\left[X_1 \notin S \mid X_0 \sim S\right] = \phi(S)$$

where $X_0 \sim S$ means that we start the walk from a vertex of S chosen proportional by degree. It follows from the above inequality that

$$\mathbb{P}\left[X_1 \in S \mid X_0 \sim S\right] = 1 - \phi(S).$$

In other words, if S has a very small conductance, then the walk will stay in S after one step with a very high probability

members of sets with low conductance (which is what we’re looking for) are more likely to be visited.

Idea: Look at distribution of random walk by time t = 1/phi, where ϕ is the target conductance, and choose the k most probable nodes to get a cluster of size k.

The above algorithm works indeed and it gives sets of small conductance; however, it is not fast enough because we need to compute the full distribution of the walk in order to find out the more probable vertices.

There are ways to go about this, for example rounding down vertices of small probability.

Local graph clustering

Algorithm 1 Local Graph Clustering

```
1:  procedure FindCluster($\tilde{P}, i, \phi$)        ▷ Use lazy walk matrix to find a cluster around $i$
2:      $x_0 \leftarrow i$
3:      $S \leftarrow \{i\}$
4:      for $t \in \{1, ..., \frac{\log n}{\phi}\}$ do        ▷ $\phi$ is the target conductance
5:          Do 1 step of "lazy" walk—that is, given $x_t$ use $\tilde{P}$ to choose $x_{t+1}$
6:          Choose $Z \sim [0, \mathbb{P}[X_1 \in S_t \mid X_0 = x_{t+1}]]$, where by $X_1$ we mean the first step of the walk started
    at $x_{t+1}$.
7:          $S_{t+1} \leftarrow \{j : \mathbb{P}[X_1 \in S_t \mid X_0 = j] \geq z\}$
8:      end for
9:      return $\operatorname{argmin}_t \phi(S_t)$
10: end procedure
```

Flow-Based Post-Processing for Improving Community Detection: We will discuss how algorithms for computing the maximum flow in flow networks can be used to post-process or improve existing partitions of the graph.

Flake et al. [36] proposed to discover web communities by using a focused crawler To first obtain a coarse or approximate community and then set up a max-flow/min-cut problem whose solution can be used to obtain the actual set of pages that belong to the same community.

Lang and Rao [57] discuss a strategy for improving the conductance of any arbitrary bipartition or cut of the graph. Given a cut of the graph (*S, S'*), their algorithm finds the best improvement among all cuts (*S, S'*) such that *S'* is a strict subset of *S*.

To construct a new instance of a max flow problem, such that the solution to this problem (which can be found in polynomial time) can be used to find the set *S* with the lowest conductance among all subsets of *S*. They refer to their method as MQI (Max-Flow Quotient-Cut Improvement).

They refer to their method as MQI (Max-Flow Quotient-Cut Improvement). They
use Metis+MQI to recursively bi-partition the input graph; at each step they bi-partition using Metis first and then improve the partition using MQI and repeat the process on the individual partitions. Anderson and Lang [7] find that MQI can improve the partitions found by local clustering as well.

Community Discovery via Shingling

- Clustering web documents through the use of *shingles* and fingerprints(sketches)
- In short, a length-*s shingle* is *s* of all parts of the object.
- For example, a length-*s* shingle of a graph node contains *s* outgoing links of the node; a length-*s* shingle of a document is a contiguous subsequence of length *s* of the document.

- Meanwhile, a *sketch* is a constant-size subset of all shingles with a specific length, with the remarkable property that the similarity between sets of two objects' sketches approximates the similarity between the objects themselves
- Here the definition of similarity being used is Jaccard similarity, i.e. $sim(A,B) = |A \cap B| / |A \cup B|$. This property makes sketch an object's fingerprint.

- Two first-level shingles are considered as relevant if they share at least one meta-shingle in common, and the interpretation is that these two shingles are associated with some common nodes.
- If a new graph is constructed in such a way that nodes stand for first-level shingles and edges indicate the above-defined relation, then clusters of first-level shingles correspond to connect components in this new graph.
- Communities can be extracted by mapping first-level shingles clusters back to original nodes plus including associated common meta-shingles. This algorithm is inherently applicable
- To both bipartite and directed graph, and can also be extended to the case of undirected graph. It is also very efficient in terms of both memory usage and running time, thus can handle graph of billions of edges.

- Gibson et al. attempt to extract dense communities from large-scale graphs via a recursive application of shingling. In this algorithm, the first-level shingling is performed on each graph node using its outgoing links.
- Each node *v* is associated with a sketch of *c*1 shingles,
- Each of which stands for *s*1 nodes selected from all nodes that *v* points to.
- An inverted index is built, containing each first-level shingle and a list of all nodes that the shingle is associated with. The second-level shingling function is then performed
- On each first-level shingle, producing second-level shingles (also called meta-shingles) and sketches.