Int. j. inf. tecnol. (March 2022) 14(2):607–618
https://doi.org/10.1007/s41870-022-00873-5

ORIGINAL RESEARCH

# Assessing the impact of the density and sparsity of the network on community detection using a Gaussian mixture random partition graph generator

**Ashani Wickramasinghe**[1] · **Saman Muthukumarana**[1]

Received: 16 August 2021 / Accepted: 5 January 2022 / Published online: 27 January 2022


**Abstract** Identification of sub-networks within a network is essential to understand the functionality of a network. This process is called as 'Community detection'. There are various existing community detection algorithms, and the performance of these algorithms can be varied based on the network structure. In this paper, we introduce a novel random graph generator using a mixture of Gaussian distributions. The community sizes of the generated network depend on the given Gaussian distributions. We then develop simulation studies to understand the impact of density and sparsity of the network on community detection. We use Infomap, Label propagation, Spinglass, and Louvain algorithms to detect communities. The similarity between true communities and detected communities is evaluated using Adjusted Rand Index, Adjusted Mutual Information, and Normalized Mutual Information similarity scores. We also develop a method to generate heatmaps to compare those similarity score values. The results indicate that the Louvain algorithm has the highest capacity to detect perfect communities while Label Propagation has the lowest capacity

**Keywords** Community detection · Similarity measures · Random partition graphs generator · Mixture of Gaussian distributions

✉ Ashani Wickramasinghe
wickrama@myumanitoba.ca

Saman Muthukumarana
saman.muthukumarana@umanitoba.ca

[1] Department of Statistics, Faculty of Science, University of Manitoba, Winnipeg, MB R3T 2N2, Canada

## 1 Introduction

Community detection is an interesting area of social network analysis. It helps us to identify hidden communities within a network. Starting from the early 2000, many community detection algorithms have been introduced and studied in multidisciplinary areas. The main types of applications of community detection in networks are recommendation systems [1, 2], link predictions [3, 4], and anomaly detection in online social networks [5, 6]. However, different algorithms can generate significantly different communities based on their algorithms. In the literature, the most common way to evaluate the community detection algorithm is comparing the algorithm's result with the ground truth communities of that network [7–9]. One can also evaluate the community detection algorithms using dynamic graphs with planted evolving community structure, as a benchmark [10]. In another study [11] the performance of the Community Density Rank (CDR) algorithm against other community detection algorithms were compared using synthetic data from Lancichinetti, Fortunato and Radicchi (LFR) algorithm [12]. Dao and the team also have done a study [13] to compare community detection methods, based on computation time and community size distribution. However, these studies have not considered about the change of community detection ability with the change of features of networks. The main similarity metrics which generally use to assess the similarity are Adjusted Rank Index (ARI) [14], Normalized Mutual Information (NMI) [15] and Adjusted Mutual Information (AMI) [16].

In this paper, we focus on identifying the impact of topological features on community detection results. We conducted a comparative analysis using sparse and dense networks. Dense networks have many connections with

others, while sparse networks have fewer edges than the possible maximum number of edges. We used Gaussian Random Partition Graph [24] to generate dense networks and our newly developed method, which uses the mixture of Gaussians, to generate sparse networks. Then used those networks as the benchmarks to evaluate community detection methods. The following section will discuss both network generation methods in detail and the main difference between those two methods.

## 2 Material and methods

In social network analysis, the community is defined as a subset of nodes within the graph that have a higher probability of being connected to each other than to the rest of the network. In our previous study [17]. The communities were identified using community detection algorithms in order to understand the nature of the spread of COVID-19. Here, we consider directed networks for the simulations and use Louvain algorithm [19], Infomap algorithm [20], Label propagation algorithm [22], and Spinglass algorithm [23] for community detection.

Most of the algorithms are developed based on modularity. Because of that, understanding the meaning of modularity is essential. It measures the strength of the division of a network into groups. Modularity is defined as,

$$Q = \frac{1}{4m} \sum_{i,j \text{ in same module}} \left( A_{ij} - \frac{k_i k_j}{2m} \right). \tag{1}$$

Here $m$ is the number of edges of the graph, $k_i$ is the degree of node $i$, and $A_{ij}$ is the adjacency matrix. The above equation shows modularity for an undirected graph. [18] developed a new modularity for a directed graph, is given by,

$$Q_d = \frac{1}{m} \sum_{i,j \text{ in same module}} \left( A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \right). \tag{2}$$

When calculating directed modularity, both in-degrees and out-degrees were considered, but when calculating undirected modularity, only the degree centrality is considered. In the following paragraphs we are discussing the different methods we used for community detection.

The first algorithm that we are focusing on is, the **Louvain algorithm** [19]. This is an unsupervised algorithm for detecting communities in networks. This method has two phases; Modularity Optimization and Community Aggregation. These two phases are executing until there are no more changes in the communities, and the maximum modularity is achieved. At the Modularity Optimization phase, each node is assigned to its own community. Then node $i$ is removing from its own community and moving it into the neighbor community $j$. The change in modularity is calculated using,

$$\Delta Q_d = \frac{d_i^c}{m} - \left[ \frac{d_i^{out} \cdot \sum_{tot}^{in} + d_i^{in} \cdot \sum_{tot}^{out}}{m^2} \right]. \tag{3}$$

Here $d_i^c$ is the degree of $i$ in community $C$ and $\sum_{tot}^{in}$ shows the number of incoming edges of community C, while $\sum_{tot}^{out}$ shows number of outgoing edges of community C. If no positive increase can be seen in modularity, the node $i$ remains in its current community. This sequence will be repeatedly performed for all nodes until no increment in modularity can be seen. The 1st phase stops when a local maximum has been found.

In the community aggregation step, each communities are considered as a single node. Then a new network is built using these community nodes. When the new network is created, the second phase has ended, and the first phase can be re-applied to the new network. These phases are carried out until there is no more change in the community, and a maximum of modularity is achieved.

The **Infomap algorithm** repeats the two described phases in Louvain until an objective function is optimized [20]. Instead of modularity, this algorithm optimizes the **'Map equation'**. Infomap algorithm finds community structure by minimizing the description length of a random walker's movements on a network. This random walker randomly moves between each node of the network. The random walker would like to move through the highly weighted edges. Hence, the weights of the connections within the community are greater than the weights of the connections between nodes of different communities.

The definition of the map equation is based on Shannon's Source Coding Theorem, from the field of Information Theory [21]. Here each module has a **'module codebook'** and these module codebooks have **'codewords'** for the nodes within each module, which are derived from the node visit/exit frequencies of the random walker. The **'index codebook'** has codewords for the modules, which are derived from the module switch rates of the random walker. The map equation shows that the average length of the code (a step of the random walker) is equal to the average length of codewords from the index codebook and the module codebooks weighted by their rates of use.

$$L(M) = q_{\curvearrowright} H(Q) + \sum_{i=1}^{m} p_{i\circlearrowright} H(\rho_i) \tag{4}$$

A network with $n$ nodes and $m$ clusters is shown by $M$, and $L(M)$ shows the per-step description length for module

partition of that. Note that $q_{\curvearrowleft}$ and $p_i\circlearrowleft$ show the rate of use of index codebook and the rate of use of module codebook respectively, while the $H(Q)$ and $H(\rho_i)$ show the frequency-weighted average length of codewords in the index codebook and module codebook i respectively.

The next algorithm is **Label Propagation (LPA)** [22] and it is one of the fast semi-supervised algorithms for finding communities in a graph. The algorithm works as follows; Every node is initialized with a unique community label (an identifier), and these labels propagate through the network. At the propagation step each node update the label based on the labels of their neighbor's. When there are ties, labels are selected uniformly and randomly. This algorithm converges when each node has the majority label of its neighbors. And this stops if either convergence or the use defined maximum number of iterations is achieved.

In LPA, the influence of a node's label on other nodes is determined by their respective closeness and the closeness between nodes is measured by (5). Here the euclidean distance between node i and j is shown by $d_{ij}$ and $\sigma^2$ shows the parameter to scale proximity. $w_{ij}$s are used to generate weight matrix and to perform label propagation weight matrix is converted into a transition matrix T using the $t_{ij}$s by (6).

$$w_{ij} = exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \tag{5}$$

$$t_{ij} = \frac{w_{ij}}{\sum_k w_{kj}}. \tag{6}$$

**Spinglass** is the last algorithm that we are discussing in this study. This method minimizes the Hamiltonian of the network [23]. Spinglass has the following four requirements, and they are considered to develop communities.

1. reward internal edges between nodes of the same group (in the same spin state)
2. penalize missing edges between nodes in the same group.
3. penalize existing edges between different groups (nodes in different spin state)
4. reward non-links between different groups.

Hamiltonian for spinglass is given by the following equation. Here $A_{ij}$ is adjacency matrix of the graph. $\delta(\sigma_i, \sigma_j)$ is known as Kronecker delta function, $\delta_{\sigma_i,\sigma_j} = 1$ if $\sigma_i = \sigma_j$, and 0 otherwise. The spin state (or group index) of the node i is showed by $\sigma_i \epsilon \{1, 2, ..., q\}$. $a_{i,j}, b_{i,j}, b_{i,j}, d_{i,j}$ are the weights of the individual contributions for the above requirements, respectively.

$$\begin{aligned} H(\sigma) = &-\sum_{i\neq j} a_{ij}A_{ij}\delta(\sigma_i,\sigma_j) + \sum_{i\neq j} b_{ij}(1-A_{ij})\delta(\sigma_i,\sigma_j) \\ &+ \sum_{i\neq j} b_{ij}(1-A_{ij})\delta(\sigma_i,\sigma_j) \\ &- \sum_{i\neq j} d_{ij}(1-A_{ij})(1-\delta(\sigma_i,\sigma_j)) \end{aligned} \tag{7}$$

This algorithm tries to minimizes the energy of spinglass with the spin states being the community indices. Here Modularity is rewritten using Hamiltonian as (8),

$$Q = -\frac{1}{M}H(\{\sigma\}). \tag{8}$$

It applies the simulated annealing optimization technique on this model to optimize the modularity.

### 2.1 Random graph generation

We now discuss two methods to generate random directed graphs with random partitions. Those methods are the Gaussian random partition graph [24] and our newly developed method. The main difference between these two methods is that the Gaussian Random Partition graph draws community size from a normal distribution. In contrast, our newly developed method draws community size from a mixture of Gaussian distributions. Since the existing method uses normal distribution, most of the communities will have a similar number of nodes. But in real life we usually see communities with different number of nodes. Hence we proposed this new method, which generates communities in different sizes.

A Gaussian random partition graph [24] is created by generating k clusters, each with a size drawn from a normal distribution with mean (s) and standard deviation (s/v). The size of the last cluster is possibly significantly smaller than the others. Here v is the shape parameter. Nodes are connected within clusters with probability $P_{in}$ and between clusters with probability $P_{out}$. To induce a more general structure of a social network, we have updated the algorithm of the Gaussian random partition graph with a mixture of Gaussian distributions. Here we can use two normal distributions; with a larger mean and a lower mean value. As a result, this method will generate network graphs with both larger and smaller communities.

In this novel method, for a given number of nodes ($n$), cluster sizes will be drawn from a mixture of two Gaussian distributions with means $s1$ and $s2$, and standard deviations ($s1/v1$) and ($s2/v2$) respectively. Let the probability density function (PDF) for the $i^{th}$ Gaussian distribution is $f_i(x)$. In the mixture of Gaussians, the probability of drawing from the first distribution is $p$ and from the second distribution is

$(1-p)$, where $p$ is known as the mixing parameter. Then the probability density function of the mixture of Gaussians can be written as below:

$$\begin{aligned} f_M(x) &= (p)N\left(s_1, \frac{s_1}{v_1}\right) + (1-p)N\left(s_2, \frac{s_2}{v_2}\right) \\ &= (p).f_1(x) + (1-p).f_2(x). \end{aligned} \tag{9}$$

The mean of Gaussian mixture can be observed by integrating:

$$\begin{aligned} \mu_M &= \int_{-\infty}^{\infty} x f_M(x)dx \\ &= \int_{-\infty}^{\infty} xp.f_1(x)dx + \int_{-\infty}^{\infty} x(1-p).f_2(x))dx \\ &= p\int_{-\infty}^{\infty} xf_1(x)dx + (1-p)\int_{-\infty}^{\infty} xf_2(x))dx \\ &= p.\mu_1 + (1-p)\mu_2. \end{aligned} \tag{10}$$

In order to find the variance, first of all we need to find the second moment. Then using that we can find the variance of Gaussian mixture.

$$\begin{aligned} E[M^2] &= \int_{-\infty}^{\infty} x^2 f_M(x)dx \\ &= \int_{-\infty}^{\infty} x^2 p.f_1(x)dx + \int_{-\infty}^{\infty} x^2(1-p).f_2(x))dx \\ &= p\int_{-\infty}^{\infty} x^2 f_1(x)dx + (1-p)\int_{-\infty}^{\infty} x^2 f_2(x))dx \\ &= p.E(x_1^2) + (1-p).E(x_2^2) \end{aligned} \tag{11}$$

$$\sigma_M^2 = E[M^2] - \mu_M^2 \tag{12}$$

Hence using above equations we can calculate mean and standard deviation values of mixture of two distributions and generate networks with partitions.

### 2.2 Simulation study

In this section, we describe our simulation process, where we consider various parameter values. The following flow chart in Fig. 1 shows the process step by step. That illustrates the simulation process when changing the number of nodes while all the other parameters are fixed. Based on this flow chart, first, we are generating a network with the number of nodes equal to 10. Next, this network will be developed with community labels for each node, and we consider those labels as actual labels. Then we use the same generated network to find communities by applying our four community detection algorithms (Louvain, Label propagation, Spinglass, and Infomap). After that, we can evaluate community results based on actual labels using similarity matrices (ARI, NMI, and AMI). In this simulation study, we generated all the synthetic networks using the 'Networkx' package in python [25]. To calculate similarity scores, we used the scikit-learn package in python [26].

We developed another method to change two parameters and understand the hidden truth of the results of different community detection algorithms. This process explains the steps to create a heat map for each community detection algorithm, which shows the variation of similarity score measure, with the change of $P_{in}$ and $P_{out}$. Here rows and columns will indicate $P_{in}$ and $P_{out}$.

Figure 2 explains the process step by step. As the first step, we fixed number of nodes (n), two mean values (s1, s2), two standard deviance ($\sigma_1$, $\sigma_2$) values and changed $P_{in}$ and $P_{out}$ values from 0.01 to 0.5 with 0.01 increment. Then we generated a network using a random partition graph with a mixture of two Gaussians and considered those partitions as true communities. Next, we applied the community detection algorithm to the network and identified the communities. Then we stored the similarity scores between true and identified communities in a vector. We repeated the same procedure 100 times and calculated the mean of those similarity values to get the average similarity score for the given $P_{in}$ and $P_{out}$. This average similarity score will be stored in the first cell of the matrix. Then repeat the same procedure for different $P_{in}$ and $P_{out}$ values and fill the matrix.

After creating the similarity score matrix, we could easily convert it to a color heat map. Heat maps appeal to the eyes, and visualization is generally easier to understand than reading the values. Through the heat maps, we could easily identify the capacity of the given algorithm to identify correct communities.

## 3 Results

As we discussed in the introduction, we have considered two main random network graph generators for this study. Figure 3 shows four random network graphs which were generated from Gaussian random network graphs. We can see that most of the networks have communities of similar size. But in reality, we would not always be able to find this kind of communities. For example, there can be communities with a large number of nodes while some other communities with few nodes. Hence, to generate more practical networks, we developed our new network graph generator using the mixture of Gaussians.

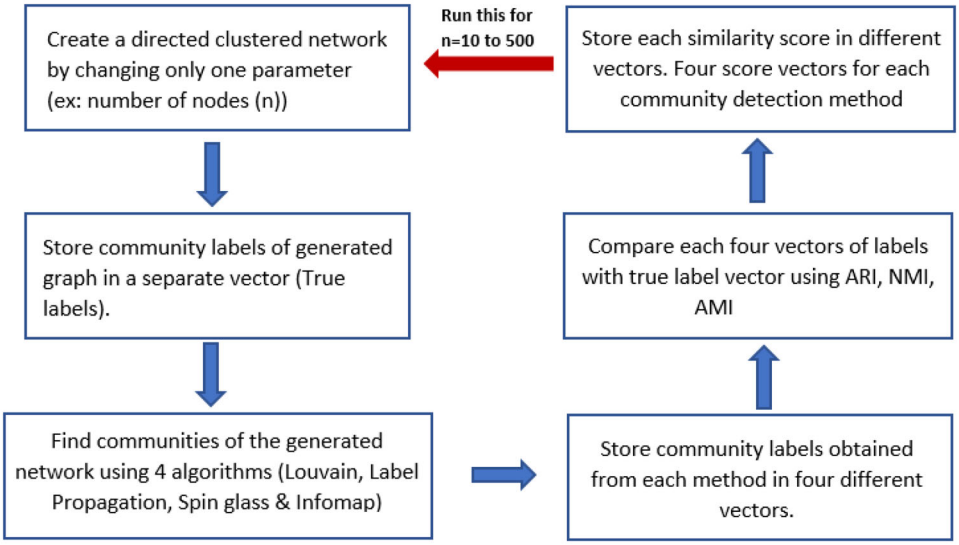


**Fig. 1** General simulation process

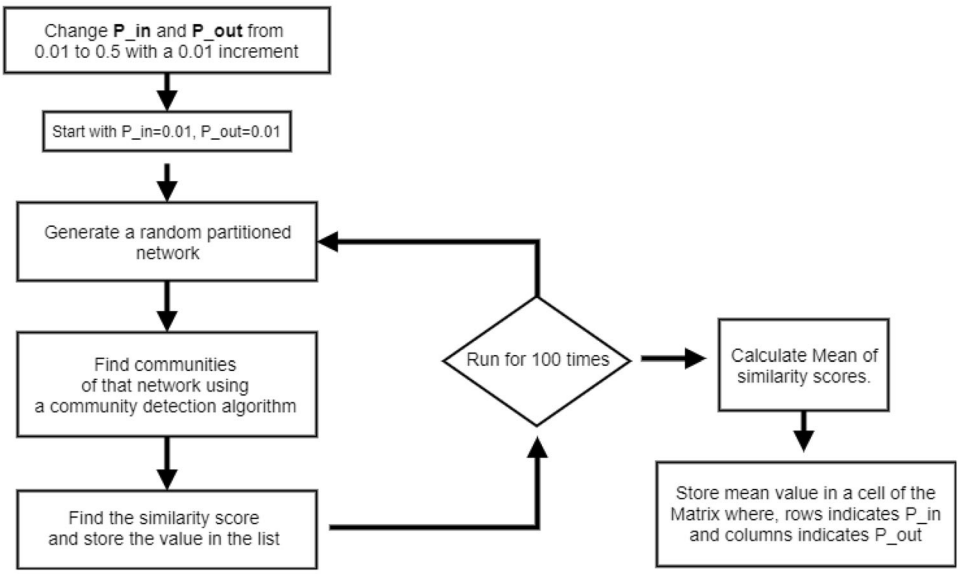


**Fig. 2** The process of creating a similarity score value matrix

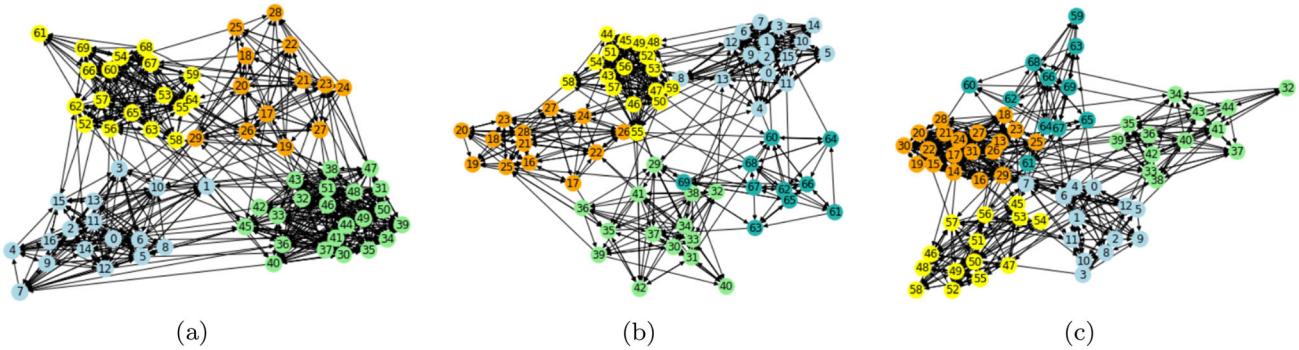


**Fig. 3** Plots **a**–**c**: Some randomly generated graphs from Gaussian random partition graph. All graphs were generated using same parameters

In Fig. 4 we can see sample network graphs that were generated from our new method. In these networks, we can observe that, some communities with a large number of nodes and others with a small number of nodes.

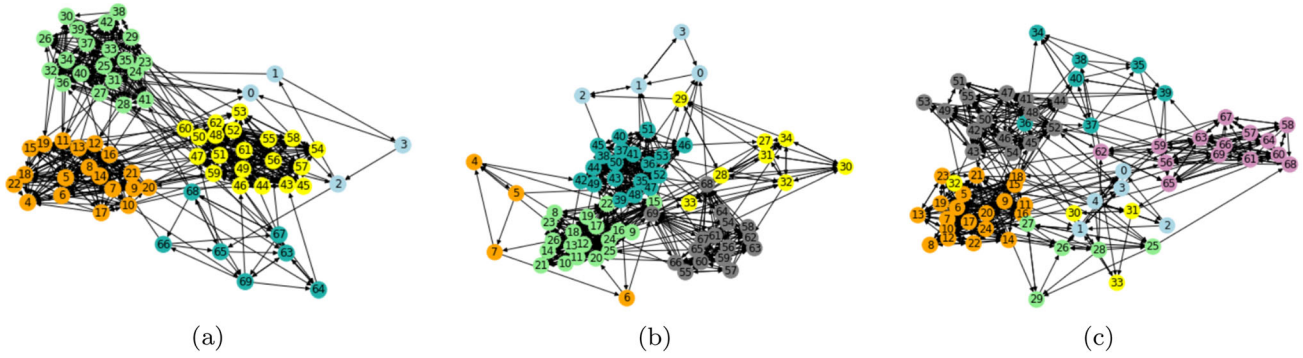

**Fig. 4** Plots **a**–**c**: Some randomly generated graphs from newly developed method using mixture of two Gaussians. All graphs were generated using same parameters

### 3.1 Results of dense networks

In this simulation, we fixed all other parameters except the parameter that we are interested in when creating networks. We considered the number of nodes, probabilities of connecting between communities (intra probability) and within communities (inter probability) as the parameters. Figure 5 shows similarity score variations with the increment of the number of nodes. Based on these plots, we can see that ARI, NMI, and AMI scores of communities based on Louvain and Spinglass methods decrease when the number of nodes increases. On the other hand, ARI, NMI, and AMI scores for Infomap and Label-propagation methods are constant with zero value. Thus, it shows that the number of nodes does not affect the similarity score of those two methods.

Figure 6 shows similarity score variations with the increment of intra probability. Here we have changed that probability from 0.1 to 0.9 with a 0.01 increment. Based on these figures, it is clear that communities found from Louvain and Spinglass methods show better agreement with the actual labels for the networks which are created with an intra probability greater than 0.3.

The variation of similarity score with the increment of inter probability is plotted in Fig. 7. Same as the intra-cluster probability, we used probabilities from 0.1 to 0.9 with 0.01 increment. All the above similarity score plots illustrate that Louvain and Spinglass algorithms have higher similarity scores with true clusters when the networks have lower inter-connection probability. The highest similarity score is achieved when the inter probability is 0.1. After that score drastically drops down and remains around zero. The results of Infomap and label propagation

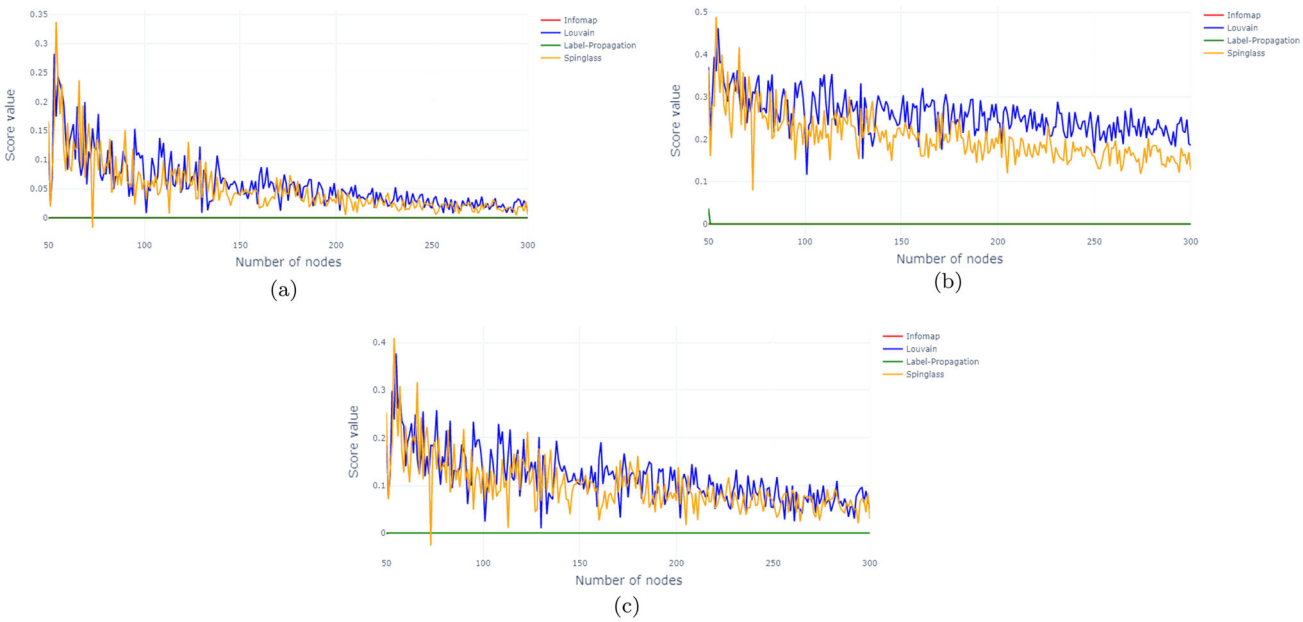


**Fig. 5** Variation of similarity scores with the increment of number of nodes. Plots **a** ARI score, **b** NMI score, **c** AMI score

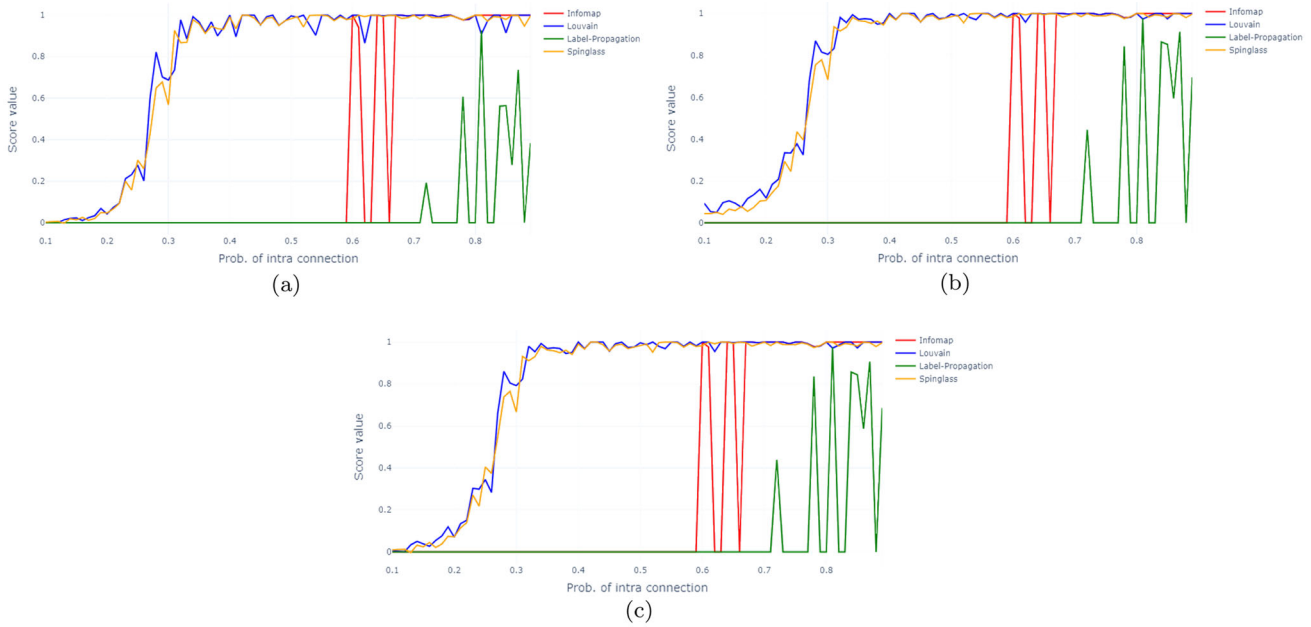


**Fig. 6** Variation of similarity scores with the increment of number of intra probability. Plots **a** ARI score, **b** NMI score, **c** AMI score

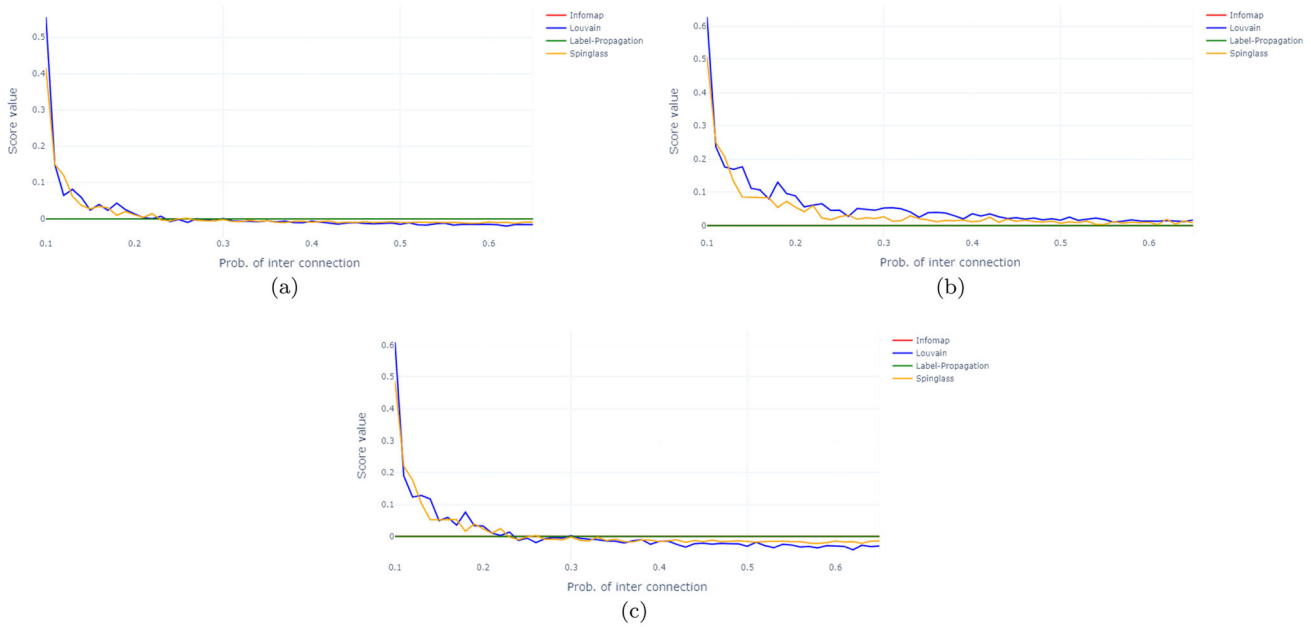


**Fig. 7** Variation of similarity scores with the increment of number of inter probability. Plots **a** ARI score, **b** NMI score, **c** AMI score

algorithms always show zero similarity scores for all three similarity scores.

### 3.2 Results of sparse networks

We now consider sparse networks using our new network generation method. We conducted the same simulation process for the sparse networks also. The behavior of the similarity scores for the results of each community detection algorithms, when increasing the number of nodes, illustrate in Fig. 8. When generating random sparse networks, some networks end up with isolated nodes. Spinglass network does not work on networks that have isolated nodes. Hence we removed Spinglass method from this simulation study.

Based on the results shown in Fig. 8, it is clear that in the sparse networks when the sample size is small, the results of all three algorithms show a higher similarity with

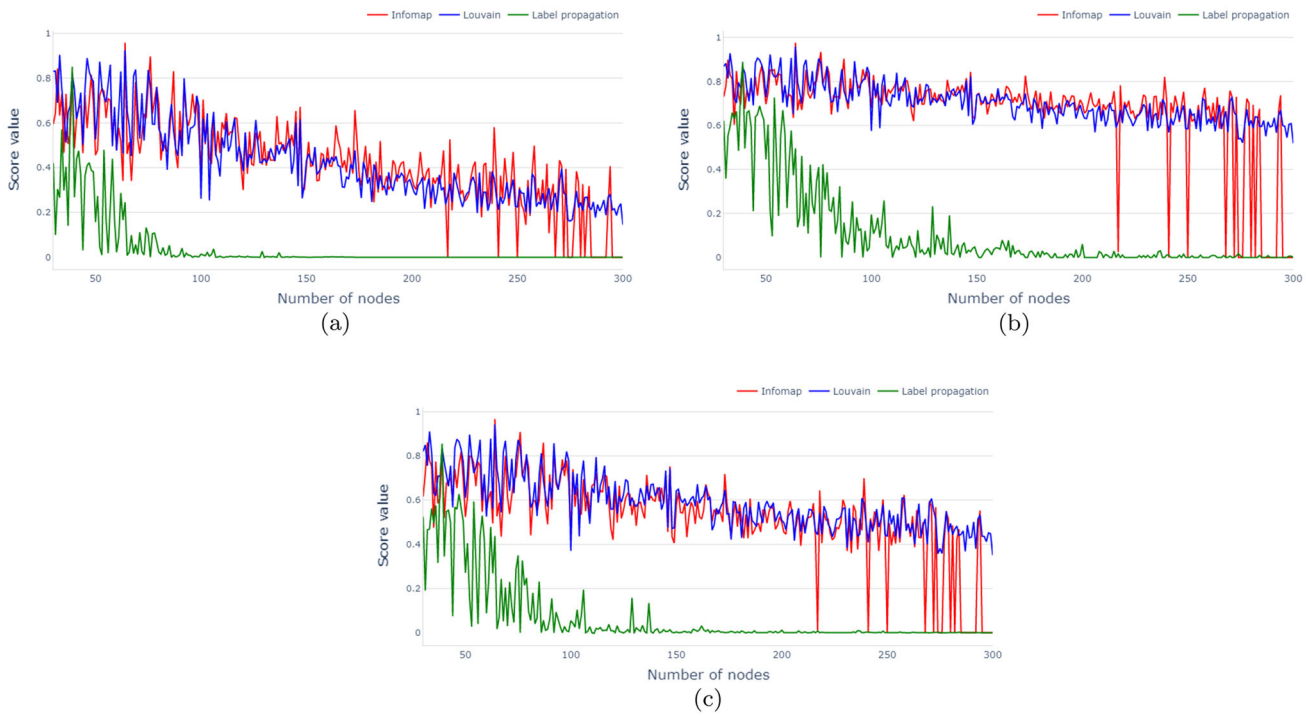


**Fig. 8** Variation of similarity scores with the increment of number of nodes in sparse networks. Plots **a** ARI score, **b** NMI score, **c** AMI score

true labels. Label propagation shows a rapid decline, while Infomap and Louvain show a gradual decline with the increase of frequency of nodes.

Then we changed intra probability from 0.1 to 0.9 with a 0.01 increment. We fixed inter probability to 0.02 and the number of nodes to 50. Figures 9 shows the results of this simulation. All three algorithms show a gradual increase of

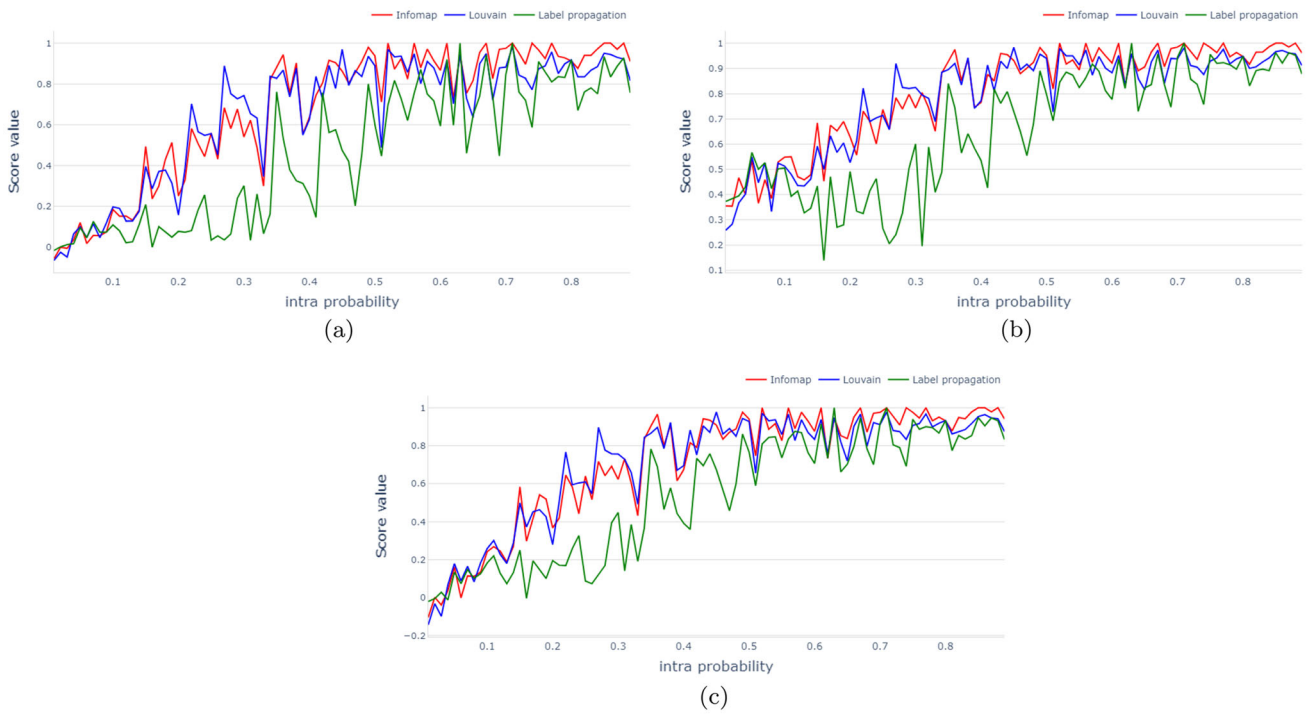


**Fig. 9** Variation of similarity scores with the increment of intra probability in sparse networks. Plots **a** ARI score, **b** NMI score, **c** AMI score

similarity score in these sparse networks with the rise of intra probability. Here the results of Louvain and Infomap show better and similar results than the Label propagation algorithm.

Finally, we changed the inter-probability. We fixed the intra probability to 0.3 and the number of nodes to 50. The variation of similarity scores (ARI, NMI, and AMI) for the results of each community detection algorithm was plotted in Fig. 10. When the inter probability is lower than 0.1, we can see higher similarity scores for all the community detection algorithms.

### 3.3 Results of heat maps

According to the steps we described in Sect. 2.2 we generated heat maps for each community detection algorithm. Since we are considering three similarity measures, ARI, NMI, and AMI, we got three heat maps for each algorithm.

Figure 11 shows ARI similarity score heat maps for the results of Infomap, Label propagation, and Louvain algorithms. The color range from dark blue to yellow shows the lowest similarity score to the highest similarity score. In these heat maps, left bottom corners have higher intra probabilities and lower inter probabilities. These left corners in yellow, indicate a high similarity score. Hence, it is clear that the results of all the community detection algorithms are acceptable if their community networks have high intra probabilities and low inter probabilities. When comparing these three heat maps, we can observe that plot (b) has the smallest yellow color area, and plot (c) has the largest yellow color area. It shows that Louvain has the highest capacity to detect acceptable communities while Label propagation has the lowest, based on different network structures.

Figures 12 and 13 show NMI and AMI similarity score heat maps, respectively. We can see yellow color areas (high similarity scores) for higher intra probabilities and lower inter probabilities in those heat maps as well. These sets of heat maps also show that the Louvain algorithm has the highest capacity to detect communities while Label propagation has the lowest capacity.

## 4 Conclusion and future direction

Since the community detection is specially developed for social network analysis which depends on edges, it is clear that the results should depend on the network structure. Our simulation study showed that most community detection algorithms gave better results for sparse networks than dense networks. However, even the community detection algorithms struggling to detect communities in dense networks could detect communities up to some level in sparse networks. In both sparse and dense networks, we observed that the ability to detect communities of algorithms increased with the increment of the probability of having edges between nodes in the same community and the increment of average community size. Conversely, the

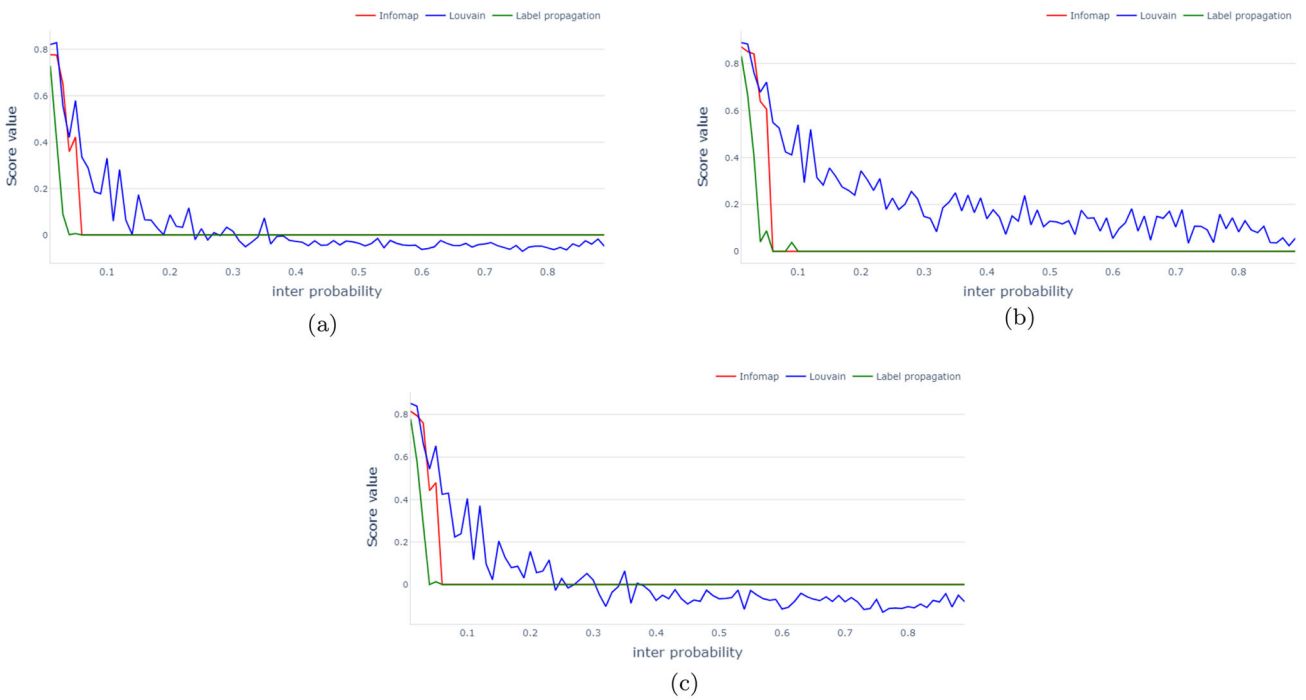


**Fig. 10** Variation of similarity scores with the increment of inter probability in sparse networks. Plots **a** ARI score, **b** NMI score, **c** AMI score

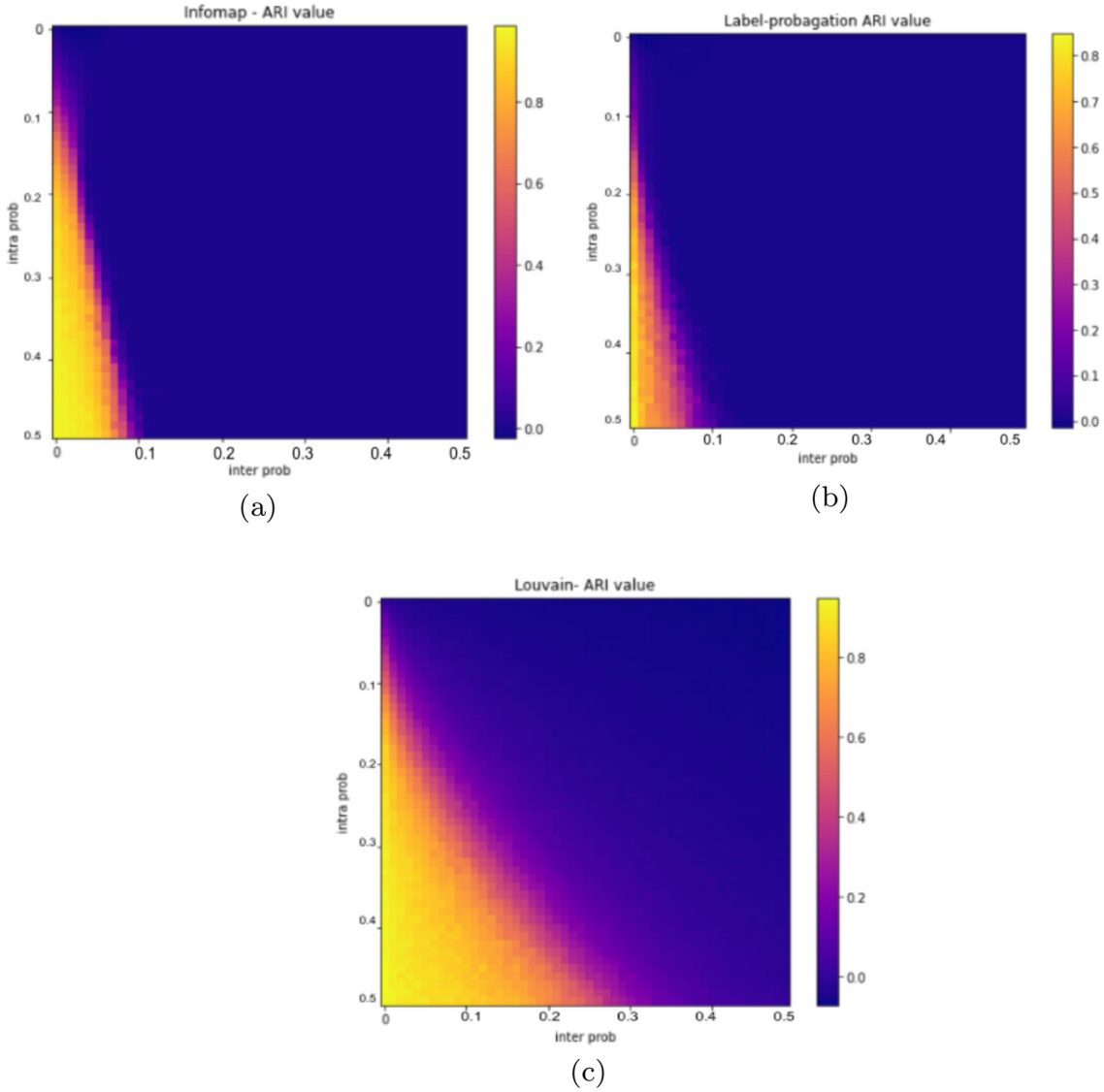


**Fig. 11** ARI similarity score heat maps for **a** Infomap, **b** Label propagation, **c** Louvain. From dark blue color to yellow color indicates low to high similarity score

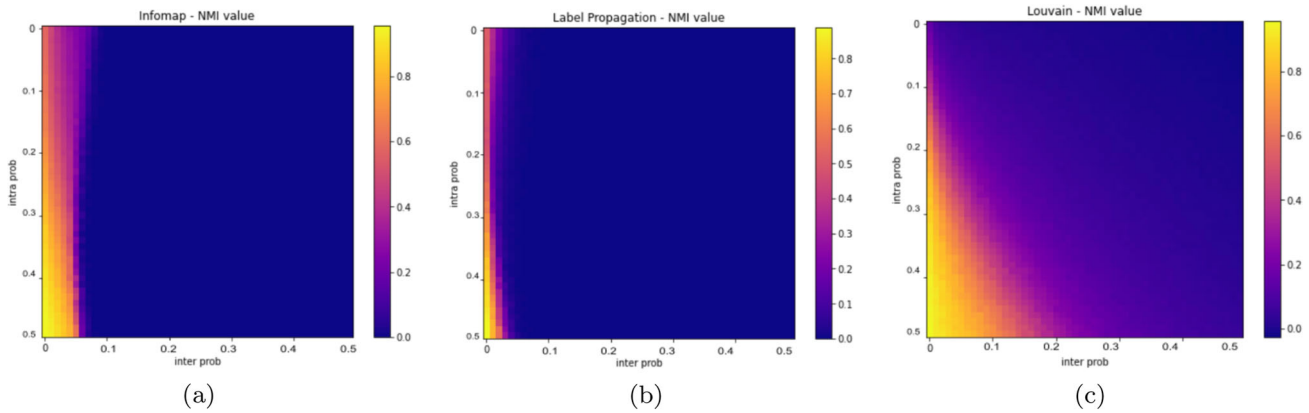


**Fig. 12** NMI similarity score heat maps for **a** Infomap, **b** Label propagation **c** Louvain. From dark blue color to yellow color indicates low to high similarity score

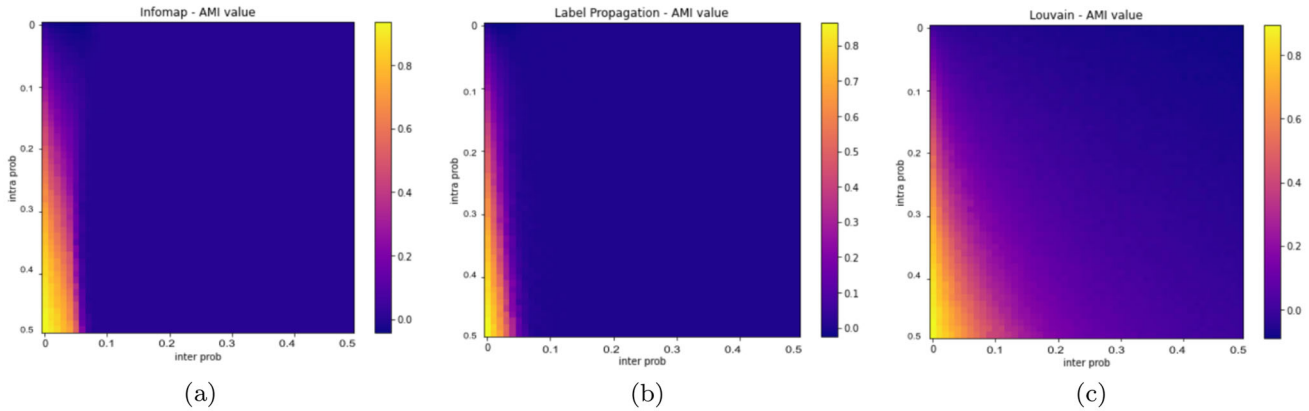


**Fig. 13** AMI similarity score heat maps for **a** Infomap, **b** Label propagation, **c** Louvain. From dark blue color to yellow color indicates low to high similarity score

detection ability decreased with the increment of the probability of having edges between nodes in different communities and the number of nodes.

Using the heat maps we generated, we could understand the community detection capacities of different community detection algorithms. Overall, the results of all the community detection methods are acceptable if the community network has higher intra probabilities and lower inter probabilities. Still, the Louvain algorithm has the highest capacity, while label propagation has the smallest. Throughout the study, we observed that the Louvain algorithm could detect communities better than other algorithms.

The Gaussian random partition graph generator is one of the best methods to generate random networks with partitions. It uses Gaussian distribution to draw the community size, and because of that, the output networks contain communities with a similar number of nodes. But in real networks, we see communities with a different number of nodes; large communities and small communities. Hence we developed this novel method to generate networks using a mixture of Gaussians and observed that it could generate networks similar to the structure of real-world disease transmission networks.

This study can be further expanded by updating the network generator using a Dirichlet process with Gaussian distribution as the base distribution. Then we can compare the network structures of the networks generated from the existing Gaussian random partition graph with the networks generated from the newly updated method with the Dirichlet process. Furthermore, we can improve our approach to evaluate temporal community detection in real-world applications.

**Acknowledgements** The authors thank the editor and anonymous reviewers whose comments/suggestions helped improve this manuscript.

**Availability of data and material** Data used in this paper is availableupon request.

**Code availability** Code used in this paper is available upon request.

**Declaration**

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethics approval** Not applicable.

**Consent to participate** Not applicable.

**Consent for publication** Not applicable.

## References

1. Zanin M, Cano P, Buldu JM, Celma O (2008) Complex networks in recommendation systems. In: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications. Stevens Point, Wisconsin, USA, pp 120–124
2. Joydeep D, Partha M, Subhashis M, Prosenjit G (2014) Clustering-based recommender system using principles of voting theory. In: Proceedings of 2014 International Conference on Contemporary Computing and Informatics, IC3I 2014, pp 230–235. https://doi.org/10.1109/IC3I.2014.7019655
3. Hao J, Zhenjie L, Chunlong L, Yansen S, Xingyi Z (2020) Community detection in complex networks with an ambiguous structure using central node based link prediction. Knowl-Based Syst. https://doi.org/10.1016/j.knosys.2020.105626
4. Tan F, Xia Y, Zhu B (2014) Link prediction in complex networks: a mutual information perspective. PLOS One. https://doi.org/10.1371/journal.pone.0107056
5. Savage D, Zhang X et al (2014) Anomaly detection in online social networks. Soc Netw. https://doi.org/10.1016/j.socnet.2014.05.002
6. Manjunatha HC, Mohanasundaram R (2019) BMADSN: Big data multi-community anomaly detection in social networks. Int J Elect Eng Educ. https://doi.org/10.1177/0020720919891065
7. Jebabli M, Cherifi H, Cherifi C, Hamouda A (2018) Community detection algorithm evaluation with ground-truth data. Physica A. https://doi.org/10.1016/j.physa.2017.10.018

8. Rossetti G, Pappalardo L, Rinzivillo S (2016) A novel approach to evaluate community detection algorithms on ground truth. In: Complex Networks VII, studies in computational intelligence, Springer, Cham, pp 133–144. https://doi.org/10.1007/978-3-319-30569-1_10
9. Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst. https://doi.org/10.1007/s10115-013-0693-z
10. Rémy C, Souâad B, Giulio R (2020) Evaluating community detection algorithms for progressively evolving graphs. J Complex Netw. https://doi.org/10.1093/comnet/cnaa027 (**Oxford University Press**)
11. George R, Shujaee K, Kerwat M, Felfli Z, Gelenbe D, Ukuwu K (2020) A comparative evaluation of community detection algorithms in social networks. Procedia Comput Sci. https://doi.org/10.1016/j.procs.2020.04.124
12. Zineb F, Rov G, Khalil S, Mohamed K (2018) Computing ranking and dynamics in social networks. In: 2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS) 59-63. https://doi.org/10.1109/SNAMS.2018.8554850
13. Dao V, Bothorel C, Lenca P (2020) Community structure: a comparative evaluation of community detection methods. Netw Sci. https://doi.org/10.1017/nws.2019.59 (**Cambridge University Press (CUP)**)
14. William MR (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc. https://doi.org/10.2307/2284239
15. Zhang P (2015) Evaluating accuracy of community detection using the relative normalized mutual information. J Stat Mech. https://doi.org/10.1088/1742-5468/2015/11/P11006
16. Vinh NX, Epps J and Bailey J (2010) Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854
17. Wickramasinghe AN, Muthukumarana S (2021) Social network analysis and community detection on spread of COVID-19. Model Assist Stat Appl. https://doi.org/10.3233/MAS-210513
18. Nicolas D, Anthony P (2015) Directed Louvain: maximizing modularityin directed networks. Research report, Universitée d'Orl éeans
19. Blondel V, Guillaume JL et al (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp. https://doi.org/10.1088/1742-5468/2008/10/P10008
20. Fang H, Liu Y (2015) A novel algorithm infomap-sa of detecting communities in complex networks. J Commun. https://doi.org/10.12720/jcm.10.7.503-511
21. Rosvall M, Axelsson D, Bergstrom CT (2009) The map equation. Eur Phys J Special Topics. https://doi.org/10.1140/epjst/e2010-01179-1
22. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev. https://doi.org/10.1103/PhysRevE.76.036106
23. Reichardt J, Bornholdt S (2006) Statistical mechanics of community detection. Phys Rev E. https://doi.org/10.1103/PhysRevE.74.016110
24. Brandes U, Gaertler M, Wagner D (2003) ESA 2003. LNCS 2832. Springer, Berlin, pp 568–57
25. Hagberg A, Swart S, Chult DS (2008) Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference. Pasadena, CA USA, pp 11–15
26. Pedregosa F, Varoquaux G, Gramfort A, et al (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830