Community Detection

Mauro Sozio

Telecom ParisTech

December 5, 2018

Community Detection

Community detection is the problem of finding groups of entities in a network, so as to reflect the structure of the network and other meaningful properties of the underlying system.

Examples of communities are: computer scientists, users in Facebook sharing similar interests, blogs dealing with the same topic, group of dolphins interacting between each other, etc.

Community detection finds application in social science, advertising, and it is a cool problem!

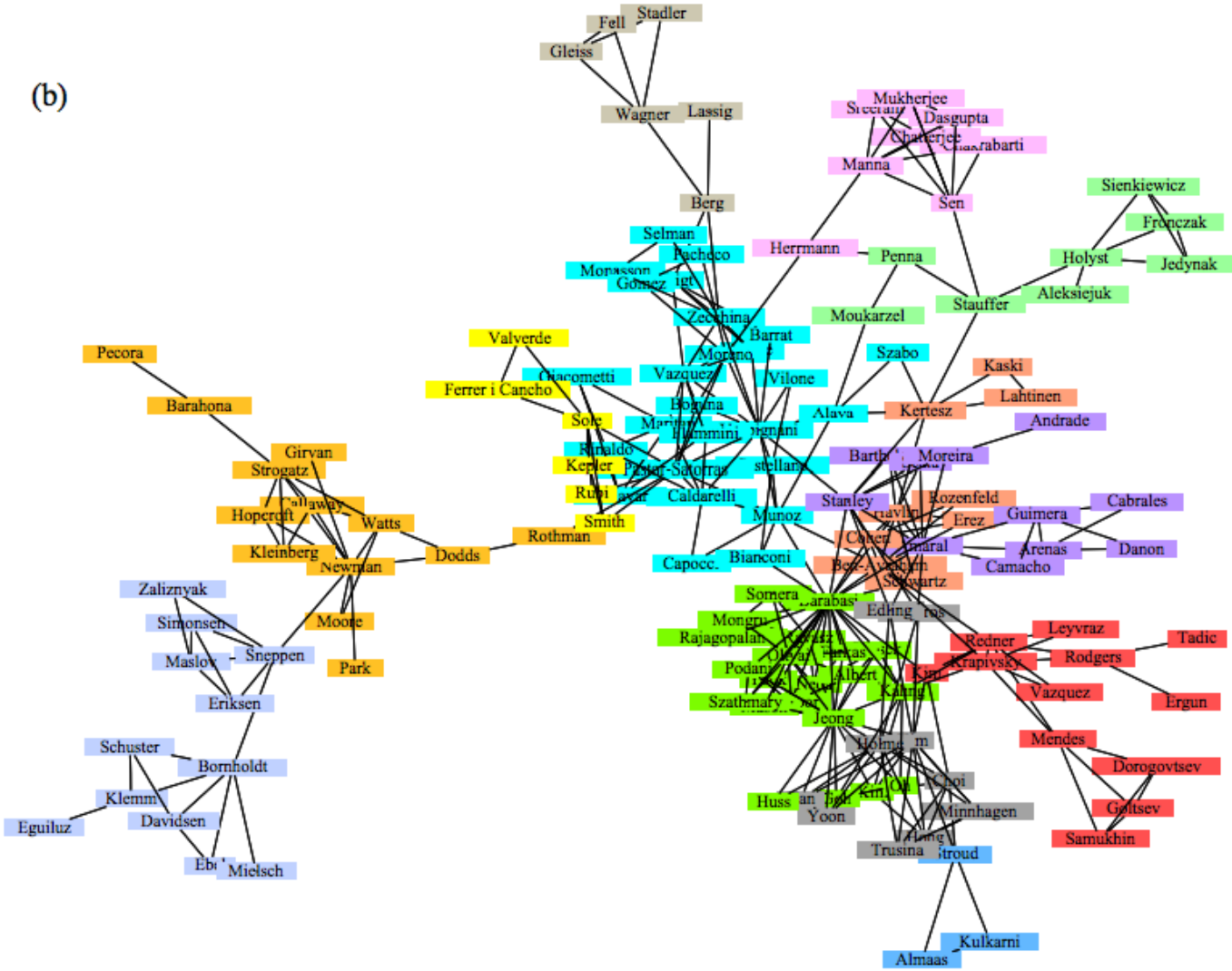

Figure: Communities of physicists and computer scientists.

Figure: Communities of characters in "*Les miserables*" by Victor Hugo.

.

Community Detection as a Graph Problem

The work of [14] sparked in 2002 a wave of interest, although this area has its roots in graph partitioning dating back to the 1970s (METIS [15]).

It can be formalized as a graph problem, where nodes represent users and edges some kind of links between them (e.g. friendship, interaction etc.).

Given a graph $G = (V, E)$, we wish to find subsets of nodes $C_1, \ldots, C_k$ corresponding to communities. The problem is called *non-overlapping* or *overlapping* community detection depending on whether the C_i's are disjoint or not. We might or might not seek to find a partition of V.

Metrics for Community Detection

We can compile the following list of metrics (as done in [2]) so as to devise an effective community detection algorithm:

- separability;
- density;
- cohesiveness;
- clustering coefficient.

Separability

Each community should be well-separated from the rest of the network.

Let $G = (V, E)$ be an undirected graph. Given a set $S \subseteq V$, let $E(S) = \{uv \in E : u \in S, v \in S\}$. The subgraph $H_S = (S, E(S))$ is the *induced subgraph* by S. Let $C(S)$ be $C(S) := \{uv \in E : u \in S, v \notin S\}$.

Definition 1 (Separability)

The separability $g(S)$ of S is defined as: $g(S) := \frac{|E(S)|}{|C(S)|}$.

Density

Each community should contain a large # of links between its members.

Definition 2 (Density)

Let $G = (V, E)$ be an undirected graph. Given a set $S \subseteq V$, we define the *average degree density* of S as $\rho(S) := \frac{|E(S)|}{|S|}$. The *clique density* of S is defined as $\alpha(S) := \frac{|E(S)|}{\binom{|S|}{2}}$.

Cohesiveness

Given a graph $G = (V, E)$, let $C_G(S) := \{uv \in E : u \in S, v \notin S\}$, $A_G(S) := \{uv \in E : u \in S \vee v \in S\}$.

Definition 3 (Conductance and Cohesiveness)

Given a set $S \subseteq V$, we define the *conductance* of S in G as

$$\phi_G(S) := \frac{|C_G(S)|}{\min(|A_G(S)|, |A_G(V \setminus S)|)}.$$

The *cohesiveness* of S in $H = (S, E(S))$ is $h(S) := \min_{S' \subseteq S} \phi_H(S')$.

High cohesiveness for a community S means that S is internally well and evenly connected: To split the community we should remove many edges.

Clustering Coefficient

A *triplet* is a (connected) graph with exactly three nodes and two edges. A *triangle* is a clique on three nodes. A triangle contains three triplets. Let $\tau(G), t(G)$ be the # of triangles and triplets in G, respectively.

Definition 4 (Clustering Coefficient)

The *global clustering coefficient* of a graph $G = (V, E)$ is defined as:

$$cc(G) = \frac{3 \cdot \tau(G)}{t(G)}.$$

$cc(G)$ denotes the probability that a triplet in G is contained in a triangle in G. It is well-known that two friends of a given person are likely to be friends. Therefore a community should have high clustering coefficient.

Modularity (defined in [14])

Definition 5

Given an undirected graph $G = (V, E)$ and a partition $\mathcal{C} = C_1, C_2, \ldots, C_k$ (communities) of the nodes in G the modularity of $\mathcal{C}$ is defined as follows:

$$Q_{\mathcal{C}} = \frac{1}{2m} \sum_{u,v \in V} \left(A_{uv} - \frac{\delta_u \cdot \delta_v}{2m} \right) \sigma(u, v),$$

where $A_{uv} = 1$ iff $uv \in E$, $m = |E|$, $\sigma(u, v) = 1$ if there is i s.t. u, v belong to a same community C_i and 0 otherwise.

Modularity (defined in [14])

Definition 5

Given an undirected graph $G = (V, E)$ and a partition $\mathcal{C} = C_1, C_2, \ldots, C_k$ (communities) of the nodes in G the modularity of $\mathcal{C}$ is defined as follows:

$$Q_{\mathcal{C}} = \frac{1}{2m} \sum_{u,v \in V} \left(A_{uv} - \frac{\delta_u \cdot \delta_v}{2m} \right) \sigma(u, v),$$

where $A_{uv} = 1$ iff $uv \in E$, $m = |E|$, $\sigma(u, v) = 1$ if there is i s.t. u, v belong to a same community C_i and 0 otherwise.

That is the fraction of edges within a same community minus the expected number of edges within a same community in a random graph having the same degree distribution of G (*null model*).

Random Graphs with the Same Degree Distribution

Given a graph $G = (V, E)$ a random multigraph H maintaining for each node its degree in G is produced as follows: each edge is divided into two halves (*stubs*) each of them being assigned uniformly at random to any stub in the graph.

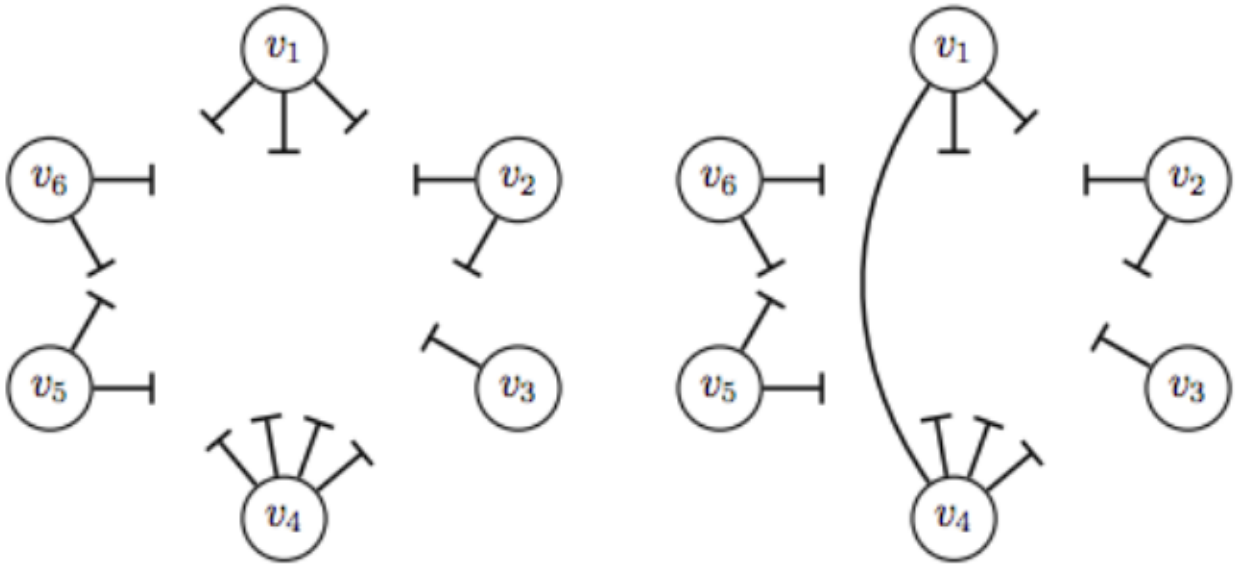

Figure: One stub of v_1 is assigned to v_4, forming an edge.

Random Graphs with the Same Degree Distribution

Each stub is assigned uniformly at random to any of the $2m$ stubs available. In H, we might have self loops, multiple edges per pair of nodes, multiple self loops but the same node degrees of G.

X_{uv} = r.v. indicating the # of edges between u and v in H. We have:

$$E[X_{uv}] = \frac{\delta_u \delta_v}{2m},$$

as the probability of assigning a stub of δ_u to a stub of δ_v is $\frac{\delta_v}{2m}$.

Facts about Modularity

- positive if the # of edges within communities $>$ the expected ones.
- It takes values in $[-1, 1)$

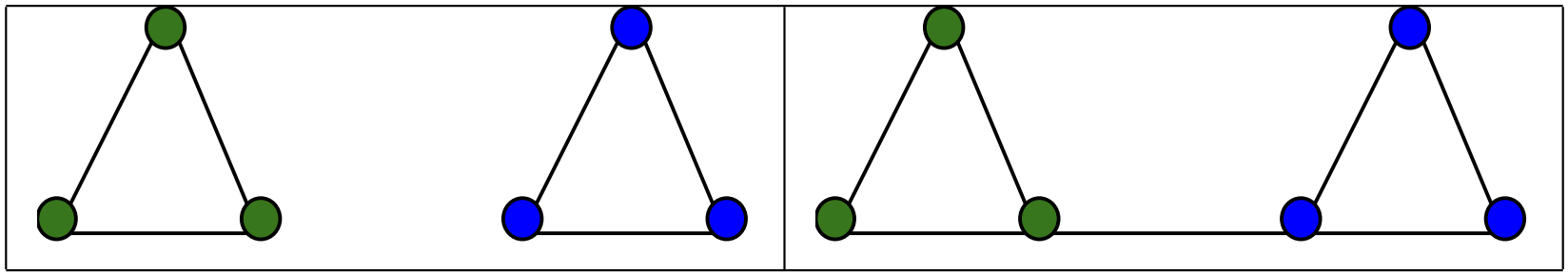

Figure: Same-color nodes belong to the same community. Modularity left graph: $\frac{1}{12} \cdot 12(1-2 \cdot 2/12) = \frac{2}{3} =$ 0.67, right graph: $\frac{1}{14}(4(1-4/12)+8(1-6/12)+2(1-9/12)) =$ 0.52.

Modularity: Issues

Merging "well-separated cliques" (that should be identified as two different communities) might result in larger modularity [1].
Resolution limit: "small" communities might be missed.

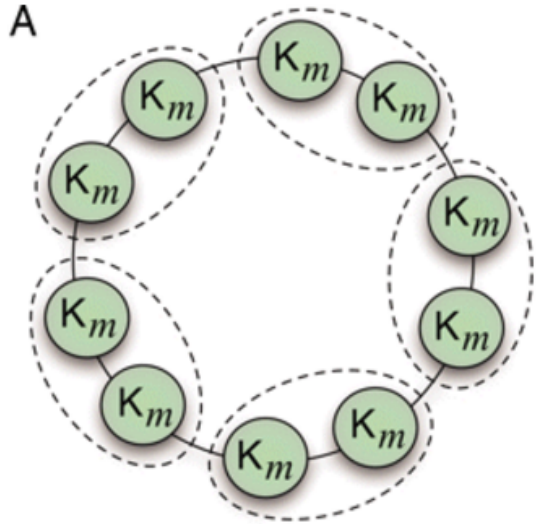

Figure: Cliques connected by one single edges. Placing two cliques K_m in a same community might result in larger modularity, when the number of cliques is large enough.

Optimizing Modularity

Optimizing most of the metrics we considered is NP-Hard. The NP-hardness of modularity optimization has been proved in [16].

Some heuristics have been developed to maximize modularity.

One of the first heuristic is the Louvain Algorithm [8] ...

The Louvain Algorithm (one of the variants)

1: **Input**: A graph G=(V,E)
2: **Output**: A partition of V into communities $\mathcal{C} = C_1, \ldots, C_k$, $k \leq n$.
3: For each $v \in V$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{v\}$.
4: **while** *(true)* **do**
5: **for** $u \in V$ **do**
6: **for** $v \in \delta(u)$ **do**
7: Let C_u, C_v be the community of u, v, respectively.
8: Let $\mathcal{C}_v$ be s.t. C_u is replaced by $C_u \setminus \{u\}$ and C_v by $C_v \cup \{u\}$.
9: Let Q_v be the modularity of $\mathcal{C}_v$.
10: Let v be such that Q_v is max among all neighbors v of u.
11: $\mathcal{C} \leftarrow \mathcal{C}_v$
12: If no improvement on modularity **return** $\mathcal{C}$.

Louvain: Running Time

Stop as soon as the "improvement " in the modularity (in two successive iterations) is $< \epsilon$, ϵ specified in input. Remember: modularity $\in [-1, 1)$. Compute modularity *incrementally*.

The worst-case running time becomes: $O(\frac{n \cdot \delta^2_{\max}}{\epsilon})$, as: the inner *for* loop requires $O(\delta^2_{\max})$, the outer *for* loop requires $O(n)$, the *while* loop $O(\frac{1}{\epsilon})$. It can actually be implemented in $O(\frac{m}{\epsilon})$.

One could speed up the algorithm by processing a sample of nodes/neighbors or approximating the modularity.

Facts about Louvain

Positive facts:

- Successfully applied in a variety of application domains.
- It produces good results, despite the limitations of modularity.
- Simple, efficient in practice, code in C available.
- Generalization to weighted graphs and hierarchical communities.

Limitations:

- Nodes are partitioned, however, often communities do overlap.
- it is recommended to always give algo. pseudocodes, prove that the algorithm terminates, study the asymptotic running time.

The SCD Algorithm

Based on a community metric called *WCC*. Given $x \in V$, $C \subseteq V$:

$$WCC(x, C) := \begin{cases} \frac{\tau(x,C)}{\tau(x,V)} \cdot \frac{vt(x,V)}{|C \setminus \{x\}| + vt(x, V \setminus C)} & \text{if } \tau(x, V) \neq 0 \\ 0 & \text{if } \tau(x, V) = 0 \end{cases}$$

where $\tau(x, C)$ is the # of triangles in C containing x, and $vt(x, S)$, $S \subseteq V$ is the # of triangles containing x, one node in S and one node in V.

Communities C with a large number of triangles inside C and a small number of triangles partially contained in C have large WCC score. "Large" communities are also penalized.

The SCD Algorithm

The WCC of a partition $\mathcal{C} = C_1, C_2, \ldots, C_k$ is the average over the nodes:

$$WCC(\mathcal{C}) = \frac{1}{n} \sum_{i=1}^{k} \sum_{x \in C_i} WCC(x, C_i)$$

The SCD algorithm consists of two main steps:

1. It starts with a partition $\mathcal{C}$ of the nodes (computed with a heuristic);
2. it refines such a partition with local improvements aiming at maximizing the WCC score (similarly to Louvain).

Community-Detection Algorithms

They can be classified as follows:

- Heuristics maximizing an objective function: OSLOM [7], Louvain [8], SCD [4].
- Algorithms based on random walks / PageRank: Walktrap [9], Infomap [10], [11]. According to [6], they perform best, however, they do not scale.
- Other techniques: DEMON (label propagation) [12], LC (borrows some ideas from clustering).

Experimental Evaluation

How to conduct a thorough evaluation:

- the CD algorithm should be evaluated on real-world networks with known ground-truth communities. Precision, recall, F1-score should be reported (see next slide).
- The algorithm should scale to graphs containing billions of edges. Running time should be reported.
- Ground-truth communities should be manually inspected, so as to make sure that they are meaningful.
- The objective is for you to learn which algorithm performs best so that you can use it when needed.

Precision, Recall, F1-score

Let C be the *ground truth* (e.g. a real-world community). The precision and recall of a set A with respect to C are defined as follows:

$$\mathit{precision}(A, C) = \frac{|A \cap C|}{|A|}, \ \mathit{recall}(A, C) = \frac{|A \cap C|}{|C|}$$

F1-score is the harmonic mean of precision and recall defined as follows:

$$F1(A, C) = \frac{2 \cdot r \cdot p}{r + p},$$

where r, p denotes precision and recall of A w.r.t. C, respectively. It holds that $\min(r, p) \leq F1(A, C) \leq 2 \cdot \min(r, p)$, therefore F1 penalizes results with low precision or low recall.

Evaluation of Community Detection Algorithms

Given the *ground truth* set of communities $\mathcal{C}$ and a set of communities $\mathcal{A}$ the F1-score of $\mathcal{A}$ and $\mathcal{C}$ is defined as follows. Given a set A and a collection of sets $\mathcal{C}$ let:

$$b(A, \mathcal{C}) = \max_{C \in \mathcal{C}} F1(A, C),$$

we define $F1(\mathcal{A}, \mathcal{C})$ as follows:

$$F1(\mathcal{A}, \mathcal{C}) := \frac{1}{2|\mathcal{A}|} \sum_{A \in \mathcal{A}} b(A, \mathcal{C}) + \frac{1}{2|\mathcal{C}|} \sum_{C \in \mathcal{C}} b(C, \mathcal{A}).$$

Used also when $\mathcal{A}, \mathcal{C}$ are not partitions of V.

Ground-truth Communities

A collection of networks with ground-truth communities are at [3].

They include excerpts from Youtube, DBLP, Amazon, ...

Ground-truth communities for DBLP:

- each node represents an author of a scientific paper, with an edge between two nodes indicating co-authorship.
- Authors publishing in a same conference or journal belong to the same community.
- Authors belonging to two connected components belong to two different communities.

Evaluation of Community Detection Algorithms: F1 score

From [4]:

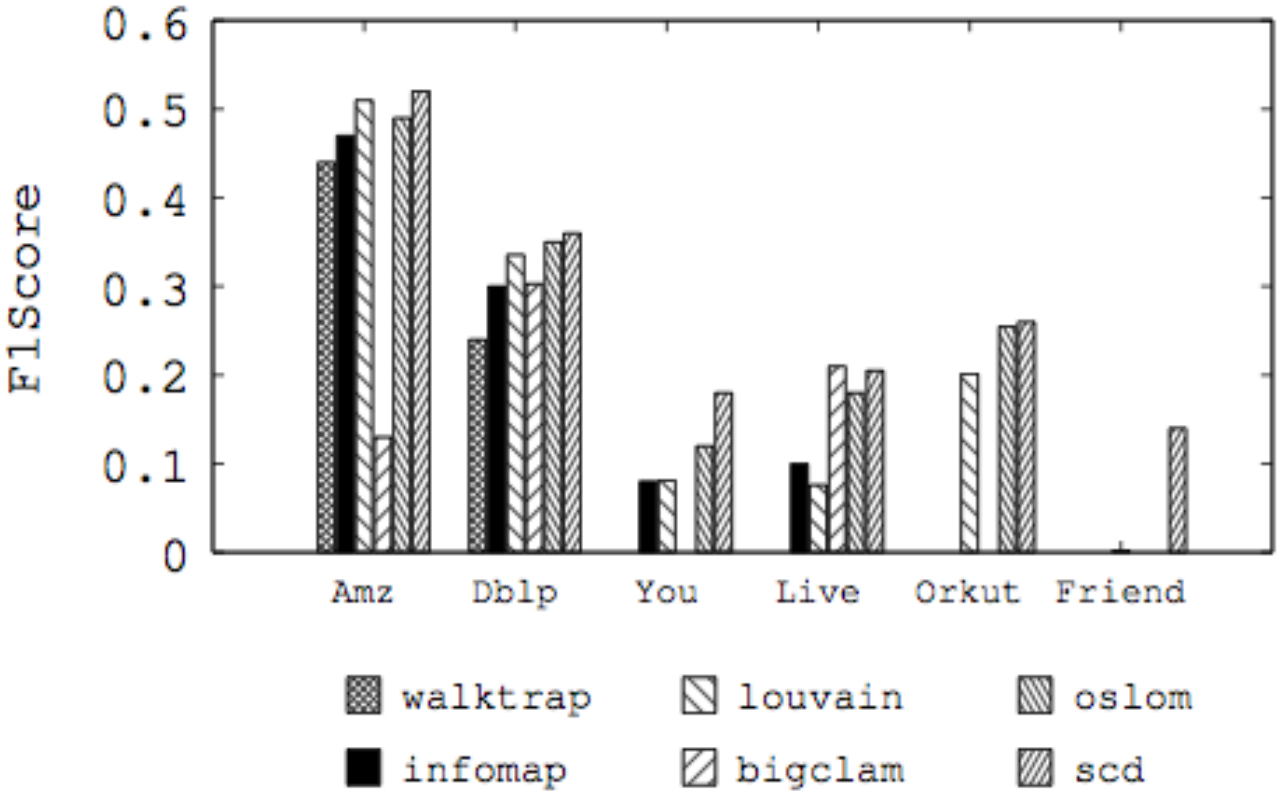

Figure: F1-score of common CD algorithms on the SNAP dataset.

Evaluation of CD Algorithms: Running Time

From [4]:

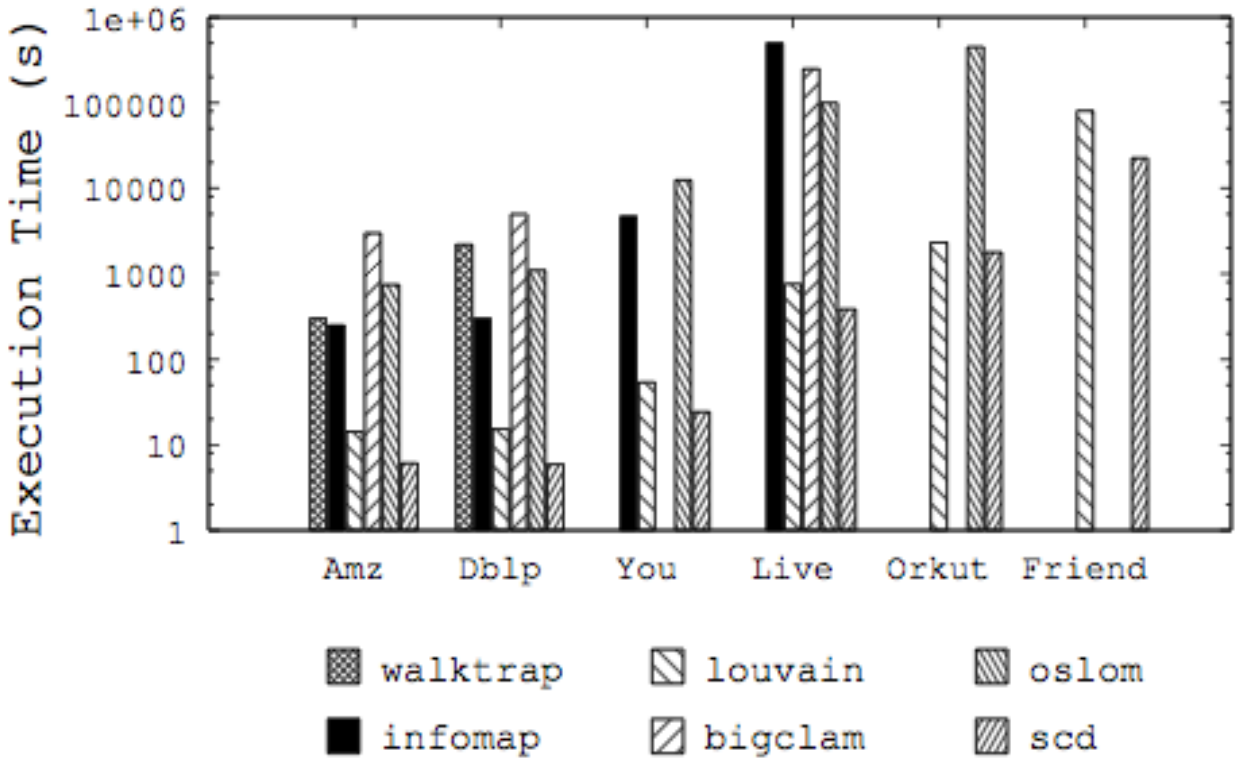

Figure: Running times of CD algorithms on the SNAP dataset (log-log scale).

Conclusions

Despite almost two decades of research, there is no consensus on which CD algorithm performs best.

This is partially due to the difficulty of obtaining large real-world communities with objective ground-truth communities.

Most algorithms ignore edge weights, while there only a few algorithms for finding overlapping communities.

Different algorithms lead often to completely different results [6].

References I

[1] Fortunato, Santo and Barthelemy, Marc
Resolution limit in community detection
Proceedings of the National Academy of Sciences, 36 – 41, (2007).

[2] Yang, Jaewon, and Jure Leskovec.
Defining and evaluating network communities based on ground-truth.
Knowledge and Information Systems, 42.1 (2015): 181-213.

[3] Jure Leskovec and Andrej Krevl.
SNAP Datasets: Stanford Large Network Dataset Collection
`http://snap.stanford.edu/data`, 2014.

[4] Prat-Prez, Arnau, David Dominguez-Sal, and Josep-Lluis Larriba-Pey.
High quality, scalable and parallel community detection for large real graphs.
Proceedings of the 23rd international conference on World wide web. ACM, 2014.

References II

[5] Li, Y., He, K., Bindel, D., Hopcroft, J. E.

Uncovering the small community structure in large networks: A local spectral approach.

Proceedings of the 24th international conference on world wide web. ACM, 2015..

[6] Abrahao, B., Soundarajan, S., Hopcroft, J., Kleinberg, R.

A separability framework for analyzing community structure.

ACM Transactions on Knowledge Discovery from Data (TKDD), 8(1), 5..

[7] A. Lancichinetti, F. Radicchi, J. J. Ramasco, S. Fortunato.

Finding statistically significant communities in networks.

PloS one, 6(4):e18961, 2011.

[8] Blondel, V. D., Guillaume, J. L., Lambiotte, R., Lefebvre

Fast unfolding of communities in large networks.

Journal of statistical mechanics: theory and experiment, 2008

References III

[9] P. Pons and M. Latapy.

Computing communities in large networks using random walks.

Computer and Information Sciences-ISCIS 2005, pages 284293. Springer, 2005.

[10] M. Rosvall and C. T. Bergstrom.

Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems.

PloS one, 6(4):e18209, 2011.

[11] V. Zlati c, A. Gabrielli, and G. Caldarelli.

Topologically biased random walk and community finding in networks.

Physical Review E, 82(6):066109, 2010.

[12] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi.

Demon: a local-first discovery method for overlapping communities.

In KDD, pages 615623. ACM, 2012.

References IV

[13] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann.

Link communities reveal multiscale complexity in networks.

Nature, 466(7307):761764, 2010.

[14] Newman, Mark EJ and Girvan, Michelle

Finding and evaluating community structure in networks

Physical review E, 2004.

[15] B. W. Kernighan and S. Lin. 1970.

An efficient heuristic procedure for partitioning graphs.

Bell System Technical Journal 49, 1, 291307.

[16] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., Wagner, D.

On modularity clustering.

IEEE TKDE, 20(2), 172-188, 2008.

References V

[17] Anagnostopoulos, Aris and Lacki, Jakub and Lattanzi, Silvio and Leonardi, Stefano and Mahdian, Mohammad

Community Detection on Evolving Graphs

NIPS, 2016