Summary of the Project

Community Detection via Maximizing Modularity Function

Author: Mohammadamin Vahedinia

Github Repository: experiments repository link

---

1. Introduction

Community detection, or finding the structure of groups in a network, is a popular problem in analyzing any kind of networks, such as social networks, and chemical reactions networks. As there exists many partitions for a given network, one must evaluate the quality of those partitions and select the best among of them. There are many metrics (or quality functions) proposed for evaluating partitions such as CPM [1], maximum likelihood [2], and Modularity [3]. It is proven that last two are equivalents [4]. In this report, we use modularity function, which is defined as followed:

$$Q = \sum_{i=1}^{c} \left(e_{ii} - a_i^2\right) \tag{1}$$

Where c is the number of communities, e_{ij} corresponds to the fractions of the edges that lie between communities i and j, and a_i is fraction of the edges that start or end in ith community. As mentioned in the literature, maximizing this metric may lead to some unwanted consequences like the resolution limit problem [1, 5] and the degeneracy problem [6], that is, not only maximizing modularity tends to favour merging small communities, but also many partitions exist that are almost optimal. This problems can be solved by various methods, like adding a resolution parameter to the Equation (1) or solving a multi-objective optimization problem instead [5].
However, maximizing modularity is extensively used in practice. Therefore, in the following section we are going to benchmark five algorithms mentioned below based on this method using LFR generated graphs [7]:

1. Clauset-Newman-Moore - $O(md \log n) \sim O(n \log^2 n)$ [8]
2. Louvain - $\sim O(n)$ [9]
3. Leiden - $\sim O(n)$ [10]
4. Simulated Annealing - $\sim O(n^{3.2})$[11, 12]
5. Leading Eigenvectors - $\sim O(n^2)$ [13]

Clearly, there are more methods proposed in the literature, including other Evolutionary algorithms [14–16] which were generating promising results.

2. Methods

In order to measure aforementioned algorithms' performance, first we need to generate test cases, which is described in detail in the `experiments.ipynb` notebook. Overall, multiple networks were created using the LFR network generation implementation in `networkx` python package, taking advantage of the multiprocessing methods to optimize case generation performance. At this step, as the LFR algorithm was not robust [17] and ended up not generating desired network, the parameters were tuned.
Furthermore, the implementation of the algorithms from `networkx` (CNM), `igraph` (Louvain, Simulated Annealing and Leading Eigenvectors) and `leidenalg` (Leiden) packages has been used for this experiments. Additionally, the `normalized_mutual_info_score` function from `sklearn` package was used to measure how two partitioning (expected and predicted) were similar, in a scale from 0 to 1.
It is needed to be mentioned that for using *leiden* algorithm, the modularity function without *resolution parameter* has been used. Also, default parameters were set for the *Simulated Annealing* algorithm.
Finally, due to the limited computing resources, all experiments are limited by a timeout, i.e., processes are halted after a certain amount of time in case of not finding the solution.

3. Conclusive Discussion

In summary, Tables 1, 2 illustrate output of the experiments for different test cases, and Figures 1, 2 visualize NMI and modularity metrics for corresponding tables.

Table 1: Algorithms' Performance for different cases where $N = 1000$ and $<k> = 10$

Algorithm	$\mu = 0.2$			$\mu = 0.35$			$\mu = 0.5$		
	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$
Expected	1	0.191		5	0.315		5	0.115	
CNM	1	0.203	21.578	5	0.328	17.519	5	0.218	15.270
Louvain	1	0.198	0.058	5	0.349	0.042	5	0.215	0.044
Leiden	1	0.225	0.049	5	0.353	0.042	5	0.229	0.033
SA	1	0.224	93.035	5	0.357	67.526	5	0.242	64.672
LE	1	0.099	0.143	5	0.244	0.175	5	0.145	0.254

Algorithm	$\mu = 0.65$			$\mu = 0.8$		
	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$
Expected	5	0.021		5	-0.043	
CNM	5	0.238	15.158	5	0.209	17.967
Louvain	5	0.232	0.067	5	0.202	0.036
Leiden	5	0.245	0.084	5	0.217	0.091
SA	5	0.262	71.017	5	0.233	77.817
LE	5	0.161	0.319	5	0.132	0.270

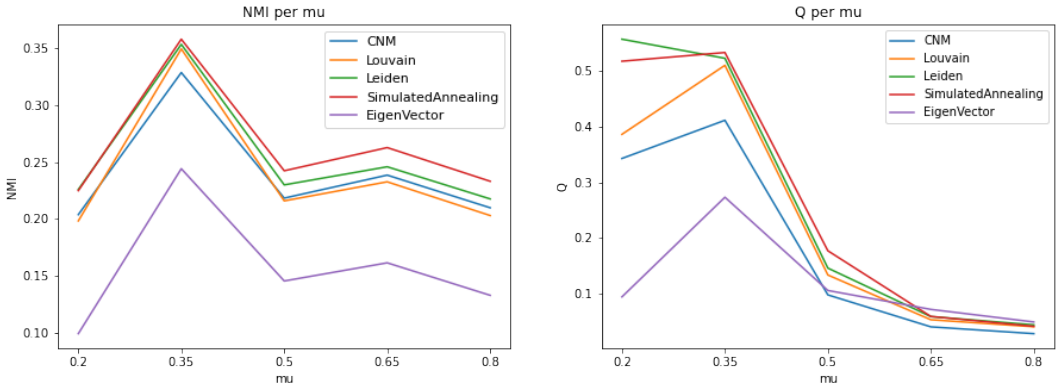

Figure 1: average NMI and modularity for test cases where $N = 1000$ and $<k> = 10$

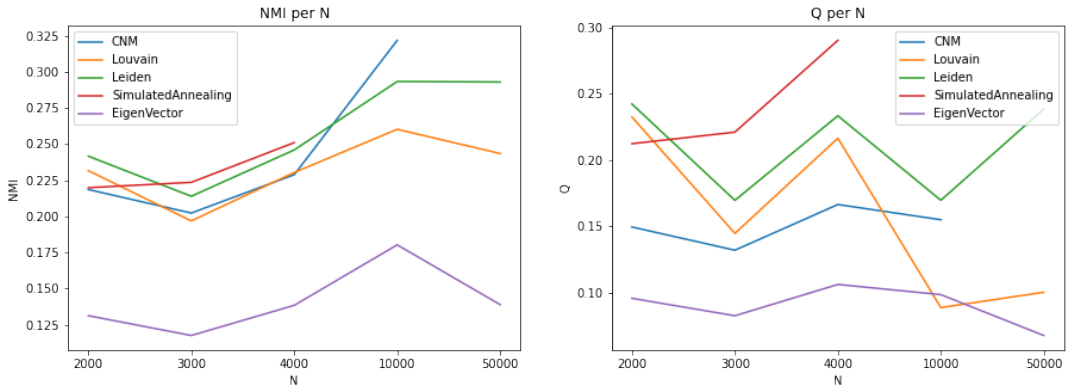

Figure 2: average NMI and modularity for test cases where $\mu = 0.5$ and $<k> = 10$

At a glance, from Figures 1 and 2 it can be inferred that Simulated Annealing method is outperforming other algorithms in both NMI and Modularity factors. However, Figure 3 reveals drawbacks of this algorithm regarding

Table 2: Algorithms' Performance for different cases where $\mu = 0.5$ and $<k> = 10$

Algorithm	$N = 2000$			$N = 3000$			$N = 4000$		
	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$
Expected	5	0.174		5	0.133		5	0.184	
CNM	5	0.218	73.214	5	0.202	168.150	5	0.228	297.048
Louvain	5	0.231	0.092	5	0.196	0.224	5	0.230	0.296
Leiden	5	0.241	0.154	5	0.213	0.249	5	0.246	0.322
SA	4	0.219	280.97	5	0.223	486.259	5	0.251	589.507
LE	5	0.131	0.630	5	0.117	1.399	5	0.138	1.986

Algorithm	$N = 10000$			$N = 50000$		
	cases	$\overline{Q}$	$\overline{time}$	cases	$\overline{Q}$	$\overline{time}$
Expected	5	0.211		5	0.242	
CNM	2	0.321	822.579		—	
Louvain	5	0.260	0.320	5	0.243	2.679
Leiden	5	0.293	0.329	5	0.293	2.256
SA		—			—	
LE	5	0.180	7.979	5	0.138	98.971

execution time.
Furthermore, it can be concluded from Figure 3 that parameter μ does not have any significant impact on algorithms' execution time. On the other hand, according to the Figure 1 increasing probability of existing extra-community links (μ), decreases algorithms' performance in maximizing Modularity drastically. This phenomenon can be due to the intrinsic feature of the network.
Finally, Figure 2 shows that by increasing size of the network, performance of the Louvain algorithm. This might be a result of the badly connected communities emerged in Louvain steps, which is guaranteed to be solved in Leiden algorithm [10]. In spite of the fact that Leiden algorithm is derived from Louvain, Leiden does not show this pattern. Thus experimental data supports this argument.

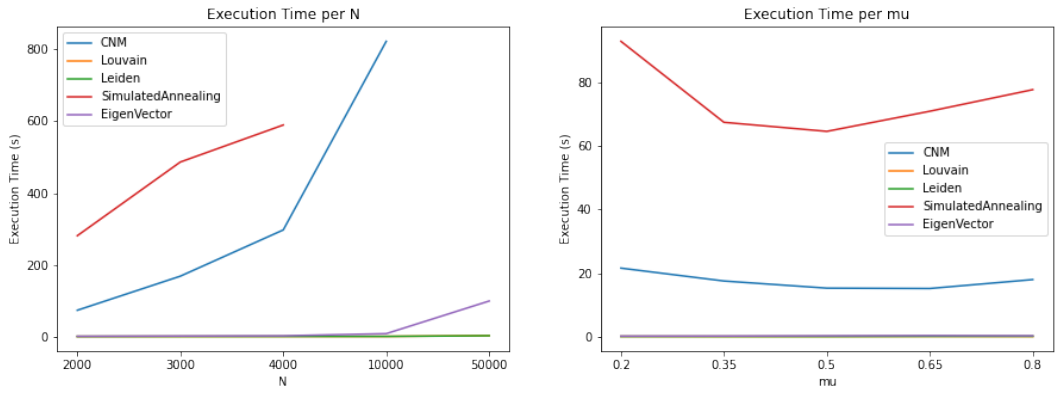

Figure 3: average execution time for test cases where $<k> = 10$

4. Conclusion

To sum up, if N (size of the network) is small enough, it is suggested to use Simulated Annealing algorithm, unless the network is not connected. In this case, Leiden has exhibited promising performance. It is also suggested to use Leiden algorithm for large networks.

References

[1] V. A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Phys. Rev. E*, 84:016114, Jul 2011. doi: 10.1103/PhysRevE.84.016114. URL https://link.aps.org/doi/10.1103/PhysRevE.84.016114.

[2] Liudmila Ostroumova and Alexey Tikhonov. Community detection through likelihood optimization: In search of a sound model. *The World Wide Web Conference*, 2019.

[3] Mingming Chen, Konstantin Kuzmin, and Boleslaw K. Szymanski. Community detection via maximization of modularity and its variants. *IEEE Transactions on Computational Social Systems*, 1(1):46–65, 2014. doi: 10.1109/TCSS.2014.2307458.

[4] M. E. J. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E*, 94:052315, Nov 2016. doi: 10.1103/PhysRevE.94.052315. URL https://link.aps.org/doi/10.1103/PhysRevE.94.052315.

[5] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007. ISSN 0027-8424. doi: 10.1073/pnas.0605965104. URL https://www.pnas.org/content/104/1/36.

[6] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, Apr 2010. doi: 10.1103/PhysRevE.81.046106. URL https://link.aps.org/doi/10.1103/PhysRevE.81.046106.

[7] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008. doi: 10.1103/PhysRevE.78.046110. URL https://link.aps.org/doi/10.1103/PhysRevE.78.046110.

[8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004. doi: 10.1103/PhysRevE.70.066111. URL https://link.aps.org/doi/10.1103/PhysRevE.70.066111.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. doi: 10.1088/1742-5468/2008/10/p10008. URL https://doi.org/10.1088/1742-5468/2008/10/p10008.

[10] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), Mar 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-41695-z. URL http://dx.doi.org/10.1038/s41598-019-41695-z.

[11] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Phys. Rev. E*, 74:016110, Jul 2006. doi: 10.1103/PhysRevE.74.016110. URL https://link.aps.org/doi/10.1103/PhysRevE.74.016110.

[12] Johan Dahlin and Pontus Svenson. Ensemble approaches for improving community detection methods. *ArXiv*, abs/1309.0242, 2013.

[13] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006. doi: 10.1103/PhysRevE.74.036104. URL https://link.aps.org/doi/10.1103/PhysRevE.74.036104.

[14] Songran Liu and Zhe Li. A modified genetic algorithm for community detection in complex networks. In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1–3, 2017. doi: 10.1109/ICAMMAET.2017.8186747.

[15] Saoud Bilal and Moussaoui Abdelouahab. Evolutionary algorithm and modularity for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 473:89–96, 2017. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2017.01.018. URL https://www.sciencedirect.com/science/article/pii/S0378437117300249.

[16] Xingyi Zhang, Kefei Zhou, Hebin Pan, Lei Zhang, Xiangxiang Zeng, and Yaochu Jin. A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks. *IEEE Transactions on Cybernetics*, 50(2):703–716, 2020. doi: 10.1109/TCYB.2018.2871673.

[17] NetworkX Developers. Lfr benchmark graph, 2021. URL `https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR_benchmark_graph.html`.