Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Performance Evaluation of Python Libraries for Community Detection on Large Social Network Graphs

Alif Dio Af'Ally[1], Fitriyani[2]
alifdio@student.telkomuniversity.ac.id, fitriyani@telkomuniversity.ac.id
School of Computing, Telkom University

Article Information

Submitted : 13 May 2024
Reviewed : 27 May 2024
Accepted : 15 Jun 2024

Keywords

Community Detection, Python Library, Runtime, Memory Usage, Modularity

Abstract

The development of social networking platforms has transformed the way people interact. The growing number of social media users generates vast amounts of data and creates communities among users that are interesting to analyze. Community detection has significant benefits in terms of understanding network structures and providing insights into these communities. However, the abundance of options can make it challenging to select the appropriate library and algorithm. Therefore, this study analyzes the effectiveness of community detection algorithms on social network graph datasets utilizing several Python libraries such as NetworkX, iGraph, Scikit-Network, and CDlib, as well as Louvain and Label Propagation algorithms. The results of this study indicate that iGraph is an optimal library, based on the execution time, memory usage, and user-friendliness. Additionally, the Louvain algorithm is effective for community detection and exhibits high modularity values.

A. Introduction

The emergence of social networking platforms has significantly changed the pattern of social interaction. Apart from facilitating communication, social media has also become an effective means of information exchange and an economic resource. According to DataReportal 2023, the global user base of social networking sites has reached 4.88 billion, equivalent to 60.6% of the world's population[1]. This huge impact of social media usage generates large and diverse data. Therefore, the processing of social network data becomes very important to gain valuable knowledge from the information contained in the data.

An example of social network problem is community detection. Community detection is the process of identifying groups or communities in a network consisting of interconnected nodes[2]. The purpose of community detection is to understand the network structure and identify groups that have similar connection patterns[3].

To perform community detection, resources and efficient algorithms are necessary[4]. Currently, there are many frameworks and libraries available for processing social network data, such as NetworkX, iGraph, Scikit-Network, CDlib, and many more. Selecting the appropriate library is crucial as it can influence the algorithm's performance in community detection.

Previous research conducted by Thomas Bonald[5] has resulted in a new library named Scikit-Network, designed for graph analysis. This library was compared with other libraries such as NetworkX, iGraph, and Graph-tool using algorithms like Louvain, Pagerank, HITS, and Spectral, focusing on runtime and memory efficiency. There are limitations in this research, such as the runtime not being available in NetworkX when implemented with the Louvain algorithm.

Another study conducted by Giulio Rossetti has led to the development of a library named CDlib[6], a community detection library for social networks based on Python. CDlib provides access to 39 community detection algorithms, such as Walktrap, Label Propagation, Girvan-Newman, Louvain, and Infomap. CDlib also features evaluation and comparison capabilities, as well as diverse visualization capabilities. This makes CDlib a comprehensive library.

This study evaluates the performance of community detection algorithms on social networks using several Python-based libraries, considering runtime, memory usage, modularity, and user-friendliness. The community detection algorithms to be implemented are Louvain[7] and Label Propagation[8]. The libraries to be used are NetworkX, iGraph, Scikit-Network, and CDlib.

B. Research Methodology

This research is conducted by comparing the performance of four libraries, namely NetworkX, iGraph, Scikit-Network, and CDlib, in community detection on datasets obtained from the Stanford SNAP repository using two algorithms, Louvain and Label Propagation. The evaluation of library and algorithm performance is carried out considering several factors such as runtime, memory usage, and modularity.

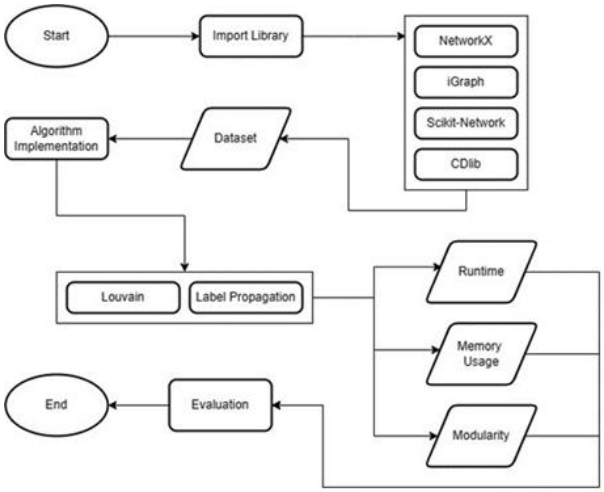

Figure 1. Research Flowchart

The research methodology depicted in Figure 1 above is divided into two distinct parts, detailed as follows:

1. **Implementation**

   The implementation stage in this research covers several parts as listed below:

   a. Import Library

   At this stage, the import process is carried out from four libraries used for community detection. Where in the four libraries there are already functions for the Louvain and Label Propagation algorithms. the four libraries are *NetworkX*[9], *iGraph*[10], *Scikit-Network*[5], and *CDlib*[11]. This import process is carried out on the device used, which is a laptop with certain specifications: Intel Core i5-8265U, 12GB DDR4 RAM, and 1TB HDD storage. The operating system used is Windows 11 Home Single Language 64-bit.

   b. Import Dataset

   At this stage, the import process of datasets that have been obtained from the Stanford SNAP repository is carried out, including the com-Youtube, com-DBLP, and com-Amazon datasets[12]. Which in the data consists of 2 features, namely FromNodeId and ToNodeId. In Table 1 below can be seen the details of each dataset used.

Table 1. Details of Datasets

Dataset	Type	Nodes	Edges	Size
com-Youtube	Undirect	1,134,890	2,987,624	37,8 MB
com-DBLP	Undirect	317,080	1,049,866	13,6 MB
com-Amazon	Undirect	334,863	925,872	12,3 MB

   c. Algorithm Implementation

In this stage, the implementation of the Louvain[7] and Label Propagation[13] algorithms is carried out using clustering functions in the NetworkX, iGraph, Scikit-Network, and CDlib libraries. Details of Louvain functions from each library can be seen in Table 2 below:

Table 2. Louvain function of each library

Louvain	
NetworkX	coms = nx.community.louvain_communities()
iGraph	coms = g.community_multilevel()
Scikit-Network	coms = Louvain()
CDlib	coms = algorithms.louvain()

Meanwhile, the function to implement the Label Propagation algorithm can be seen in Table 3 below:

Table 3. Label Propagation Functions for Each Library

Label Propagation	
NetworkX	coms = nx.community.louvain_communities()
iGraph	coms = g.community_label_propagation()
Scikit-Network	coms = PropagationClustering()
CDlib	coms = algorithms.label_propagation()

2. Evaluation

The evaluation in this research is conducted to assess the performance of each library used. Several evaluations carried out in this study include:

a. Runtime Measurement

In this stage, runtime measurement is conducted from the start of the program execution until its completion. The final result is calculated by subtracting the end timestamp from the start timestamp. This time measurement will be performed using the **"time"** function with the output formatted in seconds[14].

b. Memory Usage Measurement

Memory usage measurement is also conducted from the start of the program execution until its completion, where the final result is calculated by subtracting the end memory amount from the initial memory amount. The measurement is performed using the **"psutil.memory_info()"** function with the output formatted in MB (Megabytes)[15].

c. Modularity Measurement

Modularity measurement aims to assess the quality of the formed communities. The modularity value ranges from -1 to 1. If the modularity value approaches -1, the resulting communities are considered poor. Conversely, if the modularity value approaches 1, the communities are considered good[16]. To calculate modularity, functions provided by the four libraries are used. Some of these functions can be seen in Table 4 below:

Table 4. Modularity Functions for Each Library

Modularity	
NetworkX	Mod = modularity()
iGraph	Mod = g.modularity()
Scikit-Network	Mod = sknetwork.clustering.get_modularity()
CDlib	Mod = evaluation.erdos_renyi_modularity()

C. Result & Analysis

The results obtained from this study include measurements of Runtime, Memory Usage, and Modularity from implementations using the Louvain algorithm and label propagation. The detailed results and analysis of this research are as follows:

1. Runtime

Runtime testing is conducted to measure the time required to execute the program code that has been created. The unit used in the measurement is seconds, and the results of the Louvain runtime measurements can be seen in Table 5 below:

Table 5. Louvain Runtime Results

RUNTIME LOUVAIN						
Dataset	**Node**	**Edge**	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	1,134,890	2,987,624	212	**30**	121	444
com-DBLP	317,08	1,049,866	69	13	**12**	154
com-Amazon	334,863	925,872	60	14	**11**	95

From the above test results, it can be seen that the library with the best execution time is dominated by Scikit-Network, as evidenced by its execution time on the Com-

Amazon dataset for 11 seconds and on the Com-DBLP dataset for 12 seconds. However, on the Com-Youtube dataset, which has the largest number of nodes and edges, the best execution time is obtained by iGraph with an execution time of 30 seconds, which is faster than Scikit-Network, which took 121 seconds to process the dataset. In addition to the Louvain algorithm, runtime testing is also conducted on the Label Propagation algorithm. The test results can be seen in Table 6 below.

Table 6. Label Propagation Runtime Results

RUNTIME LABEL PROPAGATION						
Dataset	**Node**	**Edge**	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	1,134,890	2,987,624	156	274	**85**	163
com-DBLP	317,08	1,049,866	117	119	**14**	123
com-Amazon	334,863	925,872	**32**	39	50	33

From the test results above, it can be observed that Scikit-Network excels on 2 datasets, as evidenced by its execution time when performing community detection on the Com-DBLP dataset, which took 14 seconds, and on the Com-Youtube dataset, which took 85 seconds. However, for the Com-Amazon dataset, which has the smallest number of nodes and edges, the community detection process can be performed very quickly by NetworkX, taking only 32 seconds. This duration is faster than Scikit-Network, which took 50 seconds to process the dataset.

Scikit-Network generally exhibits the best execution time performance in most cases, except for the Com-Youtube dataset, which has a large number of nodes and edges when using the Louvain algorithm, and the Com-Amazon dataset analyzed with the Label Propagation algorithm. The reasons for the inefficiency of Scikit-Network in these cases include the use of the NetworkX module for graph processing, which adds complexity to the code, making it less optimal for large or complex datasets. This confirms that Scikit-Network is more suitable for smaller datasets or simpler structures. On the other hand, iGraph shows better performance on the Com-Youtube dataset, with faster execution times compared to Scikit-Network. This indicates that iGraph is effective in detecting communities in complex or large datasets, thanks to its compact and self-contained code structure, which does not require additional modules for graph processing, unlike Scikit-Network and CDlib.

2. Memory Usage

Memory Usage testing is conducted to measure how much memory the device consumes when the program execution process is running. The unit used in the measurement is megabytes, and the results of the Louvain Memory Usage measurement can be seen in Table 7 below:

Table 7. Memory Usage Louvain

MEMORY USAGE LOUVAIN						
Dataset	**Node**	**Edge**	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	1,134,890	2,987,624	2021	**580**	1989	2339
com-DBLP	317,08	1,049,866	915	**203**	645	1243
com-Amazon	334,863	925,872	1248	**181**	588	1114

From the test results above, it is evident that the best memory efficiency is achieved by iGraph. The implementation results of iGraph show relatively small memory sizes with 181 MB for the Com-Amazon dataset, 203 MB for Com-DBLP, and 580 MB for Com-Youtube when using the Louvain algorithm. Meanwhile, Scikit-Network also exhibits fairly good memory efficiency, although not as good as iGraph. In the community detection implementation with the Louvain algorithm, Scikit-Network records memory usage of 588 MB for Com-Amazon, 645 MB for Com-DBLP, and 1989 MB for Com-Youtube. In addition to the Louvain algorithm, testing is also conducted on the Label Propagation algorithm, with the results shown in Table 8 below:

Table 8. Memory Usage Label Propagation

MEMORY USAGE LABEL PROPAGATION						
Dataset	**Node**	**Edge**	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	1,134,890	2,987,624	1968	**590**	1982	1912
com-DBLP	317,08	1,049,866	639	**210**	645	621
com-Amazon	334,863	925,872	584	**193**	594	563

From the test results above, it is evident that iGraph is the library with the most memory-efficient usage when using the Label Propagation algorithm for community detection. iGraph demonstrates a very good level of memory efficiency with relatively small memory sizes, namely 193 MB for the Com-Amazon dataset, 210 MB for Com-DBLP, and 590 MB for Com-Youtube. Although CDlib also exhibits decent memory efficiency, it is not as good as iGraph. CDlib records memory usage of 563 MB for Com-Amazon, 621 MB for Com-DBLP, and 1912 MB for Com-Youtube when using the Label Propagation algorithm.

iGraph has the advantage of efficient memory usage, especially on the Com-Amazon and Com-DBLP datasets, with relatively small memory sizes compared to other libraries. This indicates that iGraph can be a good choice in situations where memory resources are limited. Meanwhile, Scikit-Network exhibits higher memory usage levels but can still be considered in situations where execution time efficiency is more important than memory usage.

3. Modularity

Modularity measurement is also conducted on the results obtained during the implementation of community detection using the Louvain and Label Propagation algorithms. The results of the Louvain modularity measurement can be seen in Table 9 below:

Table 9. Louvain Modularity Testing Results

MODULARITY LOUVAIN				
Dataset	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	0,721	0,713	0,684	**0,762**
com-DBLP	0,822	0,821	0,809	**0,824**
com-Amazon	0,926	0,926	0,925	**0,927**

From the test results above, the final results of the modularity measurement process in the implementation of the Louvain algorithm are obtained. From these results, it can be found that among the four libraries used in community detection, CDlib is quite significant and has high modularity values on three types of datasets, namely 0.927 for the Com-Amazon dataset, 0.824 for the Com-DBLP dataset, and 0.762 for the Com-Youtube dataset. Meanwhile, the modularity results from the implementation of the Label Propagation algorithm can be seen in Table 10 below.

Table 10. Label Propagation Modularity Results

MODULARITY LABEL PROPAGATION				
Dataset	**NetworkX**	**iGraph**	**Scikit-Network**	**CDlib**
com-Youtube	0,329	**0,667**	0,043	0,604
com-DBLP	0,649	**0,683**	0,670	0,654
com-Amazon	0,726	0,785	**0,833**	0,726

From the test results above, the final results of the modularity measurement process in the implementation of the Label Propagation algorithm are obtained. From these results, it can be found that iGraph has excellent modularity values on 2 datasets, with scores as follows: 0.683 for the Com-DBLP dataset and 0.667 for the Com-Youtube dataset. Meanwhile, on the Com-Amazon dataset, which is the dataset with the smallest number of nodes and edges, the highest modularity value is obtained by Scikit-Network with a score of 0.833 for the Com-Amazon dataset.

In the modularity testing using the Louvain algorithm, CDlib obtained high modularity values on all tested datasets. This is due to the effective optimization method of the Louvain algorithm in finding communities with optimal modularity values by considering network modularity and attempting to maximize it through node movement between communities. Meanwhile, the modularity results from the Label Propagation algorithm show greater variation between libraries and datasets. This occurs because of the simpler nature of the Label Propagation algorithm compared to Louvain, with an iterative approach where nodes adopt the label of their most dominant neighbor. The significant difference between Louvain and Label Propagation is also influenced by the different approaches in identifying communities. Louvain uses a partitioning approach, while Label Propagation uses a labeling approach. Additionally, variations in library implementations and the complexity of the tested dataset structures also play a role in these differences. Therefore, Louvain tends to excel in identifying communities with high modularity values.

D. Discussion

Based on the experience gained during the implementation of libraries in this study, it was found that all four community detection libraries used, namely NetworkX, iGraph, Scikit-Network, and CDlib, could support the research well, especially in providing functions for Louvain and Label Propagation algorithms. These libraries also have good documentation that is easily accessible through their respective official websites, thus aiding in learning about them.

NetworkX and iGraph, which have long been used in graph network analysis, offer a wealth of learning resources that facilitate new users in adopting and implementing them in various cases. This ease contrasts with Scikit-Network and CDlib, which are still relatively new in their applications, thus limited learning resources pose their own challenges.

This research also faced difficulties in the implementation process, which required parameter adjustments and a deep understanding of the input format required by each library. For example, the need to convert adjacency matrices into CSR (Compressed Sparse Row) format to be processed by functions in Scikit-Network, indicates a higher level of complexity in the data preparation stage.

Integration between several libraries, such as the use of the NetworkX module by Scikit-Network and CDlib for graph creation, leads to increased memory usage and longer execution times. In contrast, iGraph with its independent graph creation functions offers potentially more efficient performance solutions.

This discussion highlights the importance of choosing the right library based on project needs, the availability of learning resources, and system performance requirements. Although all libraries offer the necessary functions, a deep understanding of their strengths and weaknesses can help optimize the implementation process and minimize potential barriers.

E. Conclusion

The following conclusions are derived based on the results of the experiments that were conducted. Firstly, iGraph is recommended as an optimal choice considering its comprehensive features, abundant availability of reference sources, ability to handle large datasets quickly, and efficient resource usage, with the highest memory usage being only 580 MB for the Louvain algorithm and 590 MB for the Label Propagation algorithm. Thus, it is highly suitable for research requiring community analysis under resource constraints. Secondly, The Louvain algorithm demonstrates superior performance in identifying communities with high modularity across various types of datasets. For instance, on the Com-Amazon dataset, Louvain achieved the highest modularity value of 0.927, while on the Com-DBLP and Com-Youtube datasets, it reached 0.824 and 0.762, respectively. Therefore, Louvain can be considered an ideal algorithm for community detection analysis in complex networks. Thirdly, The selection of libraries and algorithms in community detection can be tailored to the characteristics of the data used, the device utilized, and the needs and objectives of the analysis to be achieved. Lastly, Modularity values can be influenced by the number of communities generated and the algorithm used, so differences in algorithms can have a significant impact.

F. References

[1] Simon Kemp, "DIGITAL 2023: GLOBAL OVERVIEW REPORT." Accessed: Oct. 14, 2023. [Online]. Available: https://datareportal.com/reports/digital-2023-global-overview-report

[2] V. da F. Vieira, C. R. Xavier, and A. G. Evsukoff, "A comparative study of overlapping community detection methods from the perspective of the structural properties," *Appl Netw Sci*, vol. 5, no. 1, Dec. 2020, doi: 10.1007/s41109-020-00289-9.

[3] A. K. Verma, M. Jadeja, and S. Jayaswal, "RW-HeCo: A random walk and network centrality based graph neural network for community detection in heterogeneous networks," *Multimed Tools Appl*, 2024, doi: 10.1007/s11042-024-18823-7.

[4] J. Sheng, C. Liu, L. Chen, B. Wang, and J. Zhang, "Research on community detection in complex networks based on internode attraction," *Entropy*, vol. 22, no. 12, pp. 1–16, Dec. 2020, doi: 10.3390/e22121383.

[5] T. Bonald, N. de Lara, Q. Lutz, T. Paris, and B. Charpentier BERTRANDCHARPENTIER, "Scikit-network: Graph Analysis in Python," *Journal of Machine Learning Research*, vol. 21, pp. 1–6, 2020, [Online]. Available: https://scikit-network.readthedocs.io/en/latest/.

[6] G. Rossetti, L. Milli, and R. Cazabet, "CDLIB: a python library to extract, compare and evaluate communities from complex networks," *Appl Netw Sci*, vol. 4, no. 1, Dec. 2019, doi: 10.1007/s41109-019-0165-9.

[7] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," Mar. 2008, doi: 10.1088/1742-5468/2008/10/P10008.

[8] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Sep. 2007, doi: 10.1103/PhysRevE.76.036106.

[9] M. J. Olsen, "Community Detection in Large Social Networks," 2014. Accessed: Oct. 09, 2023. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/259357/742891_FULLTEXT01.pdf

[10] F. B. De Sousa and L. Zhao, "Evaluating and comparing the IGraph community detection algorithms," in *Proceedings - 2014 Brazilian Conference on Intelligent Systems, BRACIS 2014*, Institute of Electrical and Electronics Engineers Inc., Dec. 2014, pp. 408–413. doi: 10.1109/BRACIS.2014.79.

[11] R. Maivizhi, S. Sendhilkumar, and G. S. Mahalakshmi, "A survey of tools for community detection and mining in social networks," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Aug. 2016. doi: 10.1145/2980258.2980408.

[12] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities based on Ground-truth," May 2012, [Online]. Available: http://arxiv.org/abs/1205.6233

[13] S. E. Garza and S. E. Schaeffer, "Community detection with the Label Propagation Algorithm: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 534. Elsevier B.V., Nov. 15, 2019. doi: 10.1016/j.physa.2019.122058.

[14] influxdata, "What is the Time Library in Python? A Helpful Guide." Accessed: Jan. 20, 2024. [Online]. Available: https://www.influxdata.com/blog/what-is-time-library-in-python-helpful-guide/

[15] geeksforgeeks, "Psutil module in Python." Accessed: Jan. 24, 2024. [Online]. Available: https://www.geeksforgeeks.org/psutil-module-in-python/

[16] M. Chen, K. Kuzmin, and B. K. Szymanski, "Community Detection via Maximization of Modularity and Its Variants," Jul. 2015, doi: 10.1109/TCSS.2014.2307458.