Approximation of the Maximal $\alpha-$Consensus Local Community detection problem in Complex Networks

Patricia Conde Céspedes, Blaise Ngonmang and Emmanuel Viennet

Université Paris 13
L2TI

COMPLEX NETWORKS (LIP6) -January 22nd, 2016

This work is supported by REQUEST and Open Food System projects.

Table of contents

Problem formulation

The problem, we want to tackle, can be summarized as follows:

Objective:

"Given a node n_0 of a graph $G(V, E)$ and a parameter α $(0 < \alpha < 1)$, the purpose is to find the biggest $\alpha-$consensus community $C(n_0)$ containing n_0".

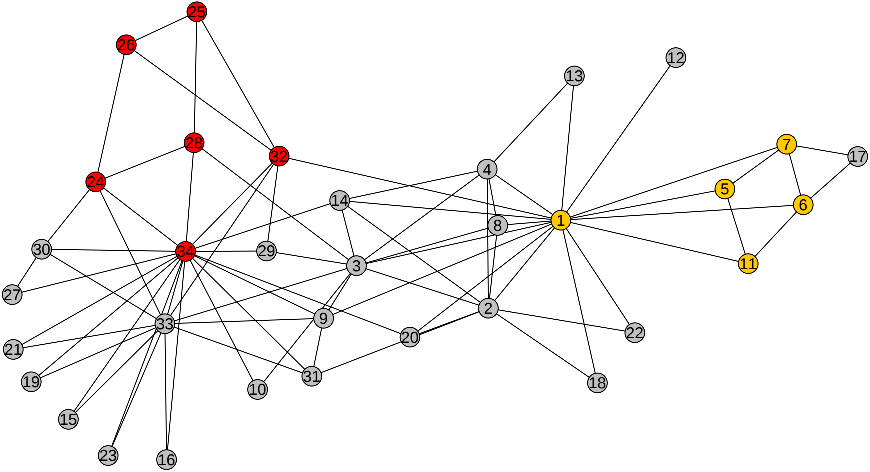
That is, the biggest group of nodes $C(n_0)$ where each node is connected to more than a proportion α of the other nodes. We call this rule, the Condorcet's majority rule. Mathematically, this problem can be formalized as follows:

$$\begin{array}{ll} \underset{C}{\text{maximize}} & |C| \\ \text{subject to} & n_0 \in C \\ \text{and} & |\Gamma(n_i) \cap C| > \alpha(|C| - 1), \forall n_i \in C. \end{array} \tag{1}$$

where $\Gamma(u)$ is the neighborhood of a node u

Examples of $\alpha-$cliques

$\alpha-$cliques for nodes 5 and 28 of Zachary karate club networks for $\alpha = 0.5$.

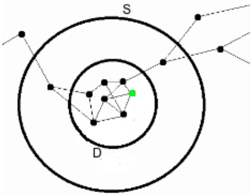


Some applications of $\alpha-$cliques

- In SNA to find groups of people who all know each other.
- In social recommendation.
- Link prediction.
- In electrical engineering to analyze communications networks.
- In bioinformatics, many problems have been modeled using cliques.

Definitions and notations

D: set of nodes identified as members of $C(n_0)$
S: set of neighbors of nodes in D (possible candidates to enter D).



The local community starts with one node $D = \{n_0\}$, at each iteration one or more nodes from S are added to D.
Each algorithm has its own criterion to choose the best candidate (s) from D to enter S.

Table of contents

SOLUTION 1, the *RANK-GAIN* algorithm

At each iteration one node from S is chosen to be added to D. The new node is chosen according to the **gain** g_n caused by its addition to D, given by:

$$g_n = \ell_n - \alpha * |D|$$

The *RANK-GAIN* algorithm chooses the node whose gain is maximum and positive to enter D. If there are ties the new node is chosen according to the number of its neighbors in S. If there are still ties the new node is chosen randomly.

Advantages of the *RANK-GAIN* algorithm

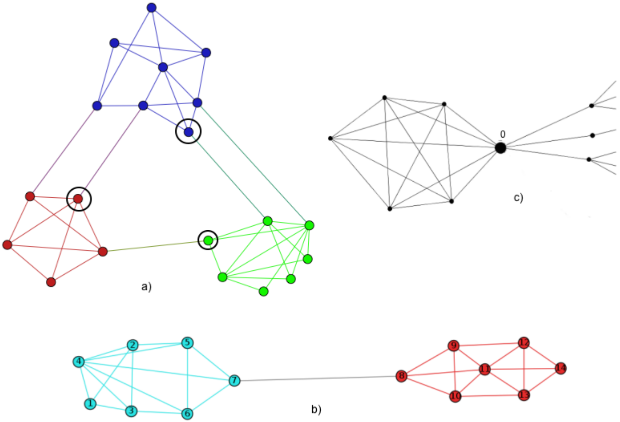
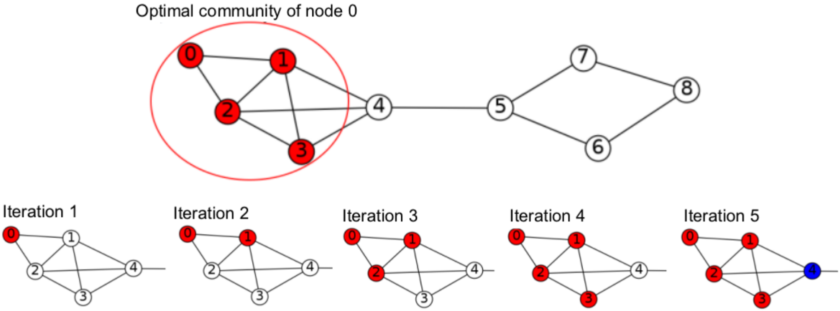


Figure: Advantages of choosing the new node according to the number of its neighbors in S

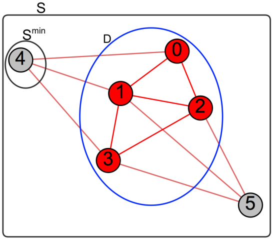
Introduction to SOLUTION 2: the internal degree



When looking for the local community of node 0, the algorithm will give $\mathcal{C}(n_0) = \{0, 1, 2, 3, 4\}$ whereas 0 does not respect the rule anymore.

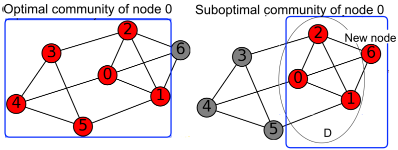
SOLUTION 2, the *RANK-GAIN+* algorithm

- At any iteration, the maximum possible size of D, D_{max}, given the minimal internal degree of nodes in D, d_{min} will be: $D_{max} = \lceil (\frac{d_{min}}{\alpha}) \rceil$
- If D_{max} has been reached the eventual new node is chosen from the set of neighbors of nodes with d_{min}, denoted S^{min}.

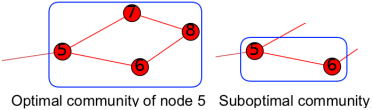


Shortcomings not solved by the *RANK-GAIN+* algorithm

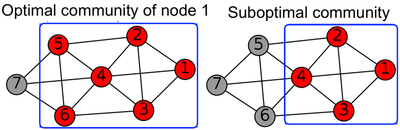
Shortcoming 1:



Shortcoming 2:



Shortcoming 3:



Introduction to SOLUTION 3

The gain is not the most important thing when choosing the new node to enter D. Reminder:

Objective:

Given a node n_0 in the graph, the purpose is the find the biggest community containing n_0 where each node is connected to more than α% of the other nodes.

So, the most important thing is the size of the community. We are not looking for the densest community but for the biggest community where all nodes must respect the majority rule.

Why not to let enter more than one node at each iteration? Why not to prioritize the number of neighbors in S rather than the gain?

Solution 3: The *RANK-NUM-NEIGHS (RNN)* algorithm

To face this problem we propose an algorithm, called *RANK-NUM-NEIGHS*, that prioritizes the entry of nodes with the **highest number of connections in** S over the gain and allows to enter one node or more at each iteration.

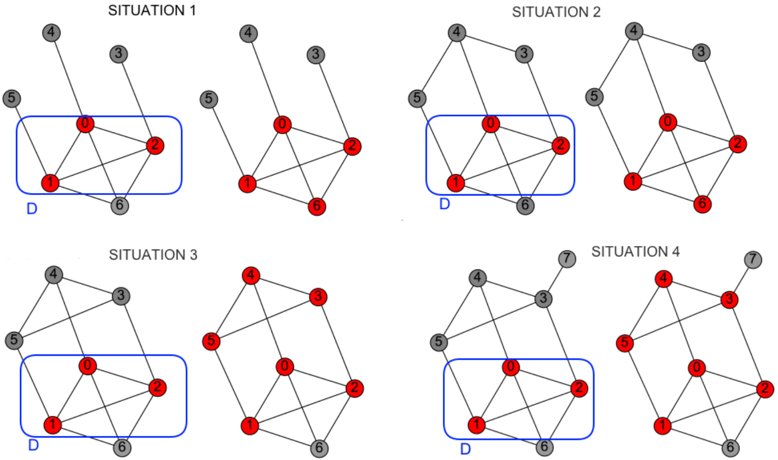
1. It prioritizes the number of common neighbors over the gain when choosing a new node (or more nodes) to enter the local community.
2. At each iteration more than one node might enter the local community.

If a node n has a negative gain, the number of neighbors n needs to enter D is:

$$x = \left\lfloor \left(\frac{\alpha * |D| - \ell_n}{(1-\alpha)} \right) \right\rfloor + 1$$

Solution 3, The *RANK-NUM-NEIGHS*: example of the 4 situations

For each situation the optimal solution is the set of red nodes on the right.



Comparison of the RANK-GAIN+ and the RANK-NUM-NEIGHS algorithms

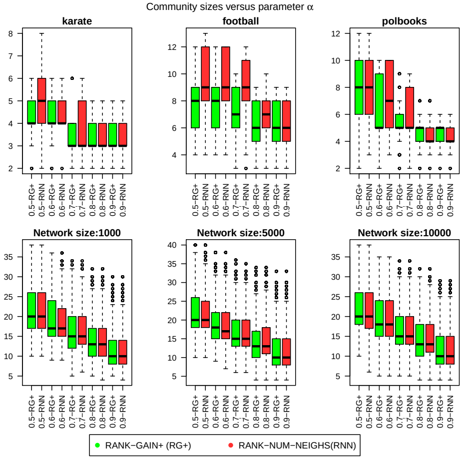
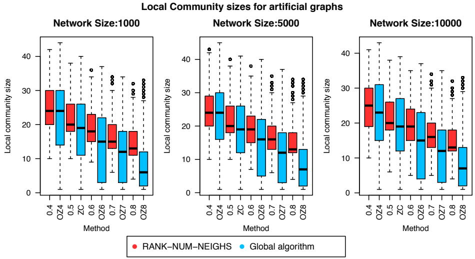
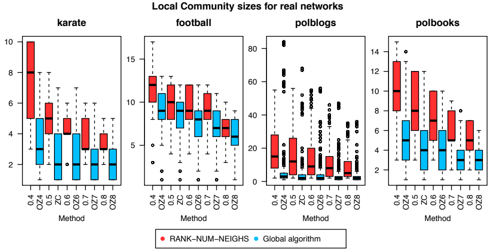


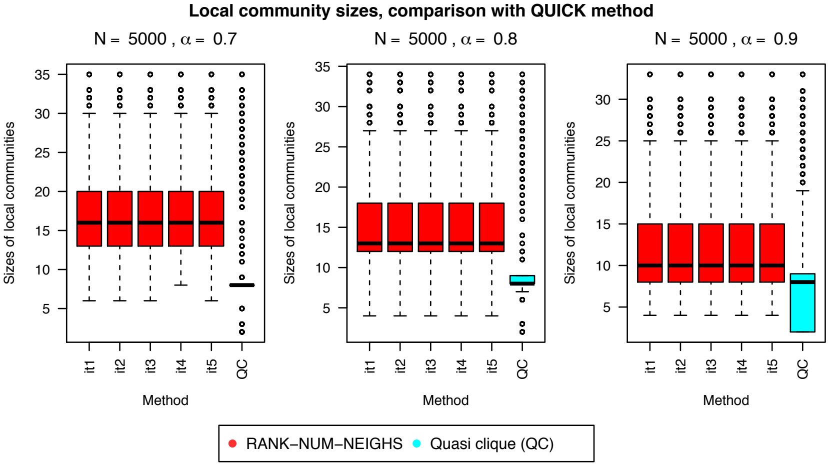
Table of contents

Sizes of the detected communities vs α, comparison to the global *Louvain* method

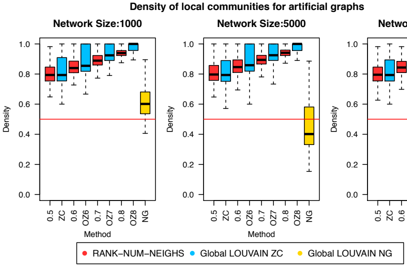


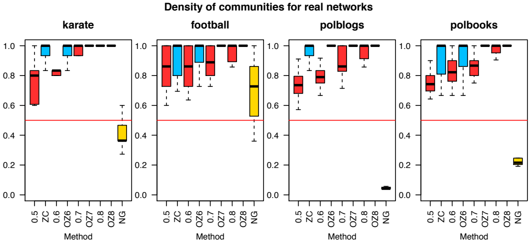


Community-Sizes vs α, comparison to the *QUICK* method



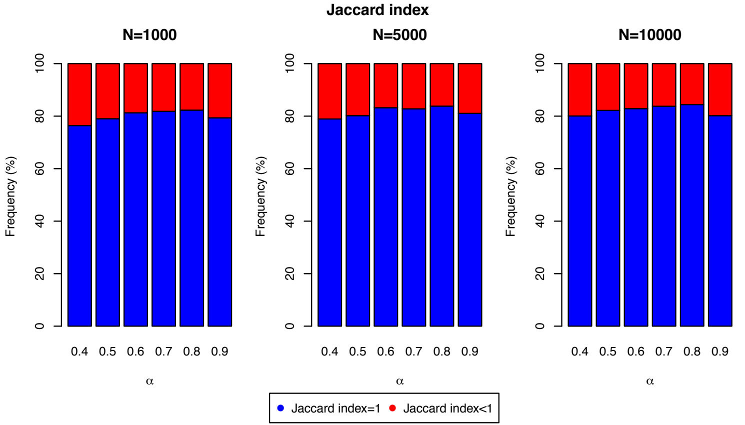
Densities of the detected communities versus α





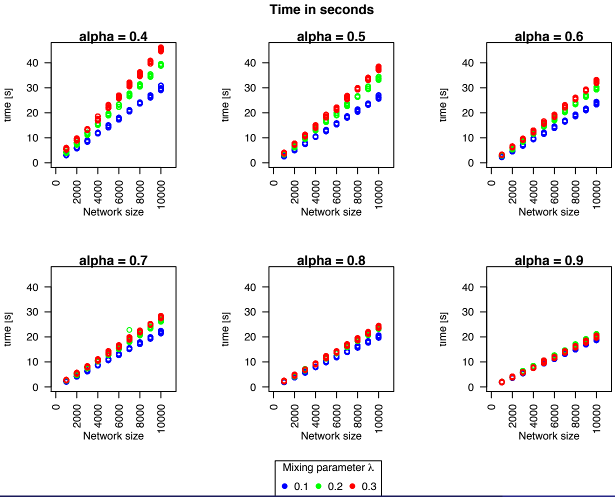
Stability of the algorithm

$$\text{Jaccard index, } J(n) = \frac{|\bigcap_{i=1}^{10} C(n)_i|}{|\bigcup_{i=1}^{10} C(n)_i|}$$



Execution time

The computer used for the experimentations has a Quad Core processor running at 3.33 GHz and 15GB RAM. The algorithm is written in python.

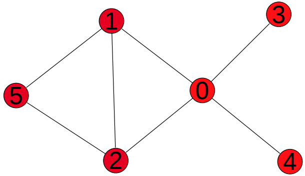


Bound on the optimal solution

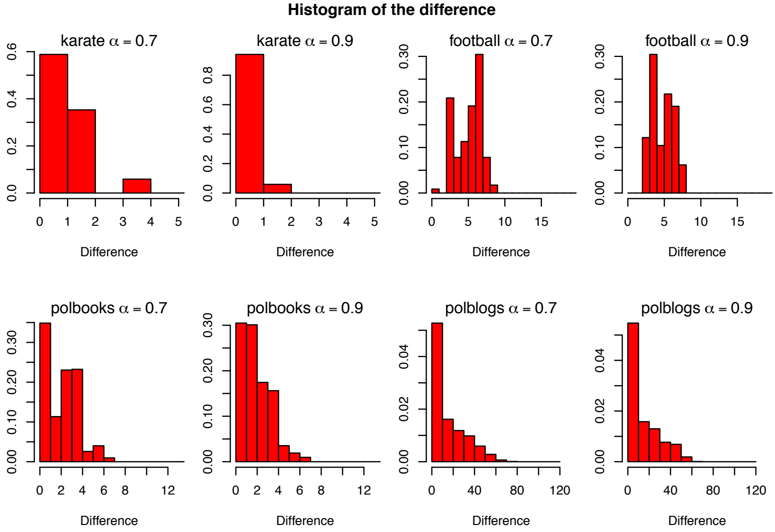
$C^*(n_0)$ is the maximal $\alpha-$consensus community of node n_0, so the optimal solution, we have the following bounds:

- Bound $B_0(n_0) = \left\lceil \left(\frac{d(n_0)}{\alpha}\right) \right\rceil$.
- Bound $B_1(n_0) = \min\left(\max_{n\in\Gamma(n_0)} B_0(n), B_0(n_0)\right)$.
- Bound B_2 found iteratively.
- $|C(n_0)^*| \leq B_2(n_0) \leq B_1(n_0) \leq B_0(n_0)$.
- Example: for node 1 of Zachary Karate Club $B_0(1) = 32$, $B_1(1) = 20$ and $B_2(1) = 10$. The solution found by *RNN* is 8.

Exemple for the calculation of bound B_2



Bound on the optimal solution



Histogram of the difference between the upper bound B_2 and the results of RNN

Table of contents

Conclusions and perspectives

- We presented a **novel method**, called *RANK-NUM-NEIGHS*, for the problem of mining the maximal local $\alpha-$ consensus community of a given node of a network. Starting from a baseline method, our method is introduced to address some shortcomings not solved by the previous ones.
- Our method contains **two important characteristics**. First, it priorizes the nodes who have the biggest number of neighbors in the neighborhood of the community, over the gain. Second, it allows to let enter more than one node at each iteration.
- The *RANK-NUM-NEIGHS* gives good results in real and generated networks in terms of size, stability and execution time. It also performs better in terms of quality than the existing method *QUICK* and than global community detection LOUVAIN method.