Scientific Research Publishing

Open Journal of Modelling and Simulation, 2020, 8, 18-34
https://www.scirp.org/journal/ojmsi
ISSN Online: 2327-4026
ISSN Print: 2327-4018

A Proposal for a Benchmark Generator of Weakly Connected Directed Graphs

José Miguel Montañana[1], Antonio Hervás[2], Pedro Pablo Soriano[3]

[1]High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Nobelstraße, Stuttgart, Germany
[2]Instituto Universitario de Matemática Multidisciplinar, Universitat Politècnica de València, Valencia, Spain
[3]SEPyC, Universitat Politècnica de València, Valencia, Spain
Email: montanana@hlrs.de, ahervas@mat.upv.es, psoriano@upvnet.upv.es

How to cite this paper: Montañana, J.M., Hervás, A. and Soriano, P.P. (2020) A Proposal for a Benchmark Generator of Weakly Connected Directed Graphs. *Open Journal of Modelling and Simulation*, **8**, 18-34.
https://doi.org/10.4236/ojmsi.2020.81002

Received: October 21, 2019
Accepted: December 20, 2019
Published: December 23, 2019

Abstract

The previous studies on detection of communities on complex networks were focused on nondirected graphs, such as Neural Networks, social networks, social interrelations, the contagion of diseases, and bibliographies. However, there are also other problems whose modeling entails obtaining a weakly connected directed graph such as the student access to the university, the public transport networks, or trophic chains. Those cases deserve particularized study with an analysis and the resolution adjusted to them. Additionally, this is a challenge, since the existing algorithms in most of the cases were originally designed for non-directed graphs or symmetrical and regular graphs. Our proposal is a Benchmark Generator of Weakly Connected Directed Graphs whose properties can be defined by the end-users according to their necessities. The source code of the generators described in this article is available in GitHub under the GNU license.

Keywords

Graphs and Networks Applications, Clustering, Cluster Analysis, Complex Networks, Social Models, Higher Education Management

1. Introduction

The interaction between the elements in many complex real-world systems can be modeled as graphs or networks, where the elements are represented as vertices and the relationships between them as edges. The networks are referred to Direct Networks where the relationship between the vertices of the network can be unidirectional, while in another case, they are referred to Symmetric Networks. The weight of each edge, if it is defined, is an associated numerical

attribute.

Additionally, the networks are referred to dense networks when the elements are highly related to them and there are many more edges than vertices. In the opposite case, which is not less common, networks are referred to dispersed networks when the number of edges is much smaller than they could possibly have. The combination of these possible definitions gives rise to a wide variety of networks, in general, large and complex, whose study requires simplified models. The use of models allows for simulating the behavior of networks and studying their structure and properties [1] [2] [3] [4].

We can consider that the study of these networks requires two fundamental elements. First, the algorithms that classify vertices into different groups and identify their main properties, and secondly, sets of graphs are needed for applying and testing those algorithms.

These algorithms search on the graphs for the existence of certain sets of highly connected vertices between them and weakly connected with the others, these sets are referred to communities. Vertices grouped in the same community have common characteristics and play a certain common role within the network. The development of those algorithms and study techniques has been a constantly evolving area of work [1] [2] [4]-[9]. **Figure 1** shows the results of

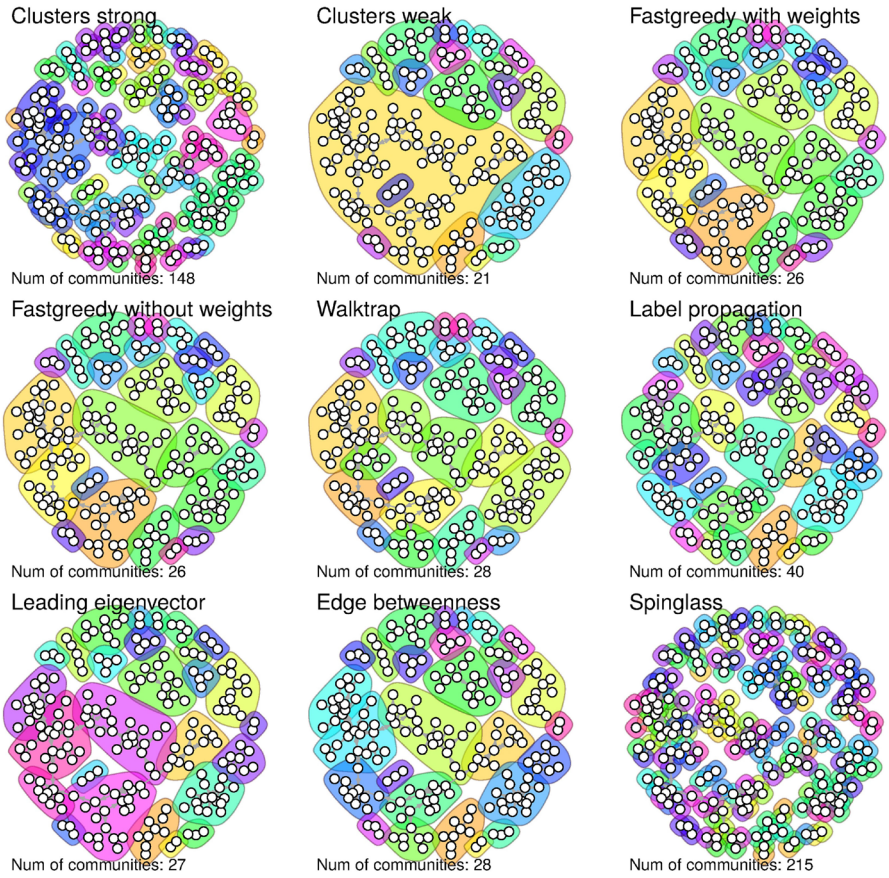


Figure 1. Example of detection of communities when using different algorithms on the same graph.

detection after applying different algorithms in the same graph.

The availability of different graphs to be analyzed by the different algorithms is important, because the quality of the algorithms is strongly related to the graph properties to which they are applied [10]. For such purpose, generators of synthetic graphs with different properties were proposed for its use on the benchmarking of the capacities of the detection algorithms [5] [11] [12].

In our work, we require detection algorithms for Weakly Connected Directed Graphs, which are common in real problems. In those graphs, the relationship between each pair of vertices can be unidirectional, or bidirectional with a different contribution in each direction. Examples of these graphs can be found in the representation of bus lines on their city map [13] [14], flights between airports [15], social networks [16], and the demand for enrolment of students in different degrees [17].

However, we found that the existing community detection algorithms are not able to correctly detect the communities in the Weakly Connected Directed Graphs. And in addition, we neither found a generator of synthetic graphs with such properties.

There are generators that, in general, work excellently for non directed graphs. [4] [5] [12] [13] However, they cannot generate such specific directed graphs as the "students' choice of university degrees" case. In particular, the results of the best approximation with the main synthetic generator used for benchmarking, to the best of our knowledge [12], are available at the same GitHub where the source code of the generators described in this article are available.

Therefore, we consider that the research on these graphs requires a generator of synthetic Weakly Connected Directed Graphs, which can be used later for developing and benchmarking new detection algorithms.

For this reason, in this article, we propose a new directed graphs generator with new modelling capabilities, which be able to modelate weakly connected directed graphs. In particular, we consider the Students' Enrolment Demand graph, which is referred to as SED-graph, to evaluate the modelling capabilities of the new generator. The extension to other models is done naturally.

2. Related Work

The benchmarking process consists of 3 steps. At the first step, it is used a synthetic generator for obtaining a set of Initial Sub-Graphs. At the second step, we apply the different detection algorithms under comparison on each Initial Graph (IG) where initially the communities are disjointed, *i.e.* there are not interconnected edges between communities, and reapplying those detection algorithms after adding new edges between communities until the amount of addition is the same amount of edges as in the IG. The percentage of the number of additional edges over the number of edges in the IG is commonly referred as a mixing parameter $0 \leq \mu \leq 1$. Notice that the additional edges are defined also by the synthetic generator.

At the last step, there are identified the detection capabilities of each algorithm, which corresponds to the highest value of mixing parameter μ for each algorithm where they are still able to detect the original communities. Typically, the coefficient Normalized Mutual Information (NMI) is used to evaluate the goodness of the detection of the original communities. The NMI is evaluated by comparing the detected communities for the different values of the Mixing Parameter μ, with respect to the initial graph (μ = 0)[1]. The value of NMI equal to 1, corresponds to a detection of communities exact to that of the original graph, while the value of NMI decreases as the detected communities differ from those of the original graph.

As an example, we can consider the initially directed graph in **Figure 2(a)**. Each edge in a directed graph has a defined direction. In that graph, we can see 3 disjointed sets of vertices with only internal edges. In the figure, each community detected by a hypothetical algorithm is represented by a different colour.

Figure 2(b) shows the same graph after adding 2 additional edges to each set of vertices, which represents an increase of 25% of the edges, and thus the mixing parameter μ is equal to 0.25.

The main synthetic generator used for benchmarking, to the best of our knowledge, was published in [12]. As an example, we have generated with it a binary, a directed, a weighted, and a weighted-directed graph, all of them with a defined weight of edges. That synthetic generator also provides these graphs with different amount of inter-communities edges. Then, we then apply the Girvan-Newman community detection algorithm to each of these graphs. The results for each of the graphs are shown with the Normalized Mutual Information (NMI) in **Figure 3**. **Figure 3** shows that the original communities are no longer detected after increasing about 40% the number of edges.

3. Objectives

The existing community detection algorithms had been used to find communities in dense and disperse networks. The former are those networks where elements are highly related among themselves, with many more edges than vertices.

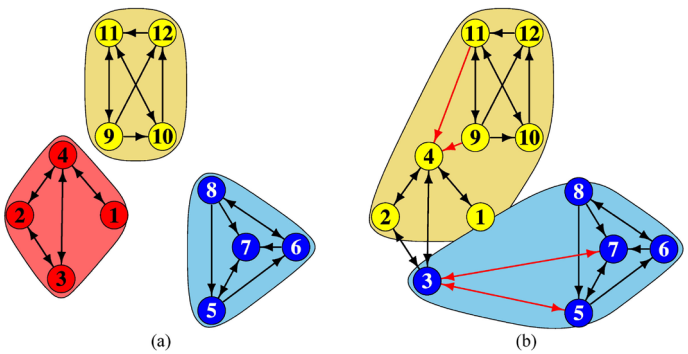


Figure 2. Example of detection of communities, before (a) and after (b) the addition of edges.

[1]The source code for calculating the NMI is available at [18].

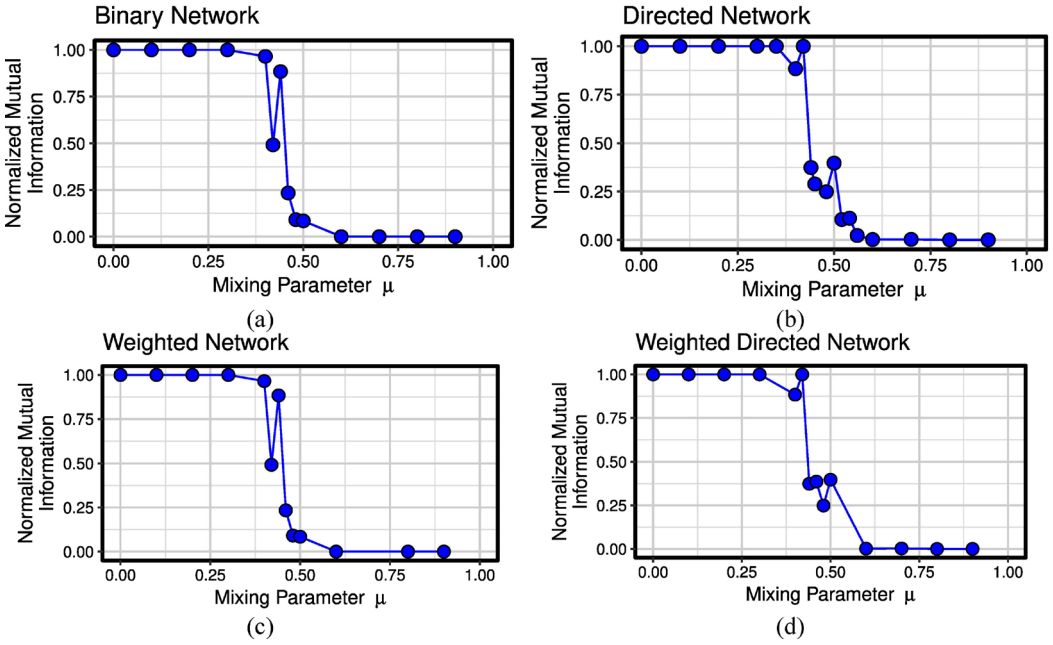


Figure 3. Example of obtained values of NMI for different values of μ. The value of NMI decreases as the detected communities differ from those of the original graph.

The latter, which is more common, consists of networks where the number of edges is much smaller. The communities in these networks have groups of vertices highly connected between them and poorly connected with the vertices in the other communities [9] [12] [13] [14] [19].

However, they have difficulties to detect the communities in directed graphs with only a few vertices that have a high output weight, while most of the vertices have a total low output weight, as we found in real scenarios.

Our objective is to obtain a synthetic generator of such type of graphs. Because those graphs are not conveniently supported by the existing generators, and the challenge of detection communities on those graphs. The difficulty on detection is due to the existing algorithms in most of the cases that were originally designed for non-directed graphs or symmetrical and regular graphs.

This new synthetic generator will allow us to evaluate and compare community detection algorithms on this kind of graphs (benchmarking of detection algorithms). It will be also of particular interest to researchers who develop new algorithms.

3.1. Proportion of Vertices with High and Low Weights on Their Output Edges

The first big difference is the proportion of vertices with high and low weights on their output edges. As an example, **Figure 4** shows the weight of the output edges sorted from highest to lowest in the largest community of the SED-graph.

Figure 4 shows that only a few vertices have a high value on their total output weight, while most of the vertices have a total low output weight. This is important because it makes it difficult to correctly detect the communities with edges

connecting them, even if those edges have a low weight.

The fitting results of Figure 4 on different functions show that the best global fit is to the exponential function, which also keeps the difference on the weight ratio among vertices with higher weight and those with a lower weight.

The results of the estimation of the fitting parameter appear in Table 1, which shows a very good adjustment.

We can consider the model as statistically significant and we have good adjustment because the p-values are less than the pre-determined statistical significance level, which is ideally 0.05 (probability of 5% [20]). We can also see that the residuals are acceptable because they are centered around zero, *i.e.* the fit function is centered in the distribution of measures. And there is not any outlier, *i.e.* there is not any measure far from the fit function.

3.2. Ratio of the Size of the Communities

The second main difference is the ratio of small communities over the total amount. We can see the number of communities sorted by size of the SED-graph in Figure 5.

We look for a function that fits with the distribution of community sizes in

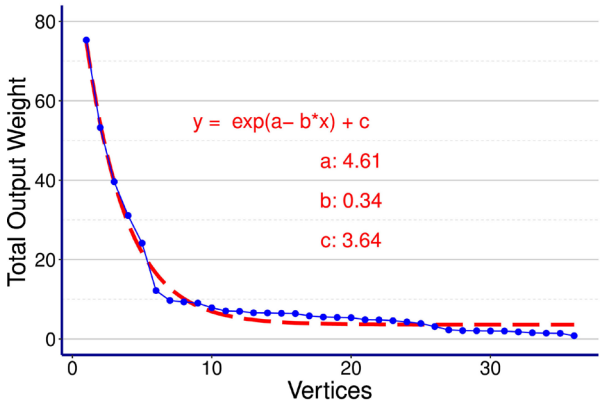


Figure 4. Weight of the output edges in the largest community of the SED-graph generator.

Table 1. Goodness-of-fit of the weight of the output edges in the SED-graph with an exponential function using R.

GOODNESS-OF-FIT STATISTICS
Formula: $y \approx \exp(a - b * x) + c$
Parameters:

	Estimate	Std. Error	T value	Pr (>|t|)	
a	4.6081	0.0322	143.122	2e−16	***
b	0.3419	0.0144	23.737	2e−16	***
c	3.6383	0.3813	9.5426	5.17e−11	***

Residual standard error: 1.875 on 33 degrees of freedom
Number of iterations to convergence: 7
Achieved convergence tolerance: 1.34e−06

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1".

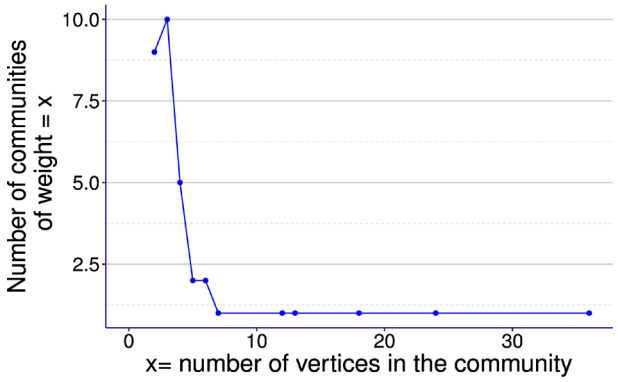


Figure 5. Amount of communities sorted by size in the SED-graph.

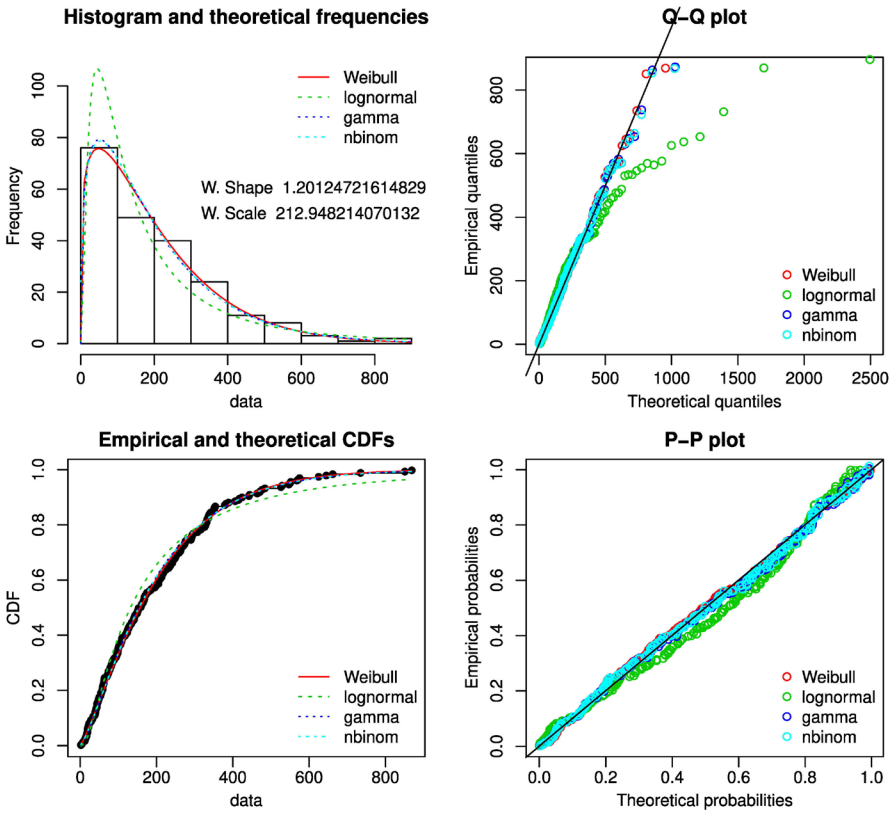


Figure 6. Fit of the community sizes in **Figure 5** (SED-graph) with different functions using R.

Figure 5. **Figure 6** shows the functions that fit better with that distribution. In particular, we found that the function that fits more accurately was the Weibull function.

Table 2 shows the numerical metrics for different criteria for measuring the fitting error with different types of functions. The Weibull function is the one that achieves the best coefficient in all the criteria (the smaller the better). In addition to considering these adjustment criteria, **Figure 6** shows that the Weibull function is always the one that best adjusts to the data samples for different types of probability analyses.

The resulting estimation of the fitting parameters of the distribution Weibull by maximum likelihood is: shape = 1.201 (Std. Error: 0.064) and scale = 212.948 (Std. Error: 12.771). The goodness of the fit appears in Table 2.

3.3. Modeling the Number of Output Edges of the Vertices

Figure 7(a) shows the number of output edges of the largest community in the SED-graph. In that figure, the vertices are sorted from smaller to the largest amount of output edges. In the figure, there are 6 possible sets of vertices when considering their number of output edges. We can see that the largest set corresponds to the set of vertices with the lowest number of output edges, and, that the number of vertices in each set is smaller or equal that in the next set with the larger number of output edges.

Our best effort for the challenge of modeling this curve resulted in the number of output edges of each set as a function of the number of the set, when the sets are numbered from largest to the smallest size. Figure 7(b) shows the fit of this function, which corresponds to polynomial $y = b + ax + c/x^2$.

The results of the estimation of the fitting parameters appear in Table 3. It is a very good adjustment, for the same reasons as shown in Section 3.1.

Table 2. Goodness-of-fit of the community sizes in the SED-graph with different functions.

GOODNESS-OF-FIT STATISTICS				
	Weibull	**gamma**	**nbinom**	**lognormal**
Kolmogorov-Smirnov statistic	0.03747684	0.04094466	0.04107869	0.08033524
Cramer-von Mises statistic	0.04279517	0.05291148	0.05271877	0.36063454
Anderson-Darling statistic	0.26663825	0.30999310	0.31165762	2.21300868
GOODNESS-OF-FIT CRITERIA				
	Weibull	**gamma**	**nbinom**	**lognormal**
Akaike's Information Criterion	2689.306	2689.608	2690.008	2717.229
Bayesian Information Criterion	2696.038	2696.340	2696.740	2723.961

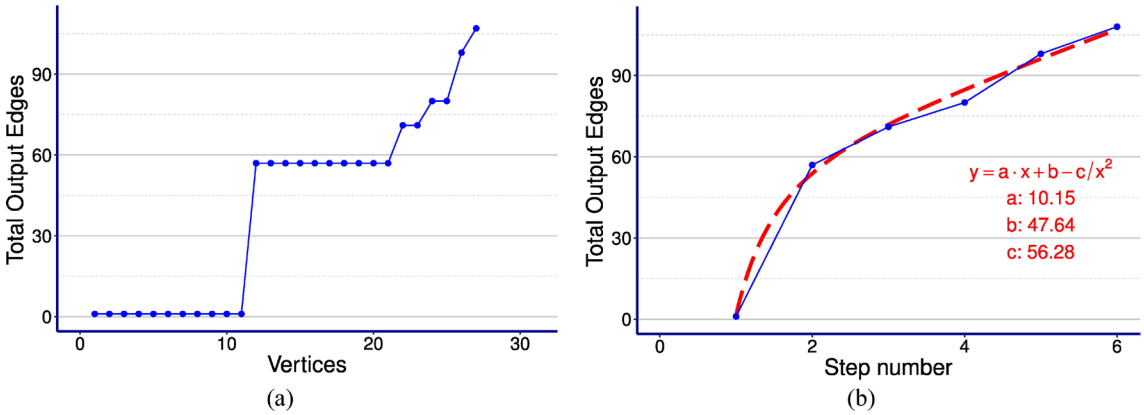


Figure 7. (a) The number of output edges of the largest community in the SED-graph, and (b) the fit of the step levels with a polynomial function.

Table 3. Goodness-of-fit of the weight of the step levels of the output edges of the largest community in the SED-graph with a polynomial function using R.

GOODNESS-OF-FIT STATISTICS
Formula: $y \approx ax + b - c/x^2$
Parameters:

	Estimate	Std. Error	t value	Pr (>|t|)	
a	10.149	1.367	7.423	0.00506	***
b	47.637	6.364	7.485	0.00494	***
c	56.284	6.785	8.295	0.00367	***

Residual standard error: 3.539 on 3 degrees of freedom
Number of iterations to convergence: 1
Achieved convergence tolerance: 2.694e−08

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1''.

In the next section, we proceed to describe the graph generator with these statistical properties.

4. Proposed Synthetic Graph Generator

In this section, we describe the proposed generator, which is highly configurable according to the needs of the user, allowing to generate directed and non-directed graphs, symmetric and regular, as well as non-symmetrical and non-regular.

In order to facilitate the description, we propose first a simplified version of the generator, and later the complete version with additional parameters which achieves the modeling our target graphs.

The first version is a simplified version of the algorithm referred to as "generator of Directed weighted graphs which vertices have an Unbounded number of Output edges" (DUO), and the second one is referred to as "Directed weighted graph which vertices have a bounded number of Output edges" (DBO).

4.1. Generator of Directed Weighted Graphs with Unbounded Number of Output Edges (DUO)

In this first generator, the number of vertices NC per community is obtained randomly from a normal function that is defined by the parameters provided at the generator input. Next, it creates a routing table, where each route is defined by a start in one of the vertices, visiting from that vertex other vertices of the same community different from those visited in that same route (See **Algorithm 1**).

The number and length of these paths are optional input parameters, the generator will use default values calculated as a function of the size of each community when the user doesn't define them.

In order to obtain dominant vertices with a stronger connection in each community, these paths will give preference to visiting certain vertices. It is achieved using the following probability function to visit a vertex i:

$$p(i) = \frac{2^{N-i}}{\sum_{j=1}^{N} 2^{N-j}}; \text{ Where } N \text{ is the amount of vertices in the community} \quad (1)$$

Algorithm 1. "Directed Unbounded number of Outputs" (DUO) Generator.

```
1 Function next_popular_vertex (N, Step,Exclude_destinations[ ])
      |  // This function returns the identifier id of a vertex
      |  // the vertex not included in the vector Exclude_destinations[0::Step - 1] if Step > 0
      |  // This identifier is a random value smaller than N and
      |  // it is generated with probability: Prob(id) = 2^(N-id) / sum_{j=1}^{N} 2^(N-j)
2     |  id ← prob_rand(N; Prob;Exclude_destinations[ ]).
3     └  return id
// N is the total amount of Vertices in the community
4 Function Generator (N, Length_paths, Total_paths)
      |  // This function defines the edges and weights of a single community
5     |  for Source = 0 → N do
6     |  |  for Destination = 0 → N do
7     |  |  |  edge_weight[Source][Destination] ← 0
8     |  for X = 0 → Total_paths do
9     |  |  for Step = 0 → Length_paths do
10    |  |  |  if Step == 0 then
11    |  |  |  └  Path[Step] ← next_popular_vertex(N, Step, Ø)
12    |  |  |  else if Step > 0 then
13    |  |  |  |  Path[Step] ← next_popular_vertex(N; Step; Path[ ])
14    |  |  |  |  Previous_v ← Path[Step - 1]
15    |  |  |  |  Current_v ← Path[Step]
16    └  └  └  └  edge_weight[Previous_v][Current_v]+=1=(2^Step)
```

Each path is composed of a list of N vertices, increasing the weight of the edge between the vertex i and the $i+1$ of the path with $1/2^i$, *i.e.* the weight contribution of each step in the path is half of the previous step.

As an example, Figure 8(a) shows a graph of 3 communities. Figure 8(b) and Figure 8(c) show the cumulative weight of the output and the input edges, respectively.

The high degree of connectivity is shown in Figure 8(a) and considering that the small-world [21] style graphs have a small set of vertices with high connectivity degree, while many other vertices have a low degree of connectivity, motivated a second version of the generator.

4.2. Generator of Directed Weighted Graphs with Bounded Number of Output Edges (DBO)

This second generator is based on previous one, which the main difference is that it bounds the number of the output edges on each vertex.

The fixed number of output edges of each vertex is defined by:

$$\text{Number of output edges of the vertex}\left(i\right) = round\left(1.2 * \sqrt{N-i}\right) \tag{2}$$

where the vertices within a community have assigned a value of i between 0 and

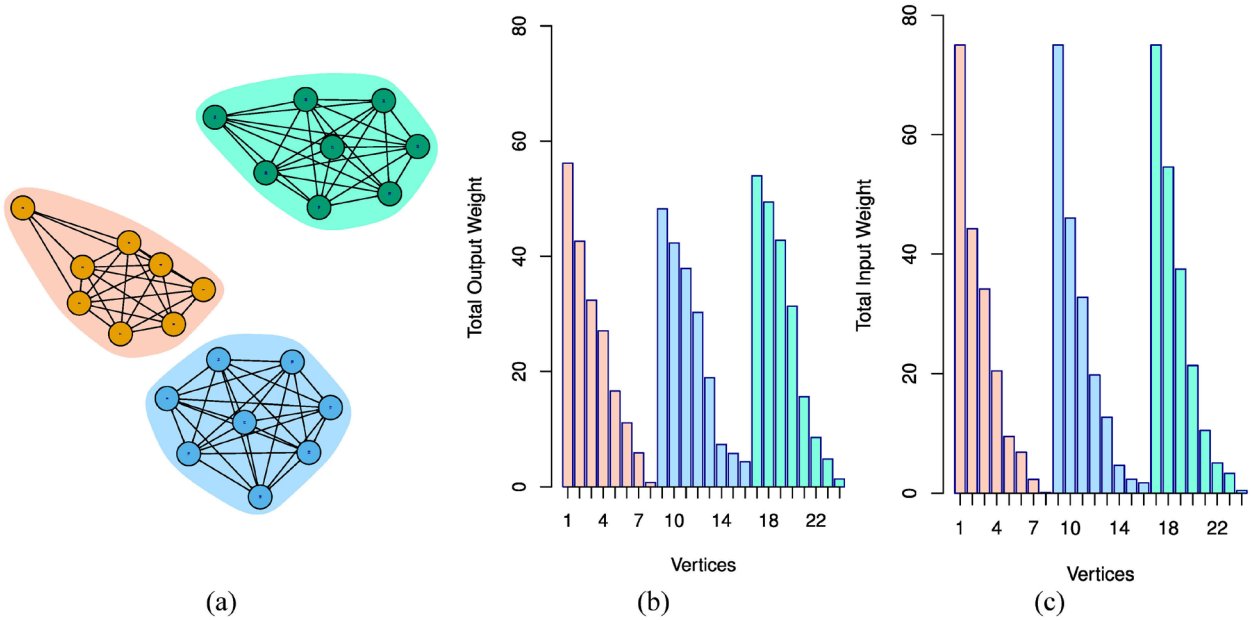
the total number of vertices within the community minus 1.

The definition of weights of edges is also done in the same way as for DUO. But, the definition of paths will be done with the restriction of the number of output edges, which limits the possible random paths that can be defined. We have to take into account that the limitation of the number of output edges impose that some of the new paths will not reach the desired length. It is because some paths reach a vertex from where the path cannot go to any other vertex which the path has not already visited.

As an example, Figure 9, shows a graph of 4 vertices where all the possible directed edges already defined, where it is not possible to define a 3 hops path without visiting a vertex more than one time. In particular, the only possible path starting at A has to be the path A → B. It is limited to having a single hop because no vertex can be visited more than once within the same path, and the only one output edge from vertex B leads to vertex A which is already visited by this path.

Figure 10(a) shows a graph with 3 communities, where the number of vertices with the highest connectivity degree is smaller than the number of vertices with a lower connectivity degree.

The existing random graph generators require input statistics such as the bounding limits on the number of vertices per community, number of communities,



(a) (b) (c)

Figure 8. Example of (a) a graph generated by the DUO generator, (b) total weight of the output edges of each vertex, and (c) total weight of the output edges of each vertex.

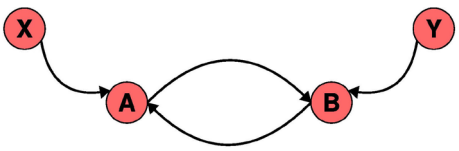


Figure 9. Example of a graph on with a bounded number of output edges.

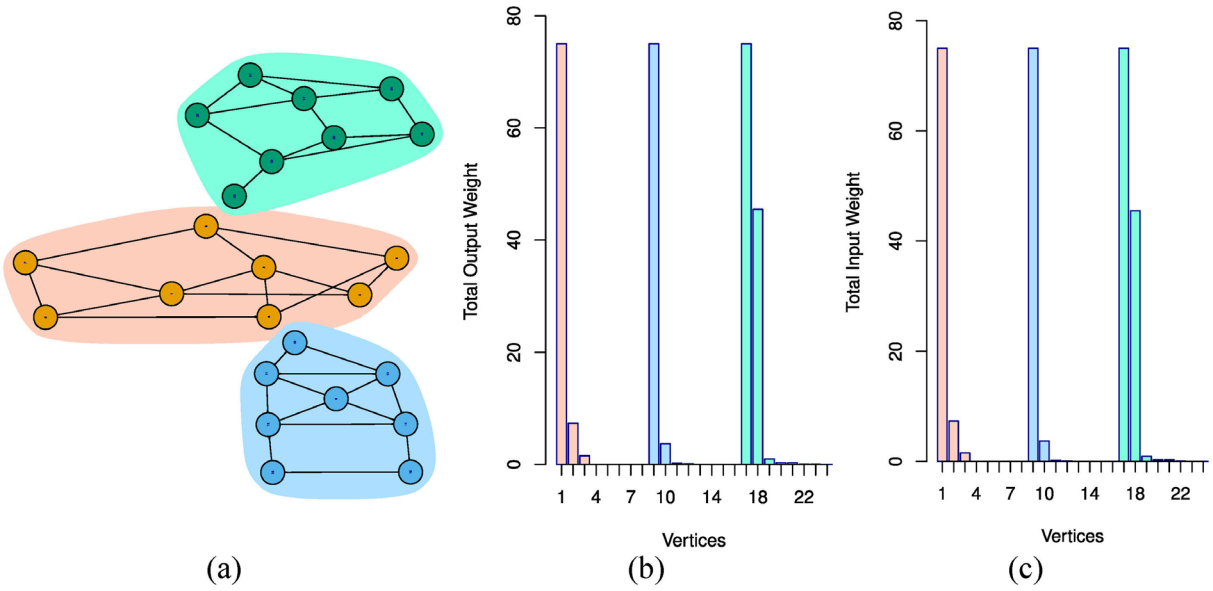


Figure 10. Example of (a) a graph generated by DBO, (b) sum of the output edges per vertex, and (c) sum of the input edges per vertex.

and the number of edges per vertex. However, it is not enough for generating graphs like in the SED-graph as shown and analyzed in this section.

Therefore, in this paper, we consider that are needed additional parameters, such as the number of vertices per community and number of communities. In this paper, we propose **Algorithm 2**, which generates communities with the distribution of sizes defined by the Weibull function, and the edges and their weights are defined random paths inside each community.

Figure 11 shows (a) a graph generated by DUO, (b) the SED-graph, and (c) a graph generated by DBO. These graphs show the improvement of the DBO generator for providing graphs with the interconnection ratio of the SED-graph. This was the reason to develop the DBO generator, which is an evolution of the DUO generator.

5. Analysis of a Generated Random Graph

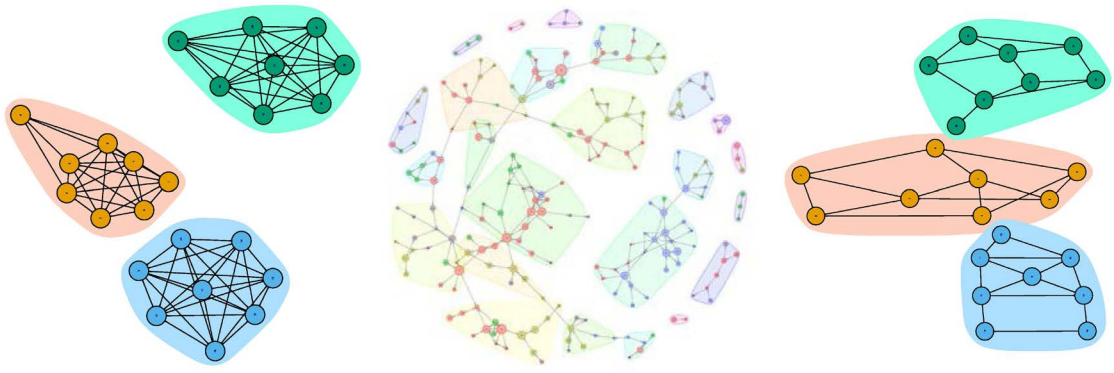
The purpose of this section is to analyze if the generated synthetic graphs have the main properties of the weak connected directed graphs, which make difficult to detect communities on them. Those properties are the Ratio of the Total Output Weights, and the Ratio of the size of the communities.

5.1. Ratio of the Total Output Weights

The distribution of output edges of the larger community in the graph is represented by a dotted line in **Figure 12**. We considered the larger community because it is the one that provides the greatest number of values to adjust curves. The adjustment is shown in **Figure 12(a)** for the SED-graph, and in **Figure 12(b)** for a random graph generated by the DBO generator proposed in this article. In both cases, the best fit function was the function $y = \exp(a - b * x) + c$.

Algorithm 2. "Directed Bounded number of Outputs" (DBO) Generator.

```
1 Function next_popular_vertex (N, Step, Exclude_Dests[ ], Possible_Dests[ ])
      // This function returns the identifier id of a vertex
      // The vertex must be in the list Possible_Dests[ ]
      // excepting those included in the array Exclude_Dests[0::Step - 1] if Step > 0
      // Such identifier is a random value smaller than N and
      // it is generated with probability:  Prob(id) = 2^(N-id) / Σ_{j=1}^{N} 2^(N-j)
2     id ← prob_rand(N; Prob;Exclude_Dests[ ]; Possible_Dests[ ]).
3     return id
// N is the total amount of Vertices in the community
4 Function Generator (N, Length_paths, Total_paths)
5     for Source = 0 → N do
6         Visited_Vertices[Source][ ] = Ø
7         Max_output_edges[Source]= (int) (1.2 * sqrt(N - Source))
8     for Source = 0 → N do
6         for Destination = 0 → N do
7             edge_weight[Source][Destination] ← 0
8     for X = 0 → Total_paths do
9         for Step = 0 → Length_paths do
10            if Step == 0 then
11                Path[Step] ← next_popular_vertex(N, Step, Ø, ALL)
12            else if Step > 0 then
13                if Max_output_edges[Previous_v]> 0 then
14                    Path[Step] ← next_popular_vertex(N, Step, Path[ ] ,ALL)
15                    if Current_v not in Visited_Vertices[Previous_v][ ] then
16                        Max_output_edges[Previous_v] - = 1
17                        insert(Visited_Vertices[Previous_v][ ], Current_v)
18                    else
19                        Path[Step] next_popular_vertex(N, Step, Path[ ],Visited_Vertices[Previous_v][ ])
20                    Current_v ← Path[Step]
21                    Previous_v ← Path[Step - 1]
22                edge_weight[Previous_v][Current_v]+=1=(2^Step)
```



(a) (b) (c)

Figure 11. Example of (a) a graph generated by the DUO, (b) the SED-graph, and (c) an example of a graph generated by the DBO. The interconnection degree between vertices in the same community in the SED-graph is more similar to the interconnection degree in the graph generated by the DBO.

5.2. Ratio of the Size of the Communities

Figure 13(a) shows, with a line with points, the number of communities ordered by size, in the case of (a) SED-graph and (b) a random graph generated by the DBO generator proposed in this article. In both cases, the adjustments with the Weibull function appear as a dashed line.

5.3. Modeling the Number of Output Edges of the Vertices

Figure 14(a) and Figure 14(b) show the fit of the number of output edges of the largest community with a polynomial function in the SED-graph and a synthetic random graph generated by the DBO algorithm. These figures show that the generator provides graphs with edges with the ratio of output edges as in the SED-graph.

As the last result, we can see in Figure 15 that the plot shows some level steps when the vertices ordered by their total number of output edges, for both cases, the SED-graph and the generated graph with the DBO algorithm. This results from the way in which the graph was generated, although this pattern was not imposed in the algorithm. We also consider this last result as satisfactory, since

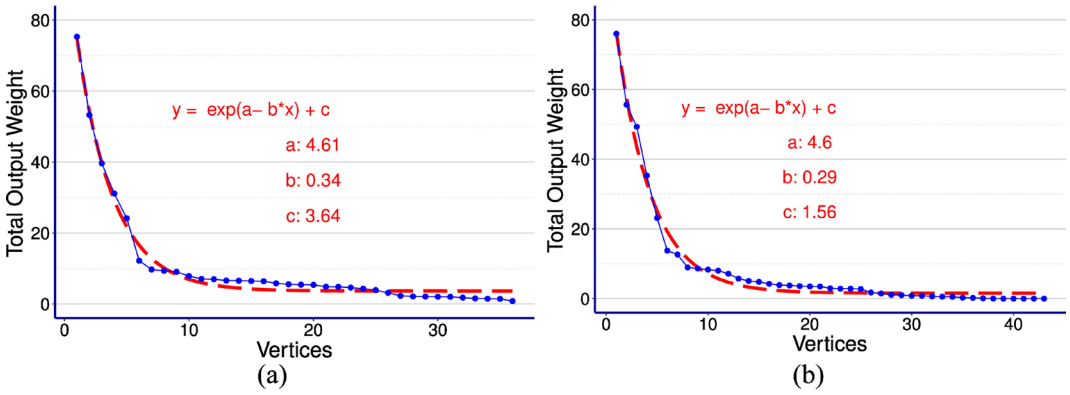


Figure 12. Weights of the output edges of the largest community, (continuous line), in the case of (a) SED-graph and (b) a random graph generated by the DBO generator. The adjustment with the function $y = \exp(a - b * x) + c$ is represented by a dashed line.

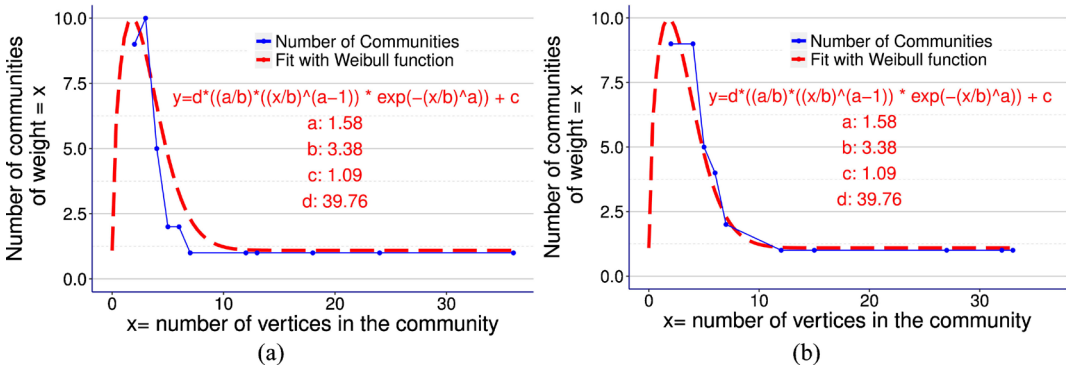


Figure 13. Lines with points show the number of communities ordered by size, in the case of (a) SED-graph and (b) generator by the proposal of this article. The adjustments with the Weibull function appear as a dashed line.

to the best of our knowledge none of the existing graph generation algorithms produce this type of step pattern.

5.4. Computation time

Figure 16(a) shows the mean value with confidence intervals of the computation time required of the DBO algorithm for generating a graph with 8 different

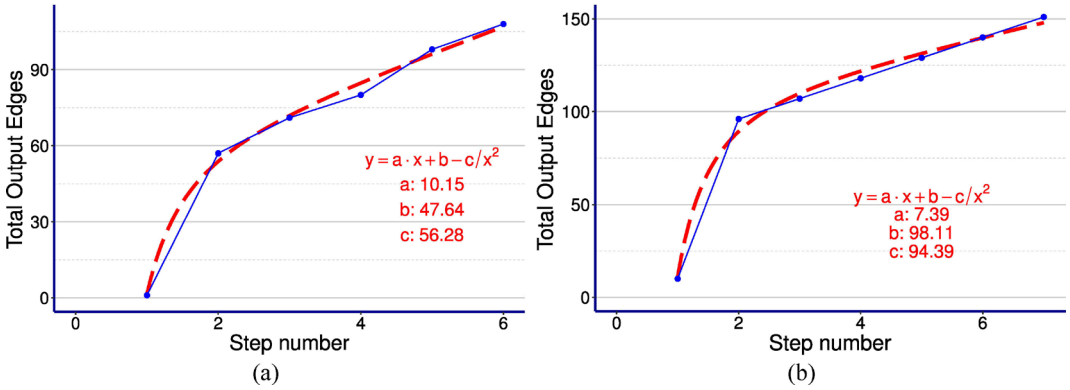


Figure 14. Fit with a polynomial of the step levels of the number of output edges of the largest community in the (a) SED-graph, and (b) in a synthetic random graph generated by the DBO.

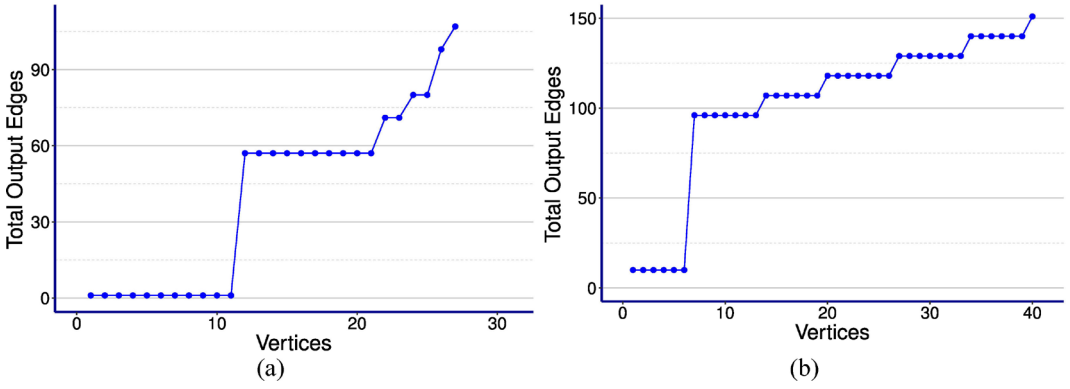


Figure 15. The number of output edges of the largest community in (a) the SED-graph and in (b) a generated graph.

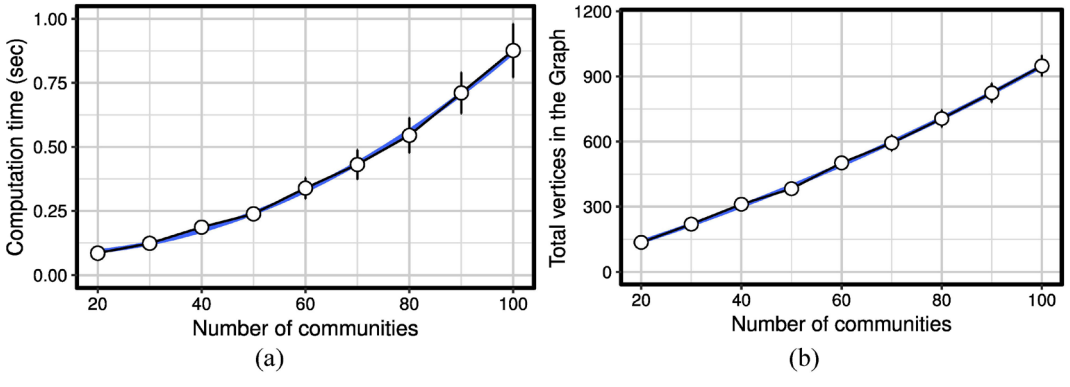


Figure 16. Mean value with confidence intervals (95%) of (a) the total computation time of graphs generated and (b) the total amount of vertices in the graph. Data from both plots corresponds to the DBO algorithm, and each point is calculated from a set of 10 different graphs.

amount of interconnection edges between communities when using an Intel I5-5300U 2.3 GHz running Ubuntu 16.04 (64 bits). **Figure 16(b)** shows the mean value with confidence intervals of the number of vertices on each graph.

6. Conclusions

In this paper, we have proposed two parametrizable benchmarking algorithms that can generate a wide range of graphs, including graphs not-supported by the existing generators. In particular, these previously not-supported graphs are needed for the study of general problems in "badly conditioned" directed graphs from the traditional point of view, for which it is particularly difficult to detect their communities. In this way, the proposal in this paper intends to cover a space that until now has not been studied due to its difficulty.

We consider that the availability of synthetic directed graphs is essential for the development of new community detection algorithms, and therefore the proposed generators in this paper can be a key element. The source code of the proposed generators (written in C) is available in GitHub [22].

Funding

This work has been supported by the Project "Complex Networks" from the *Instituto Universitario de Matemática Multidisciplinar* (*IUMM*) of the Universitat Politècnica de València (UPV) [under Grant number (266500194) 20170251-Complex-Networks-UPV] [23].

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. and Hwang, D.U. (2006) Complex Networks: Structure and Dynamics. *Physics Reports*, **424**, 175-308. https://doi.org/10.1016/j.physrep.2005.10.009

[2] Fortunato, S. (2010) Community Detection in Graphs. *Physics Reports*, **486**, 75-174. https://doi.org/10.1016/j.physrep.2009.11.002

[3] Newman, M.E. (2003) The Structure and Function of Complex Networks. *Society for Industrial and Applied Mathematics* (*SIAM*) *Review*, **45**, 167-256. https://doi.org/10.1137/S003614450342480

[4] Van Der Hofstad, R. (2016) Random Graphs and Complex Networks, Volume 1. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781316779422

[5] Lancichinetti, A. and Fortunato, S. (2009) Benchmarks for Testing Community Detection Algorithms on Directed and Weighted Graphs with Overlapping Communities. *Physical Review E*, **80**, Article ID: 016118. https://doi.org/10.1103/PhysRevE.80.016118

[6] Newman, M.E.J. (2004) Fast Algorithm for Detecting Community Structure in Networks. *Physical Review E*, **69**, Article ID: 066133. https://doi.org/10.1103/PhysRevE.69.066133

[7] Newman, M.E.J. (2006) Finding Community Structure in Networks Using the Eigenvectors of Matrices. *Physical Review E*, **74**, Article ID: 036104. https://doi.org/10.1103/PhysRevE.74.036104

[8] Newman, M.E.J. and Girvan, M. (2004) Finding and Evaluating Community Structure in Networks. *Physical Review E*, **69**, Article ID: 026113. https://doi.org/10.1103/PhysRevE.69.026113

[9] Raghavan, U.N., Albert, R. and Kumara, S. (2007) Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks. *Physical Review E*, **76**, Article ID: 036106. https://doi.org/10.1103/PhysRevE.76.036106

[10] Lancichinetti, A. and Fortunato, S. (2009) Community Detection Algorithms: A Comparative Analysis. *Physical Review E*, **80**, Article ID: 056117. https://doi.org/10.1103/PhysRevE.80.056117

[11] Girvan, M. and Newman, M.E. (2002) Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences*, **99**, 7821-7826. https://doi.org/10.1073/pnas.122653799

[12] Lancichinetti, A., Fortunato, S. and Radicchi, F. (2008) Benchmark Graphs for Testing Community Detection Algorithms. *Physical Review E*, **78**, Article ID: 046110. https://doi.org/10.1103/PhysRevE.78.046110

[13] Feng, S.M., Hu, B.Y., Nie, C. and Shen, X.H. (2016) Empirical Study on a Directed and Weighted Bus Transport Network in China. *Physica A*: *Statistical Mechanics and Its Applications*, **441**, 85-92. https://doi.org/10.1016/j.physa.2015.08.030

[14] Sienkiewicz, J. and Hołyst, J.A. (2005) Statistical Analysis of 22 Public Transport Networks in Poland. *Physical Review E*, **72**, Article ID: 046127. https://doi.org/10.1103/PhysRevE.72.046127

[15] U.S. Department of Transportation, Bureau of Transportation Statistics (2017) Transportation Economic Trends. https://www.bts.gov

[16] Himelboim, I., Smith, M.A., Rainie, L., Shneiderman, B. and Espina, C. (2017) Classifying Twitter Topic-Networks Using Social Network Analysis. *Social Media + Society*, **3**, Article ID: 2056305117691545. https://doi.org/10.1177/2056305117691545

[17] Hervas, A., Soriano, P.P., Jimenez, A., Peinado, J., Capilla, R. and Montañana, J.M. (2017) Modeling Human Behavior: Individuals and Organizations, Chapter "Applying a Graph Model for the Spanish Public University System". Nova Science Publishers, Inc., New York, 9-24.

[18] Joseph, K., Carley, K.M. and Hong, J.I. (2014) Project tist_article. https://github.com/kennyjoseph/tist_article/tree/master/mutual3

[19] Latora, V., Nicosia, V. and Russo, G. (2017) Complex Networks. Principles, Methods and Applications. Cambridge University Press, Cambridge. https://doi.org/10.1017/9781316216002

[20] Crawley, M.J. (2007) The R Book. John Wiley & Sons Ltd., Hoboken.

[21] Watts, D.J. and Strogatz, S.H. (1998) Collective Dynamics of "Small-World" Networks. *Nature*, **393**, 440-442. https://doi.org/10.1038/30918

[22] Montañana, J.M., Hervas, A. and Soriano-Jiménez, P.P. (2018) Project Graph-Generators. https://github.com/jmmontanana/Graph-Generators/tree/master/Weakly-Connected-Directed-Graphs

[23] Universidad Politècnica de València. http://data.crossref.org/fundingdata/funder/10.13039/501100004233