Received: June 17, 2022. Revised: August 27, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

GN-PPN: Parallel Girvan-Newman-Based Algorithm to Detect Communities in Graph with Positive and Negative Weights

Neny Sulistianingsih[1,2]* **Edi Winarko[2]*** **Anny Kartika Sari[2]**

[1]*Departement of Engineering and Design, Universitas Bumigora, Indonesia*
[2]*Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia*
* Corresponding author's Email: ewinarko@ugm.ac.id

Abstract: The Girvan-Newman (GN) method is one of the most popular methods for detecting communities. However, the method is applied to graphs with only positive weights, while graphs with positive and negative weights are in the real world. This paper proposes improving the GN method to work on graphs with positive and negative weights. The method is called Girvan-Newman Parallel Positive Negative (GN-PPN). Other than the benefit related to the use of negative weights, GN-PPN has a parallelization process that accelerates processing time. This method also uses edge betweenness and pair dependencies as calculations. The GN-PPN method is evaluated using modularity score, Normalized Mutual Information (NMI), and processing time. Our experiments show that the modularity score of the GN-PPN is similar to GN, modified GN (mGN), and Parallel GN (PGN). The accuracy of the GN-PPN method in detecting communities evaluated by NMI shows that the GN-PPN is better than similar methods such as Infomap, which ranges from 0.651 to 0.937. The processing time of the GN-PPN method shows a significant acceleration, which is a decrease of around 45% to 95% compared to other methods such as GN, mGN, and PGN.

Keywords: Community detection, Edge betweenness, Girvan-newman, Pair dependency, Parallel processing, Weighted graphs.

1. Introduction

Girvan-Newman (GN) is one of the benchmark methods for community detection. The GN method uses the concept of edge betweenness and Dijkstra for marking each connected node in the graph. After the highest edge betweenness value is found, the nodes with the highest value will be deleted. The edge betweenness calculation again will be carried out on the graph with the node pairs that have been removed. Furthermore, the GN method also introduces modularity, which measures the quality of the formed community. Many researchers have proved the GN method to produce a high-accuracy community [1, 2].

Despite the advantages of the GN method, it still has problems. One of the problems is that the modularity optimization is NP-hard, and this method is unsuitable for large graphs because of the long processing time [3]. The weaknesses make research related to this method continue to be carried out. The goal of ongoing research [4] is to reduce this method's processing time and accuracy [5], especially in the era of big data. The graphs representing future data are likely to grow larger.

Furthermore, problems related to community detection have not been solved. One of them is the application of community detection to graphs with negative weights. Most existing methods related to community detection only consider graphs with positive weights, while there are also graphs with negative weights. An example of such graphs is in the research conducted by [5, 6]. The weight values of the edges in the graphs range from -10 (strongly distrust) to 10 (strongly trusts), representing the Bitcoin users' confidence in each other. Studies on community detection using graphs with positive and negative weights are still rare. The lack is because

the algorithms used to detect the community only work on graphs with positive weights. We must transform the negative weights into positive values when applied to graphs with positive and negative weights. This graph weight problem, combined with the long processing time, raises how to make the algorithm more flexible to the graph used but can still be processed quickly and maintain the accuracy of the resulting community results.

Based on the problems mentioned earlier, this paper proposes a new method of community detection based on the GN algorithm. The method is called GN's Parallel Positive Negative (GN-PPN). The main contributions of this proposed method are as follows:

- GN-PPN can process graphs with positive and negative weights. The method proposed in this study uses the Johnson algorithm to find the shortest path in graphs and re-weighting negative weight into positive weight. Furthermore, after all the weights on the graph become positive, several steps are carried out for community detection of the graph used
- GN-PPN uses the multiprocessing concept by dividing the existing graph into several subgraphs based on the number of threads entered at the beginning of the process. This concept is used to speed up the processing time of the proposed method.
- GN-PPN utilizes a parallelization process to calculate the edge betweenness and pair dependencies' value simultaneously. The value of edge betweenness and pair dependencies are calculated simultaneously for all pairs of vertices in the graph in each thread. Then the pair of nodes with the highest value betweenness edge and pair dependency from each thread is sent to the main host to select the pair of nodes with the highest value betweenness edge and pair dependency among all threads used. The following process will be done on the main host.

The rest of the paper is detailed as follows. Section 2 discusses some of the critical related works in community detection. Section 3 describes the GN method. The proposed GN-PPN algorithm is discussed in Section 4. Section 5 explains the scenario and result of the experiments, followed by the discussion in Section 6. Section 7 contains the conclusions and future works.

2. Related work

The GN method was first introduced by Girvan and Newman [7] and is still one of the most popular methods for community detection. The GN method uses the edge betweenness to find the betweenness value for nodes in the graph and removes the nodes with the highest value. Furthermore, the concept of modularity is used to measure the quality of the community generated from this algorithm. The modularity score value is between 0 to 1. The closer the modularity value is to 1, the better the resulting community results [8, 9].

Research has been carried out and developed in various fields over the last decade. Studies such as the ones conducted by Budic et al. [2] use GN to detect complex radio access network (RAN) communities, Yadav et al. [10] for social network analysis, and Saputri et al. [11] for student behavior based on students' browsing habits who use wi-fi at the university under study. Furthermore, studies employing GN are conducted, such as [12, 13]. These studies prove that the accuracy of the GN method is better than other methods [12, 13].

However, the process of calculating the edge betweenness in GN has the complexity of $O(en)$, so that the overall complexity of the GN method is $O(e^2n)$. Furthermore, the GN method is also inefficient for extensive data [14].

Based on these problems, several studies were conducted to reduce the processing time of the GN method. Research [13] removed the multi-edge with the highest edge betweenness value in his research. This study shows that the processing time of the enhanced GN method is faster than the GN method. Research [15] modifies the GN method by combining edge betweenness and pair dependencies calculations. Furthermore, other studies such as that conducted by [8, 16, 17] modify GN using the concept of parallelization, either by adding the concept of MapReduce [16], based on GPU [17] and dividing the graph into several components [8]. These studies can speed up processing time when compared to the GN method.

However, the previous studies used graph datasets with positive weight edges. Whereas, in the real world, there are also graphs with positive and negative weights. The current research focuses on speeding up computational time and on datasets processed by proposed methods. The study by developing GN with the proposed method aims to process datasets with positive weights and those that weigh positive and negative. State-of-art techniques and current research can be seen in Table 1.

3. The GN method

A graph $G = (V, E)$ is a set of vertices $V = \{v_1, v_2, v_3, \dots, v_n\}$ and edges $E = \{e_1, e_2, e_3, \dots, e_n\}$.

Table 1. State-of-art techniques and current research

No.	Research	Betweenness Used	Concept of Time Acceleration Used	Shortest Path Method
1.	Girvan and Newman [7]	Edge Betweenness	No	Dijkstra
2.	Jisha *et al.* [13]	Edge Betweenness	Multiple erase edge betweenness	Dijkstra
3.	Bocu and Tabirca [15]	Edge betweenness and Pair dependencies	No	Dijkstra
4.	Moon *et al.* [16]	Edge Betweenness	MapReduce	Dijkstra
5.	Fan *et al.* [17]	Edge Betweenness	GPU Based	Dijkstra
6.	Jamour *et al.* [8]	Edge Betweenness	Divided graph into several components	Dijkstra
7.	Current research	Edge betweenness and Pair dependencies	Divided graph into several components	Johnson

Edge consists of pairs of nodes that are connected. Each of these edges can have weight (w). The weights on these edges represent the intensity of the interaction between pairs of nodes.

The definition of a community is a subgraph of graph G in which the interactions between nodes with other nodes (intra-connected) inside the community are solid. In contrast, the interactions with nodes in other communities (inter-connected) are weak [18]. An example of a community detection application can be seen in the study by Ljucovi & Tomovic [19]. In [19], a study on the collaboration of authors at the University of Montenegro is conducted. Nodes represent authors, and edges represent a collaboration between researchers. Edges will form if two or more authors publish a paper together. A community creates based on a network where authors write on the same topic. The stronger the intra-connected relationship and the weaker the inter-connected relationship, the better the resulting community will be.

The GN method used in conducting community detection consists of three steps as follows:

1. Calculate the edge betweenness for each edge in the graph
2. Remove the edge with the highest edge betweenness value
3. Recalculate the edge betweenness for the remaining edges
4. Repeat steps 1-3 until all edges are removed.

The GN method uses several different short path search methods based on the type of graph used to visit each node in the graph and then calculate edge betweenness. The GN method will use Breadth-First Search (BFS) if the graph is unweighted. The Dijkstra algorithm is used to find the shortest paths if the graph is a weighted graph. Because of this, The GN method can only process graphs with positive weights since the Dijkstra algorithm only works in graphs with positive weights.

The quality of the community is measured using modularity. Modularity has a range score between -1 to 1 [20]. Modularity measures the density of links within a community compared to links between communities. The modularity value can be negative if there are fewer links in the community than the existing probability base and the number of links among the communities formed is greater than the expected base. The modularity value will increase with the more robust the formed structural community. However, in its implementation, the communities formed have average modularity ranging from 0.30 to 0.70 [20, 21]. Modularity higher than this value is rarely found in GN community detection tests. Modularity's formula specifies in Eq. (1) [20, 21].

$$Q = \frac{1}{2m} \times \sum ij \left[a_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \qquad (1)$$

Here, a_{ij} is the adjacency matrix from vertex i to j, m is the number of edges in the graph, k_i is the degree of vertex i, and k_j is the degree of vertex j. Meanwhile, $\delta(c_i, c_j)$ can be determined as follows:

$$\delta(c_i, c_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \qquad (2)$$

The value of $\delta(c_i, c_j)$ is 1 if nodes i and j are in the same community, otherwise, 0 when nodes i and j are different.

4. The proposed GN-PPN method

This section explains the proposed algorithm, the GN-PPN method. The proposed algorithm can detect graphs with positive and negative weights. A multiprocessing concept is used in the proposed GN-PPN method to speed up the processing. Besides that, the parallelization process is also used to enhance the accuracy of the GN method.

In the proposed GN-PPN method, the graph that is processed is not only a graph with a positive weight but also a graph with a negative weight. Johnson's algorithm finds the shortest path in all pairs of nodes in the graph and re-weight every weight in the graph with negative and positive weights. Johnson's algorithm consists of several stages as follows.

1. Add a new vertex s to the graph connected to all existing vertices and weights 0.
2. Use Bellman Ford's algorithm to find the shortest path in the graph starting from the new vertex s to all vertices v with a minimum weight $h(v)$ of a path from s to v. The Bellman-Ford algorithm chooses because this method can work on graphs with negative values. In addition, the Bellman-Ford algorithm can detect cycles, and if the graph contains cycles, the search for the shortest path will be stopped.
3. Reweight the edges of the initial graph using the values obtained from Bellman Ford's algorithm. The new weight value is calculated using Eq. (3).

$$w'(u,v) = w(u,v) + h(u) - h(v) \qquad (3)$$

$w'(u,v)$ is the re-weighted value from node u to node v, $w(u,v)$ is the initial weight value from node u to v, $h(u)$ is the length of the shortest path from s to u, and $h(v)$ is the length of the shortest path from s to v.

4. Remove s, and use Dijkstra's algorithm to find the shortest path in the re-weighted graph.

After all the weights become positive, the next steps of the GN-PPN method can be carried out. The GN-PPN consists of three main steps. These steps are as follows.

1. Divide the graph into subgraphs based on the number of threads entered using the multiprocessing concept
2. Calculate edge betweenness and pair dependency for every pair of nodes simultaneously
3. Erase the node pair with the highest edge betweenness if one node contains the highest pair dependency value.
4. Repeat Steps 1 to 3 until all edges are removed.

4.1 Step I: Divide the graph into subgraphs based on the number of threads entered using the multiprocessing concept

The next stage of the GN-PPN method is to divide the graph into subgraphs based on the number of threads entered. The concept used in dividing the graph into subgraphs is multiprocessing. The multiprocessing concept in the GN-PPN method uses CPU cores simultaneously and does not share resources. The graph is divided into several subgraphs based on the number of threads, and then each subgraph is assigned to each thread. Then in each thread, the edge betweenness and pair dependencies are calculated. The multiprocessing concept is implemented using a process-based parallelism library from Python, namely multiprocessing. The multiprocessing concept of the proposed method shows in Fig. 1.

4.2 Step II: Calculate edge betweenness and pair dependency simultaneously

At this stage, the parallelization process is used in the proposed GN-PPN method, namely calculating the edge betweenness and pair dependency in parallel. If the GN method only calculates edge betweenness, then in GN-PPN, pair dependencies will also be calculated. The process of calculating edge betweenness and pair dependencies is carried out on each subgraph that is in each thread used. Each thread will produce two results based on these two calculations: the node pair with the highest edge betweenness value and the node with the highest pair dependencies value. The two results will then be sent to the main host CPU and used for the next stage of the GN-PPN method.

Edge betweenness is the number of shortest paths between all nodes in the network that pass through the edges [21]. The more often the shortest path between nodes on the network passes through the edge, the greater the edge betweenness of that edge will be.

The pair dependency is calculated using the formula below [15].

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}} \qquad (4)$$

σ_{st} denotes the number of shortest paths from s to t and $\delta_{st}(v)$ is the number of shortest paths from s to t which go through v [15].

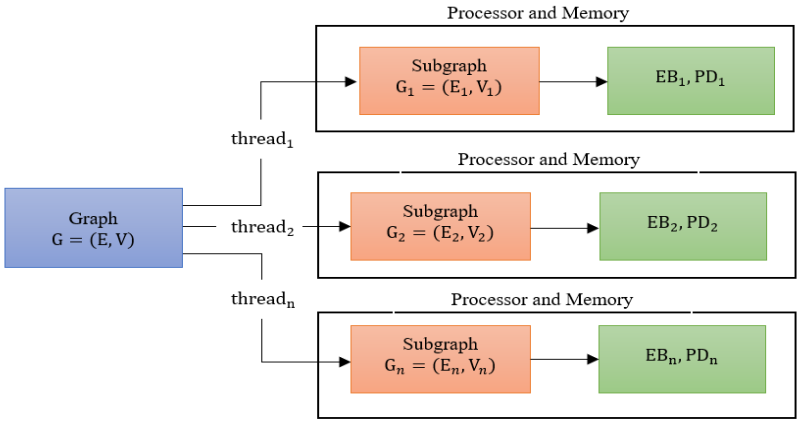


Figure. 1 Multiprocessing concept of the proposed GN-PPN method

4.3 Step III: Erase node pair with the highest values of edge betweenness and pair dependency

The last step in GN-PPN is performed on the main host CPU. From the two values sent by each thread, the node pair with the highest edge betweenness value and the node with the highest pair dependencies will be selected. Next, if a node (a) with the highest pair dependency value is the same as the node in the edge (a, b) with the highest edge betweenness value, the nodes are deleted. If several pairs of nodes meet this condition, those pairs of nodes will be deleted simultaneously. In the case there is no pair of nodes with the highest value betweenness and pair dependencies, matching is used for the pair of nodes with the seconds, the third-highest value of edge betweenness and pair dependencies until the node pair is deleted. Like the original GN method, the modularity calculation is then carried out to measure the quality of the resulting communities.

5. Experiments and results

This section describes the dataset, experimental scenarios, and results of the experiments.

5.1 Dataset

The dataset used two types of datasets for evaluating the GN-PPN method. The first type is five graph datasets with only positive weights, i.e., Celegens [22], Jazz [23], Coauthorship [24], and Usairports by Pajek Dataset. The second type is two graph datasets with positive and negative weights, i.e., BitcoinAlpha and BitcoinOTC [5, 6]. Not all BitcoinAlpha and BitcoinOTC datasets are used in the evaluation; only part is used due to the computation limitation. We experimented with different edges for these two dataset types to see how the computation time progresses as the graph size increases. For example, with the BitcoinOTC dataset, we took different edges, starting from 113; the data is labeled oct100.

5.2 Experiment scenario

To evaluate our proposed method, we performed two sets of experiments. The first set of experiments is to compare the performance of GN-PPN with GN in creating the community on data containing positive weights.

The second set of experiments is to compare the performance of the proposed method (GN-PPN) with GN [7], modified GN (mGN) [15], and parallel GN (PGN). The mGN method is a modified version of the GN method. Modifications are made by calculating edge betweenness in the GN method and pair dependencies as research conducted by Bocu and Tabirca [15]. The PGN method is a parallel version of the original GN method [8]. In PGN, the graph is divided into several subgraphs based on threads. The number of threads used in this evaluation is 2, 4, 8, and 16. Implementation of this concept was using a multiprocessing module from Python Standard Library. Furthermore, The number of threads is attached to the method name for easy identification. For example, PGN_2 means we used PGN with two threads, while GN-PPN_4 means we used GN-PPN with four threads.

In this second set of experiments, we measure the modularity, Normalized Mutual Information (NMI), and processing time required to perform community detection on the input data. For modularity, the formula used is as defined in Eq. (1).

NMI is used to measure the accuracy of the proposed method using mutual information of similarity (or dissimilarity) compared to the ground-truth method [25]. The NMI is defined in Eq. (5).

$$NMI(X|Y) = \frac{-2\sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} N_{ij}\log\frac{N_{ij}N}{N_{i.}N_{.j}}}{\sum_{i=1}^{C_X} N_{i.}\log\frac{N_{i.}}{N}+\sum_{j=1}^{C_Y} N_{.j}\log\frac{N_{.j}}{N}} \quad (5)$$

where C_X and C_Y denote the number of communities in partition X and Y. The matrix N has rows and columns corresponding to communities in X and Y, respectively. The element of N, N_{ij} is the number of mutual nodes between communities i and j. The terms $N_i.$ and $N_{.j}$ are the sum over rows and columns. The value of NMI ranges from 0 to 1, where NMI = 1 when partition X is identical to partition Y. If partition X is independent of partition Y, then NMI = 0.

The processing time is calculated from running the program until its results are displayed in seconds. For implementation of the methods, we use Python language.

5.3 Results

In this section, we discuss the result of our experiments. We performed experiments for each dataset and method five times and took the average modularity, NMI and execution time.

The comparison of community generated from GN-PPN and GN method can be seen in Fig. 1. The community formed from the GN and the GN-PPN methods has several differences. For example, in the Celegens data, the GN method generated 11 communities and nine communities for the GN-PPN method, in Jazz data generated seven communities for the GN method and four communities for the GN-PPN method, the Co-authorship data generated 273 communities for the GN method and 281 communities for the GN method. Usair97 data generated 22 communities for the GN method and 63 for the GN-PPN method. The Celegens data produced 29 and 116 isolated nodes, the Jazz data contained 30 and 27 isolated nodes, the Coauthorship data did not produce isolated graphs and 15 isolated graphs, and the usair97 data generated 81 and 47 isolated nodes for the GN and GN-PPN methods, respectively.

Although there are differences in the number of communities and isolated nodes formed, the GN and GN-NPP methods can detect the same large community in the data, such as the Celegens and Jazz data.

The comparison of the modularity using Dijkstra dan Johnson methods can be seen in Table 2 and 3, respectively. GN-PPN with the Dijkstra algorithm uses only positive weights (Table 2), while GN-PPN with the Johnson algorithm uses graph datasets with positive and negative weights (Table 3).

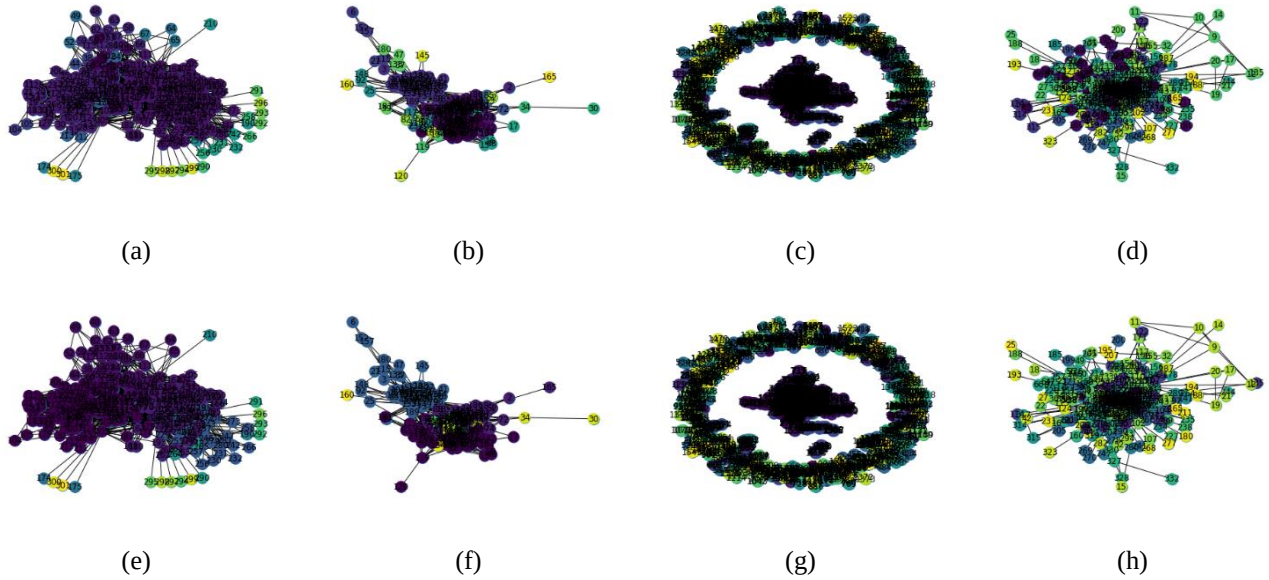
Modularity analysis of the GN-PPN and PGN methods shows that the GN-PPN method has better modularity than the PGN method on all threads. In addition, compared with the mGN and GN methods, the GN-PPN method shows stable results and is not much different from the two methods. The result shows that the quality of the community as measured by using modularity and resulting from the GN-PPN method is not much different from the GN and mGN methods.

We only use the data containing positive and negative weights combined with the Johnson method to evaluate the NMI and processing time. A comparison of NMI can be seen in Fig. 3. The evaluation results show that the NMI value of the GN-PPN is better than the NMI value of GN, mGN, and all PGNs. For example, in the otc1000 data, the NMI of GN-PPN from threads 2 to 16 increases from 0.704 to 0.806. Similar results are also found in the other data. In addition, the result also shows that the NMI value of GN-PPN increases with the increasing number of threads. However, we found that the NMI value in all methods is equal to 1 for alpha100 and alpha500 data. The condition means that no community was formed.

The processing time comparison is shown in Fig. 4. The results show that the processing time of the GN-PPN method is faster than other methods. However, the processing time of the GN-PPN is slower than the GN and mGN methods on otc100 and alpha100 data. For otc100 data, the GN and mGN methods require a time of 1.424 seconds and 1.711 seconds, respectively. In comparison, the GN-PPN method takes 3.572 seconds to 5.137 seconds. Furthermore, for alpha100 data, the GN and mGN methods take 1.209 seconds and 1.056 seconds, respectively. At the same time, the GN-PPN method takes 3.357 seconds to 10.552 seconds.

6. Discussion

The experiments show that the proposed method, i.e. GN-PPN, gives the best performance in terms of the processing time. The weight values of the graphs, whether positive or negative, do not influence the performance. The percentage of decreased processing time for the GN-PPN method compared to the GN method ranges from 47.443% to 90.180%, for the mGN method is between 76.093% to 89.881%, while for the PGN method, it is between38.541% to 95.427%. This decrease in time is significant for increasing the amount of data used in processing the GN-PPN method.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure. 2 The community results of GN method on data: (a) Celegens, (b) Jazz, (c) Coauthorship, and (d) Usair97 and GN-PPN method on data: (e) Celegens, (f) Jazz, (g) Coauthorship, and (h) Usair97

Table 2. The modularity results of GN, mGN, PGN, and GN-PPN using Dijkstra's algorithm

No.	Data	GN	mGN	PGN_2	PGN_4	PGN_8	PGN_16	GN-PPN_2	GN-PPN_4	GN-PPN_8	GN-PPN_16
1	Celegens	0.477	0.434	1.000	1.000	0.901	0.883	0.545	0.588	0.501	0.501
2	Jazz	0.580	0.394	0.692	0.416	0.426	0.401	0.420	0.400	0.388	0.413
3	Coauthorship	0.801	0.826	0.815	0.830	0.822	0.835	0.809	0.804	0.802	0.808
4	Usair97	0.811	0.870	0.024	0.217	0.174	0.408	0.907	0.907	0.907	0.925

Table 3. The modularity results of GN, mGN, PGN, and GN-PPN using Johnson's algorithm

No	Data	Method									
		GN	mGN	PGN_2	PGN_4	PGN_8	PGN_16	GN-PPN_2	GN-PPN_4	GN-PPN_8	GN-PPN_16
1	otc100	0.582	0.588	0.394	0.484	0.559	0.592	0.563	0.555	0.575	0.574
2	otc500	0.354	0.320	0.000	0.018	0.041	0.124	0.262	0.262	0.250	0.236
3	otc1000	0.361	0.332	0.006	0.021	0.022	0.016	0.242	0.213	0.208	0.203
4	otc2000	0.405	0.400	0.003	0.008	0.014	0.013	0.247	0.250	0.216	0.214
5	alpha100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	alpha500	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	alpha1000	0.461	0.461	0.000	0.000	0.000	0.008	0.332	0.339	0.277	0.086
8	alpha2000	0.566	0.511	0.000	0.000	0.000	0.000	0.484	0.440	0.338	0.182

Furthermore, the evaluation results of the proposed method also show significant results. Evaluation using NMI shows the results of the GN-PPN method are better when compared to other methods. Evaluation using modularity also shows modularity's value, which is not much different from other methods. Furthermore, the GN-PPN method can also detect communities similar to other methods such as GN.

However, the evaluation results of the GN-PPN method related to processing time also show a slow processing time when the GN-PPN method processes data with a small amount of 100 data lists. However, this delay in processing time does not occur if the processed data is more significant than 500. The result proves that the GN-PPN method is suitable for extensive data.

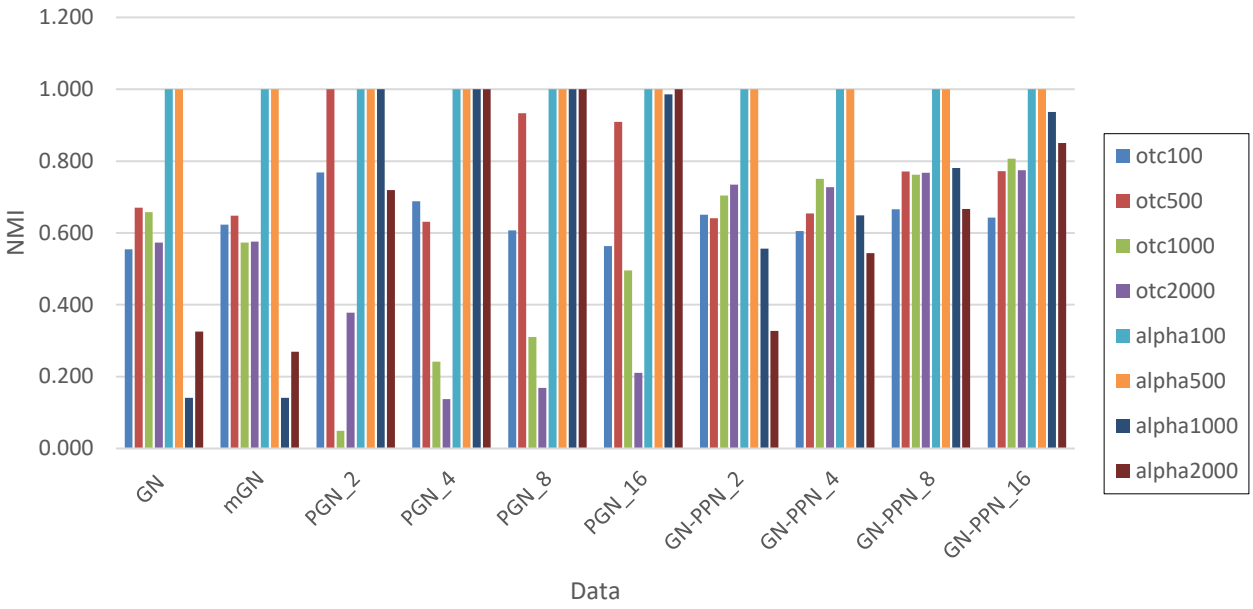


Figure. 3 The NMI of GN-PPN compared to GN, mGN, and PGN

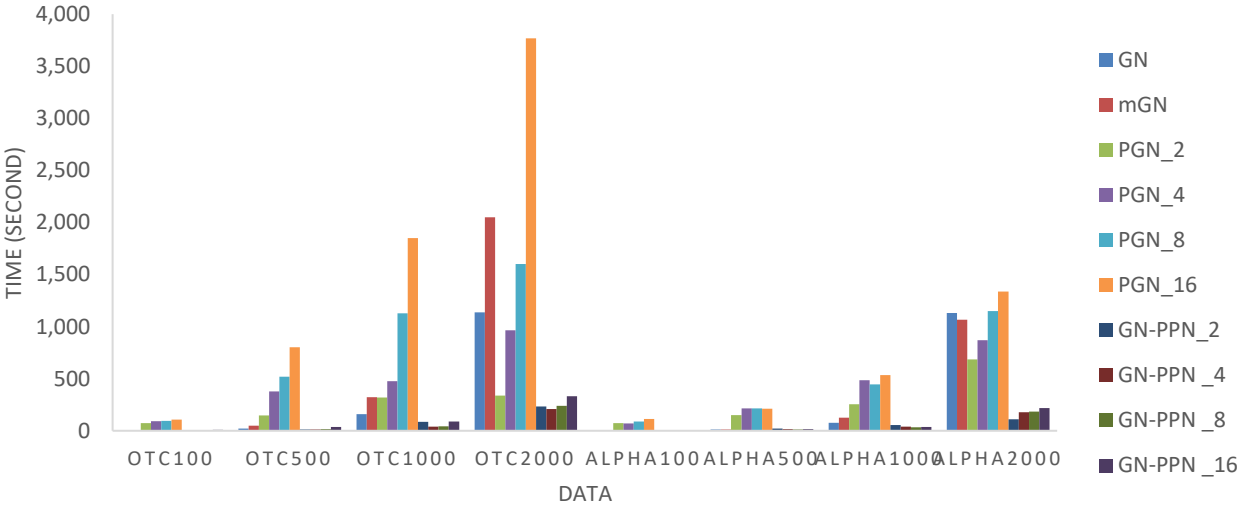


Figure. 4 The processing time of GN-PPN compared to GN, mGN, and PGN

7. Conclusion

This paper proposes the GN-PPN method for detecting community in graphs based on the GN method. The GN-PPN can process graphs with positive and negative weights and utilizes parallelization processes to speed up the processing time. This method uses edge betweenness as the GN method and combines it with pair dependencies. The experiment results on graphs with only positive or positive and negative weights show that GN-PPN outperforms other methods in terms of processing time, the accuracy of detection community and modularity. This result shows promising results of GN-PPN.

For future work, several developments can be done, primarily related to the modularity of the GN-PPN method. The evaluation results of the GN-PPN method show that this method has a modularity that is not much different from other methods such as GN, mGN and PGN. However, this method's processing time and accuracy are better than other methods. Further research needs to be done significantly to increase the modularity of the GN-PPN method. Furthermore, the density of the dataset used can be investigated to analyze further the impact of a graph's density on this GN-PPN method.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization and validation are conducted by Neny Sulistianingsih, Edi Winarko, and Anny Kartika Sari. Methodology, software, formal analysis, investigation, visualization, writing—Neny Sulistianingsih conducts original draft preparation. Neny Sulistianingsih and Edi Winarko provide resources; for writing, such as review and editing, and Edi Winarko and Anny Kartika Sari conduct supervision.

References

[1] Y. Du, J. Wang, and Q. Li, "An android malware detection approach using community structures of weighted function call graphs", *IEEE Access*, Vol. 5, pp. 475-479, 2018.

[2] D. Budic, K. Skracic, and I. Bodrušic, "Optimizing Mobile Radio Access Network Spectrum Refarming using Community Detection Algorithms", In: *Proc. of International Convention on Information and Communication Technology*, pp. 475-479, 2019.

[3] P. N. H. Pham, B. N. T. Nguyen, Q. T. N. Co, N. T. Nguyen, P. Tran, and V. Snášel, "An efficient hybrid algorithm for community structure detection in complex networks based on node influence", *ICIC Express Letters, Part B: Applications*, Vol. 12, No. 10, pp. 899-908, 2021.

[4] V. Karyotis, K. Tsitseklis, K. Sotiropoulos, and S. Papavassiliou, "Enhancing Community Detection for Big Sensor Data Clustering via Hyperbolic Network Embedding", In: *Proc. of International Conference on Pervasive Computing and Communications Workshops*, pp. 266-271, 2018.

[5] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge Weight Prediction in Weighted Signed Networks", In: *Proc. of IEEE International Conf. in Data Mining*, pp. 221-230, 2016.

[6] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. S. Subrahmanian, "REV2: Fraudulent User Prediction in Rating Platforms", In: *Proc. of ACM International Conference on Web Search and Data Mining*, pp. 333-341, 2018.

[7] M. Girvan and M. E. J. Newman, "Community Structure in Social and Biological Networks", In: *Proc. of the National Academy of Sciences of the United States of America*, Vol. 99, No. 12, pp. 7821-7826, 2002.

[8] F. Jamour, S. Skiadopoulos, and P. Kalnis, "Parallel Algorithm for Incremental Betweenness Centrality on Large Graphs", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 29, No. 3, pp. 659-672, 2018.

[9] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labelled graphs", In: *Proc. of International Conference on Machine Learning, Washington*, pp. 321-328, 2003.

[10] A. K. Yadav, R. Johari, and R. Dahiya, "Identification of Centrality Measures in Social Network using Network Science", In: *Proc. of International Conference on Computing, Communication and Intelligent Systems*, pp. 229-234, 2019.

[11] M. S. Saputri, A. Wibisono, A. Krisnadhi, A. Y. L. Yohanes, T. A. Faisal, A. W. Utama, M. A. M. Ariefa, A. Ramadhani, and A. Muda, "Browsing Behavior Analysis from Wi-Fi Logs Based on Community Detection: Case Study on Educational Institution", In: *Proc. of International Workshop on Big Data and Information Security*, pp. 87-92, 2018.

[12] A. Nita, S. Manolache, C. M. Ciocanea, and L. Rozylowicz, "Characterizing Protected Areas Management using Ego-networks", In: *Proc. of International Conference on Advances in Social Networks Analysis and Mining*, pp. 642-643, 2017.

[13] R. C. Jisha, P. S. Indrajith, and S. Abhishek, "Community Detection Using Graph Partitioning", In: *Proc. of 2nd Global Conference for Advancement in Technology*, pp. 1-6, 2021.

[14] M. Y. Daha, M. S. M. Zahid, A. Alashhab, and S. U. Hassan, "Comparative Analysis of Community Detection Methods for Link Failure Recovery in Software Defined Networks", In: *Proc. of International Conf. on Intelligent Cybernetics Technology & Applications*, pp. 157-162, 2021.

[15] R. Bocu and S. Tabirca, "Protein Communities Detection Optimization Through an Improved Parallel Newman-Girvan Algorithm", In: *Proc. of Roedunet International Conference*, pp. 380-385, 2010.

[16] S. Moon, J. Lee, and M. Kang, "Scalable Community Detection from Networks by Computing Edge Betweenness on MapReduce", In: *Proc. of International Conference on Big Data and Smart Computing*, pp. 145-148, 2014.

[17] R. Fan, K. Xu, and J. Zhao, "A GPU-based solution for fast calculation of the betweenness centrality in large weighted networks", *PeerJ Computer Science*, Vol. 3, pp. 1-23, 2017.

[18] S. Fortunato, "Community detection in graphs",

Physics Reports, Vol. 486, No. 3-5, pp. 75-174, 2010.

[19] J. Ljucovi and S. Tomovic, "Analyzing Clusters in the University of Montenegro Collaboration Network", In: *Proc. of Mediterranean Conference on Embedded Computing*, pp. 264-267, 2016.

[20] W. Li and D. Schuurmans, "Modular Community Detection in Networks", In: *Proc. of International Joint Conf. on Artificial Intelligence Modular*, pp. 1366-1371, 2011.

[21] P. Xiong, W. Ping, and H. Chen, "Comparison of Community Detection Algorithms on Contracts Networks", In: *Proc. of Chinese Control Conference*, pp. 7475-7479, 2021.

[22] D. J. Watts and S. H. Strogatz, "Collective Dynamics of Small-World Networks", *Nature*, Vol. 393, pp. 440-442, 1998.

[23] P. M. Gleiser and L. Danon, "Community Structure in Jazz", *Advance Complex Systems*, Vol. 06, No. 04, pp. 565-573, 2003.

[24] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices", *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics*, Vol. 74, No. 3, 2006.

[25] H. Zare, M. Hajiabadi, and M. Jalili, "Detection of Community Structures in Networks with Nodal Features based on Generative Probabilistic Approach", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 33, No. 7, pp. 2863-2874, 2021.