Received: 20 November 2018 | Accepted: 22 June 2018
DOI: 10.1002/int.22034

RESEARCH ARTICLE

WILEY

Synthetic generation of spatial graphs

Vicenç Torra[1] | **Annie Jonsson**[2] | **Guillermo Navarro-Arribas**[3,4] | **Julián Salas**[4,5]

[1]School of Informatics, University of Skövde, Skövde, Sweden

[2]School of Biosciences, University of Skövde, Skövde, Sweden

[3]Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Barcelona, Spain

[4]Center for Cybersecurity Research of Catalonia (CYBERCAT), Catalonia, Spain

[5]Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya (UOC), Barcelona, Spain

Correspondence
Vicenç Torra, School of Informatics, University of Skövde, Skövde 54128, Sweden.
Email: vtorra@ieee.org;
Vicenç Torra, Hamilton Institute, Maynooth University, W23 A5Y6 Ireland.

Funding information
Swedish VR, Grant/Award Number: VR 2016-03346; Spanish MINECO, Grant/Award Numbers: TIN2014-57364-C2-2-R, TIN2014-55243-P

Abstract

Graphs can be used to model many different types of interaction networks, for example, online social networks or animal transport networks. Several algorithms have thus been introduced to build graphs according to some predefined conditions. In this paper, we present an algorithm that generates spatial graphs with a given degree sequence. In spatial graphs, nodes are located in a space equiped with a metric. Our goal is to define a graph in such a way that the nodes and edges are positioned according to an underlying metric. More particularly, we have constructed a greedy algorithm that generates nodes proportional to an underlying probability distribution from the spatial structure, and then generates edges inversely proportional to the Euclidean distance between nodes. The algorithm first generates a graph that can be a multigraph, and then corrects multiedges. Our motivation is in data privacy for social networks, where a key problem is the ability to build synthetic graphs. These graphs need to satisfy a set of required properties (e.g., the degrees of the nodes) but also be realistic, and thus, nodes (individuals) should be located according to a spatial structure and connections should be added taking into account nearness.

KEYWORDS
data privacy, graphs generating algorithms, network modeling, spatial graphs

1 | INTRODUCTION

Graphs can be used in a large number of scenarios to model interactions. For example, they have been used to model online social networks, relationships between colleagues, coauthorship networks, and transportation networks. Applications can be found in areas as different as computer science, biology, economics, epidemiology, and social sciences.

The broad applicability of network modeling has brought forth the development of measures and algorithms for studying graphs. There are methods that focus on the generation of graphs from a probability distribution on the degree sequences. This is the case of the Barabasi algorithm[1], which is available in a variety of software packages as, for example, igraph[2] of R. It permits the generation of graphs that are scale-free, that is, that the degree distribution follows a power law.

The literature for graph generation is extense. There are two main approaches for uniform graph generation: the configuration model (CM)[3,4] and the Markov chain approach.[5]

In Jerrum and Sinclair,[5] a polynomial algorithm for generating a graph with a given degree sequence from a distribution that is arbitrarily close to uniform is presented. In Jerrum and McKay,[6] it is noted that for the generating algorithm to have an efficient solution the input degree sequence has to be restricted to a P-stable class, and a characterization of P-stability is obtained. This is further improved in Salas and Torra[7] and such theoretical considerations are used for generating k-anonymous graphs in Salas and Torra.[7,8]

The Markov chain approach is used for generating connected graphs in Gkantsidis et al[9] and Viger and Latapy.[10] It is used in Ying and Wu[11] for privacy preserving social network publishing. Cooper et al,[12] Kannan et al,[13] and Miklós et al[14] proved that the Markov chains they used for generating graphs are rapidly mixing, and Milo et al[15] make a comparison between both aforementioned methods that they call the "switching" and "matching" methods, recommending the switching method for its combination of speed and accuracy.

In Britton et al,[16] the authors generate graphs based on the CM. Briefly, the CM constructs a graph in the following way: for each vertex v_i of degree d_i, generate d_i stubs (or half-edges) and then join the stubs at random to form edges between the vertices. This algorithm may obtain multigraphs, and to avoid them, the authors propose to remove edges. However, Viger and Latapy[10] mention that very little is known about the impact of these removals on the obtained graphs and on their degree distribution, apart from the expected size of the main component.[17-19]

Two main results are of relevance in our work. First, the condition by Erdös-Gallai[20] checks whether a degree sequence is graphical. Second, the algorithm by Havel[21] and Hakimi[22] that builds a graph from a given graphical degree sequence. The former condition is used to ensure that we can build a graph. The Havel-Hakimi algorithm (as well as other algorithms for graph generation) is discussed as an alternative approach in Section 3.

1.1 | Our motivation

The problem of graph generation is of relevance in data privacy for social networks.[23,24] On the one hand because we can use synthetic data to test and tune algorithms without the difficulties of doing this on real data.[25] On the other hand because we can use synthetic data of good quality as a way to mask the original data, as common practice in regular databases.[26] This diminishes or avoids reidentification problems.[27,28]

Recall that a way to protect sensitive information from social networks is to protect their underlying structure: its graph. There are different approaches according to the type of information available to the intruder. When we assume that the information available to the intruder is the degree sequence (or a set of degrees), an approach is to perturb the degrees of nodes. k-Anonymity[29] has also been considered for graphs (see, e.g., Stokes and Torra[27] for a discussion of this concept on graphs). There are methods to modify the edges of the original graphs,[30] that construct supernodes,[31-33] and that generate directly a new graph. The concept of P-stability has recently been used related to both k-degree anonymity and graphic sequences in Salas and Torra,[7] Salas and Torra,[8] Torra et al.[34]

In this area, we need that synthetic graphs satisfy the properties of the original graphs. Spatial properties are important to model correctly social networks. For example, distribution of individuals is not uniform on the space. Because of that we study the generation of spatial graphs.

Spatial graphs are graphs for which the nodes are located in a space equipped with a metric.[35] In our case, we deal with a two-dimensional space (a map) with an underlying probability distribution representing the density of the individuals of a population.

Then, we generate spatial graphs from degree sequences. In this way, we can, for example, first protect a degree sequence (masking it and resulting into a k-anonymous sequence) and then generate a spatial graph from the protected sequence.

1.2 | Our contribution

The objective of this paper is to introduce a method to generate spatial graphs where nodes are located on a map according to an underlying probability distribution. For example, regions with high population density will have more nodes than regions with low population density. This may be used for modeling networks on top of maps.

Then, edges will be assigned according to the nearness between nodes. Our assumption is that individuals will be more likely to be related to individuals that are close in the map. A very simple example could be the friendship relations on online social networks, that are more likely to occur between individuals that are or (have been) close geographically.

The construction of graphs from a degree sequence is not easy when we want additional constraints than the degree sequence itself. In addition, methods should be fast when we consider a large number of edges. See, for example, Blitzstein and Diaconis[36] for details and references.

Following Håkanson et al[37] and Lennartsson et al,[38] we consider the problem of generating the graph in a spatial setting. In addition, we add the constraint of a given degree sequence. Thus, we develop an algorithm that at the same time considers the constraints of spatial graphs and the ones of the given degree sequence.

Our method is a greedy algorithm. We prove mathematically some results about the construction of the graph, and show the validity of the approach experimentally.

1.3 | Paper structure

The structure of the paper is as follows. We present in Section 2 some concepts and results that are needed in the rest of the paper. In Section 3, we describe our procedure and its properties. In Section 4, we report our experiments. The paper finishes with some conclusions and lines for future work.

2 | PRELIMINARIES

A graph is defined in terms of a set of nodes and a set of edges on pairs of nodes. It is formally defined as the pair $G = (V, E)$ where V is the set of nodes and E the set of edges ($E \subseteq V \times V$). We denote edges with (a, b) or, when no confusion arises, with ab. A loop is an edge (a, b) with $a, b \in V$ such that $a = b$. If E is a multiset of edges instead of a set, then we say that (V, E) is a multigraph. The *degree sequence* of a graph is the monotonic nonincreasing sequence of its vertex degrees.

A graph is *simple* if it does not contains loops or multiedges.

An *alternating circuit* is a sequence of nodes $v_0 v_1 ... v_{2r} = v_0$ such that the edge $v_{2i-1}v_{2i}$ belongs to the graph and the edge $v_{2i}v_{2i+1}$ is in the complement of the graph.

Complementing along an alternating circuit $v_0 v_1 ... v_{2r} = v_0$ is removing the edges $v_{2i-1}v_{2i}$ and adding the edges $v_{2i}v_{2i+1}$ to the graph.

We present below a theorem by Havel[21] and Hakimi[22] about the construction of a graph from a degree sequence. This is a constructive theorem about when a degree sequence is graphical. That is, given a degree sequence, is there any graph that can be built that has such degree sequence?

Theorem 1 *There exist a simple graph with n nodes and degree sequence $d_1 \geq ... d_n \geq 0$ ($n \geq 3$) if and only if there exists one with degree sequence*

$$d' = d_2 - 1, ..., d_{d_1+1} - 1, \quad d_{d_1+2}, ..., d_n. \tag{1}$$

This result permits to build a graph for a graphical degree sequence in a greedy way. That is, first assigning to a first node v_1 the d_1 edges, then computing d' using expression (1), and then considering a node v_2 (with maximal d'_{v_2}) and repeating the process.

3 | PROCEDURE

We propose an algorithm consisting of three steps. The first one about the construction of the underlying map would be skipped when the underlying map is given. In that case, we say that the whole graph is partially synthetic. Otherwise, we will say that it is completely synthetic.

- Step 1. Construction of the underlying map.
- Step 2. Location of nodes, according to dense regions.
- Step 3. Addition of edges between nodes proportional to nearness between nodes, in such a way that the graph obtained has a prespecified degree sequence.

We detail the three steps below. As we will see, Step 3 can generate a multigraph. To transform this multigraph into a graph, we will present in Section 3.1 some theoretical results and an algorithm.

Construction of the map. The construction of the underlying map is based on a random generation of a surface in a grid-like form, and then smoothing this surface filtering its fast Fourier transform (denoted FFT). This approach follows Håkanson et al[37] and Lennartsson et al.[38] The initial map is defined assigning a random value following a Gaussian distribution $N(0.5, 1)$ (ie, mean 0.5 and variance 1) to each point in the map. As values may be negative, values are set to zero when negatives.

We have applied two different approaches for this filtering to compare their results. Let H be a suitable filtering matrix; then, the procedure is as follows:

1. $X(x, y) \backsim N(0.5, 1)$
2. $F = \mathrm{FFT}(X)$
3. $F^{'} = F \cdot H$
4. $X^{'} = \mathrm{FFT}^{-1}(F^{'})$.

The two different approaches are two different ways to define H. We use H_1 and H_2 to denote these matrices. The two approaches are as follows:

- One approach multiplies the FFT of the matrix by a filtering function, in fact, a filtering matrix that approximates a Gaussian distribution. In more detail, we define

$$H_1(x, y) = \frac{1}{K}\frac{1}{2\pi\sigma^2}\exp^{-0.5\left(\frac{(x-x_0)^2}{\sigma^2}+\frac{(y-y_0)^2}{\sigma^2}\right)}$$

and K is such that $H_1(x_0, y_0) = 1$. Here (x_0, y_0) is the origin of the matrix of the FFT. In this approach, the filtering depends on σ.

- The second approach is to filter the FFT using the function

$$H_2 = \mathrm{FFT}(X)^{-\gamma/2}$$

where γ is a parameter of the filtering.[38] We used $\gamma = 0.5$.

Location of nodes. Once the underlying map is built, we assign k nodes to the map. The probability of assigning a node in position (x, y) is proportional to $X^{'}(x, y)$. To do so, k values in $[0, 1]$ are drawn from a uniform distribution and then using the quantile function of the cumulative distribution of $X^{''}(x, y) = X^{'}(x, y)/\sum_x \sum_y X^{'}(x, y)$. We find the values (x, y). Note that the cumulative distribution is $\tilde{X}^{''}(x_0, y_0) = \sum_{x<x_0} \sum_y X^{''}(x, y) + \sum_{x=x_0} \sum_{y \leq y_0} X^{''}(x, y)$.

Multiple nodes can be assigned to the same position (x, y). As the edges are links between pairs of nodes and not between pairs of positions, the graph will be well defined.

Addition of edges. To add the edges, we consider an iterative approach selecting a node at a time and assigning edges to this node. At any point there is a set of nodes that have not yet been processed and thus may need additional edges. We order these nodes according to their degrees, and compare them with the degree sequence (both ordered in decreasing order). We process the node that requires more edges.

Let $\boldsymbol{d} = (d_1, d_2, \ldots, d_n)$ be the original degree sequence in decreasing order, let $\boldsymbol{a} = (a_1, \ldots, a_n)$ be the degrees already assigned in decreasing order, and let $a_{\sigma(i)}$ be the degree of node v_i (where σ is the appropriate permutation of the indices). Then, we consider at each step the node with maximum $d_i - a_i$. That is, $v_{\sigma^{-1}(i)}$ for $\arg\max d_i - a_i$. This follows the condition of Havel-Hakimi[21,22] (Proposition 1 above) for building a graph for a graphical sequence.

Let v_0 be the node selected. Then, the given node v_0 is linked to nodes v_i in a way inversely proportional to the distances dist(v_i, v_0). Only available nodes are considered. With available nodes we mean nodes that still need edges (ie, nodes v_i such that $(d_i - a_i) > 0$, and with $p(v_i, v_0) > 0$). At this step, $d_i - a_i$ different nodes are considered. Nevertheless, the algorithm can result into a multigraph when these edges are combined with previously established ones. That is, it is possible that two or more edges link the same two nodes.

The algorithm may result into a situation in which only one node exists with a nonzero degree. This can be solved with loops. Nevertheless, the algorithm does not generate loops, it terminates when such situation is detected. As we discuss in the experiment section, we run the algorithm several times if such case arises and as edges are randomly assigned, a new execution eventually solves the problem.

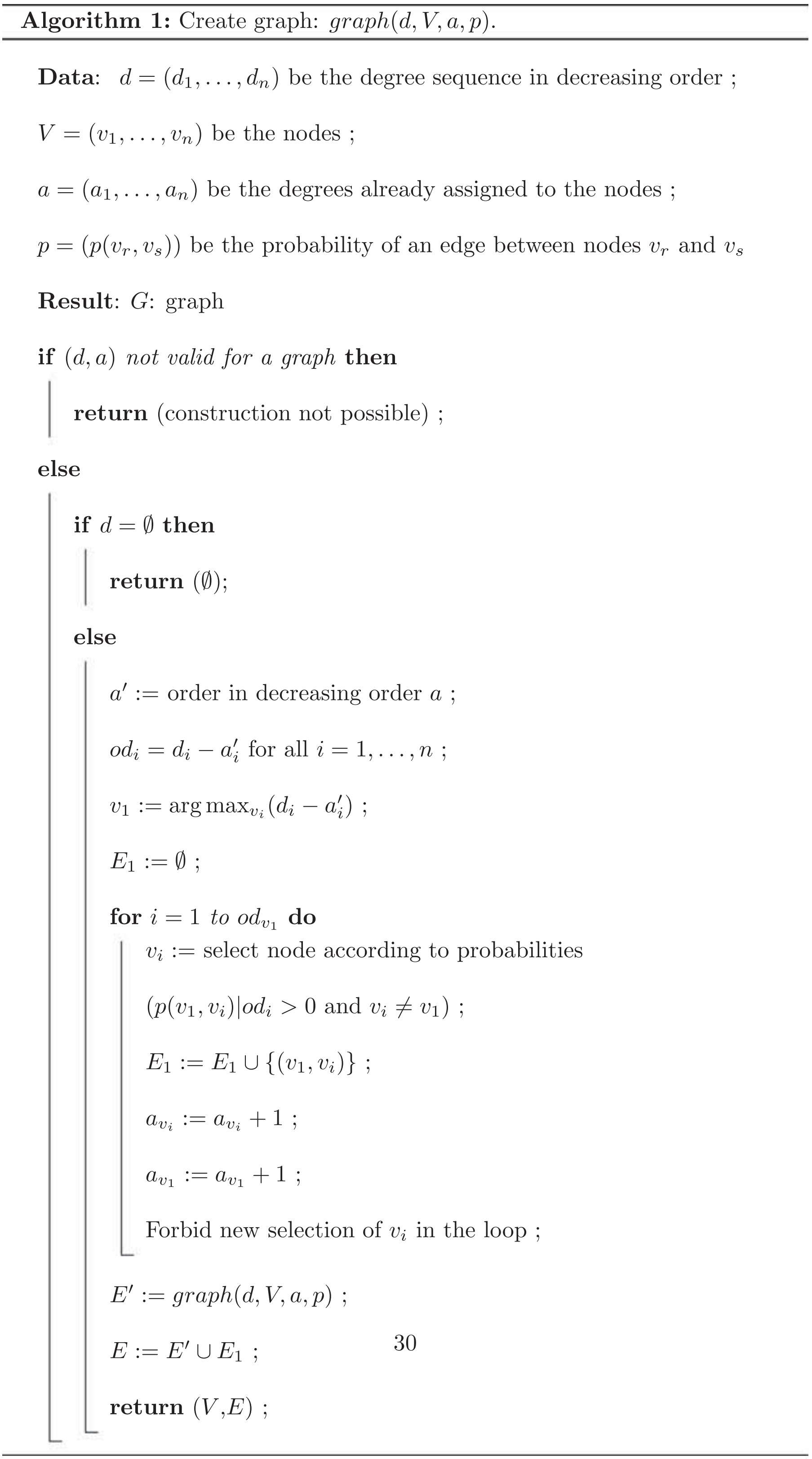
In contrast, as stated above, we allow multiple edges. We have implemented this greedy approach because the algorithm would otherwise need to backtrack if we have the constraint that nodes v_0 and v_i cannot be already connected. Backtracking becomes infeasible for large graphs.

This procedure is described in Algorithm 1. To remove multiple edges, we have implemented a new function to transform multigraphs into graphs, described in Algorithm 3.1 below. This is achieved by means of edge swaps. For each repeated or loop edge (v_1, v_2), the algorithm selects another edge (v_3, v_4) and replaces (v_1, v_2) and (v_3, v_4) by the edges (v_1, v_4) and (v_3, v_2). At this point, it is checked that the new edges are correct (not already present or a loop). Some theoretical considerations about the correctness of this transformation are given in Section 3.1.

Algorithm 1 has as parameters the degree sequence, the nodes V and the probabilities p. The probabilities are defined between pairs of nodes from their distances. We do so as follows, $p_1(x, y) = \text{maxDist} - \text{dist}(x, y)$ for all $x \neq y$ (and $p_1(x, x) = 0$). Then, $p(x, y) = f(p_1(x, y))$ for a given f so that $\sum_x \sum_y p(x, y) = 1$.

An alternative approach for establishing the edges is to use the Havel-Hakimi algorithm, or any of the algorithm for graph generation discussed in the introduction. Nevertheless, this alternative approach will lead to graphs where the spatial character is lost. That is, there is no guarantee that the probability of having an edge between two nodes is larger when these nodes are near than when they are far away, and the spatial setting is completely omitted. Therefore, our algorithm compares positively against this alternative way.

Create graph: graph(d, V, a, p).



Algorithm 1: Create graph: $graph(d, V, a, p)$.

```
Data: $d = (d_1, \ldots, d_n)$ be the degree sequence in decreasing order ;
$V = (v_1, \ldots, v_n)$ be the nodes ;
$a = (a_1, \ldots, a_n)$ be the degrees already assigned to the nodes ;
$p = (p(v_r, v_s))$ be the probability of an edge between nodes $v_r$ and $v_s$
Result: $G$: graph
if $(d, a)$ not valid for a graph then
    return (construction not possible) ;
else
    if $d = \emptyset$ then
        return $(\emptyset)$;
    else
        $a'$ := order in decreasing order $a$ ;
        $od_i = d_i - a'_i$ for all $i = 1, \ldots, n$ ;
        $v_1 := \arg\max_{v_i}(d_i - a'_i)$ ;
        $E_1 := \emptyset$ ;
        for $i = 1$ to $od_{v_1}$ do
            $v_i$ := select node according to probabilities
            $(p(v_1, v_i)|od_i > 0$ and $v_i \neq v_1)$ ;
            $E_1 := E_1 \cup \{(v_1, v_i)\}$ ;
            $a_{v_i} := a_{v_i} + 1$ ;
            $a_{v_1} := a_{v_1} + 1$ ;
            Forbid new selection of $v_i$ in the loop ;
        $E' := graph(d, V, a, p)$ ;
        $E := E' \cup E_1$ ;
        return $(V, E)$ ;
```

30

Code. The program has been developed and tested in R and it is available in http://www.ppdm.cat/links.php.

3.1 | Correctness of the algorithm

Above we have defined a procedure to transform a multigraph into a graph by means of swappings. Here we have studied the mathematical properties for this type of transformation.

The first issue is whether it is always possible to transform a multigraph with a given graphical degree sequence into a graph with the same degree sequence only by means of swaps of the multiple edges.

The first result obtained by a counterexample is that this is not always possible when the swap needs to generate two edges that are not already in the graph. Figure 1 (left) shows a graph with six nodes that is a multigraph (there are two edges between nodes v_3 and v_4) and whose degree sequence is graphical. Figure 1 (right) shows an example of graph with the same degree sequence. Note that there is no edge in the graph that we can swap with edge (v_3, v_4) without causing another multiedge.

Nevertheless, it is possible to transform the graph in Figure 1 (left) to the graph in Figure 1 (right) by means of swaps. To do so, we compute the difference between the two graphs. The difference is given in Figure 2. Then, we can proceed with the swapping of edges. A description of the transition from one graph to the other by means of swapping is given in Figure 3. Note that in this transition a swap can generate new multiedges.

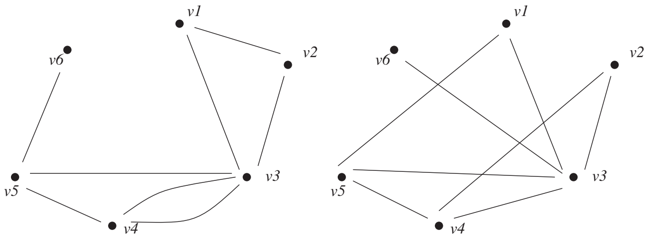


FIGURE 1 Multigraph (left) and graph (right) with degree sequence $d = (5, 3, 3, 2, 2, 1)$

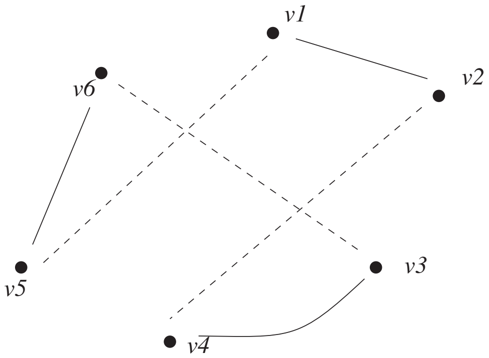


FIGURE 2 Difference between the multigraph in Figure 1 (left) and the graph in the same figure (right)

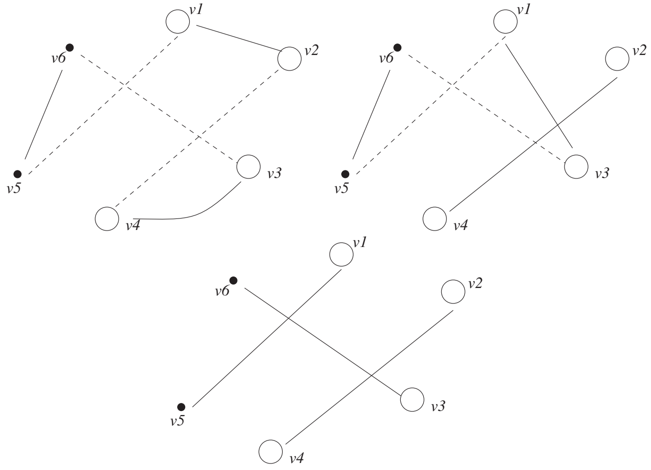


FIGURE 3 Steps for reaching the graph in Figure 1 (right) from the multigraph in the same figure (left)

Alternatively, we can swap (v_6, v_5) and (v_3, v_2) generating (v_6, v_3) and (v_5, v_2), and then swap (v_4, v_3) and (v_1, v_2) generating (v_4, v_1) and (v_3, v_2). Note that this transformation starts swapping an edge that is not a multiedge.

The next theorem proves that swapping is a valid procedure that permits to obtain a graph from a multigraph if the degree sequence is graphic.

Theorem 2 *Let G be a graph with degree sequence $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ and let H be a multigraph with the same degree sequence $\mathbf{d}$. Then, H can be modified into a graph (its multiedges may be removed) by a sequence of swaps such that no swap generates new multiple edges.*

Proof First we will prove that for any multiedge in H there is an alternating circuit C_1 such that complementing along C_1 removes the multiedge from H. Then, we will prove that complementing any alternating circuit can be done by a sequence of edge swaps that do not generate multiple edges.

Let uv be a multiedge in H. Since G is simple, it does not contain the uv edge, and as v has the same degree in G and in H, then G must contain an edge vv_1 that is not contained in H. Since $vv_1 \in E(G/H)$ and v_1 has the same degree in both G and H, then there is another edge $v_1v_2 \in E(H/G)$. Following the same argument for each edge $v_{2i-1}v_{2i} \in E(G/H)$, there is another edge $v_{2i}v_{2i+1} \in E(H/G)$. That is, this alternating path follows until it closes on a vertex u of odd degree in the path. Hence, it must be an alternating circuit $uvv_1 \ldots v_{2i}u$. By complementing along this circuit, we remove the edge uv and do not add multiedges to H. Following the same procedure for each multiedge, we obtain a simple graph.

To prove that complementing along an alternating circuit can be done by a sequence of swaps that do not create new multiple edges, we note that an alternating circuit is the union of alternating cycles and prove the theorem by induction on the length of the alternating cycles.

Suppose that the alternating cycle is $v_1v_2v_3v_4v_5v_6v_1$, that is, it contains only the edges v_1v_2, v_3v_4 and v_5v_6. If $v_1v_4 \notin E(G)$, then we may swap v_1v_2 with v_4v_3 to obtain the edges

v_1v_4 and v_1v_2, and then swap the edges v_1v_4 with v_6v_5, obtaining the edges v_1v_6 and v_4v_5, which is equivalent to complementing along the cycle $v_1v_2v_3v_4v_5v_6v_1$. Otherwise, if $v_1v_4 \in E(G)$, to avoid creating multiple edges, we swap v_1v_4 with v_6v_5 before and then swap v_1v_2 with v_4v_3, obtaining the same result as complementing along the cycle $v_1v_2v_3v_4v_5v_6v_1$.

In general, if we have an alternating cycle $v_1v_2...v_{2n-1}v_{2n}v_1$ with $n > 3$. In the case that $v_2v_{2n-1} \notin E(G)$ by swapping the edges v_1v_2 and $v_{2n}v_{2n-1}$, we obtain the edge $v_{2n}v_1$ and an alternating cycle $v_{2n-2}v_1v_2...v_{2n-3}v_{2n-2}$, that can be obtained by swappings (without creating multiple edges) by induction hypothesis. In the case that $v_2v_{2n-1} \in E(G)$, the alternating cycle $v_{2n-2}v_1v_2...v_{2n-3}v_{2n-2}$, can be obtained by swappings (without creating multiple edges) by induction hypothesis, then swapping the edges v_1v_2 and $v_{2n}v_{2n-1}$. In both cases, we obtain the same result as complementing along the cycle $v_1v_2...v_{2n-1}v_{2n}v_1$. Therefore, the multigraph H can be modified into a graph by a sequence of swaps that do not generate multiple edges. □

We have proved that the multigraphs with graphical degree sequence can always be modified into graphs by a sequence of swaps. This approach is effective and optimal as the number of swaps is minimal. Nevertheless, it requires the comparison of multigraph H and graph G. Hence, in the case of having only the multigraph H, a graph G with the same degree sequence as H must be obtained before being able to apply it.

There are other sufficient conditions to guarantee that a multigraph can be modified into a graph. For example, it is easy to prove that if there is a vertex w at distance greater than 2 from the multiedge uv in H, then we can take an edge wz that is at distance at least two from both vertices u and v and swap it with the edge uv, hence removing the edge uv without creating another multiedge.

When for a multiedge uv there is no vertex w at a distance larger than 2, approaches as the ones described in Theorem 2 are needed. For large enough graphs, and in particular if they are scale-free, it is possible to find such vertex at distance larger than 2 and remove multiedges.

In general, Theorem 2 permits to build a simple heuristic program that randomly swaps multiedges. The procedure is an iterative process that interleaves one swap between a randomly selected pair of edges, and one swap between a multiedge and a randomly selected edge. This is formalized in Algorithm 3.1.

Transform multigraph H into a graph:

Algorithm 2: Transform multigraph H into a graph:

```
while $H$ is still a multigraph do
    Randomly select edges $v_i$ and $v_j$ and swap them ;
    Randomly select multiedge $v_r$ and edge $v_s$ and swap them ;
return ($H$) ;
```

These swaps are enough to modify the multigraph H into a graph G. This is observed in the experiments presented below. In fact, in the experiments performed, we can even proceed without swapping edges that are not multiedges. That is, we can skip the first statement of the while loop. Nevertheless, as proven above with the example of Figure 1, in the most general case, the first statement is required.

3.2 | Costs

As a summary, the proposed algorithm consists on the three steps described at the beginning of Section 3 and the fourth step described in Algorithm 3.1 for transforming the multigraph into a graph.

The cost of the algorithm to generate the multigraph is $O(\max(n^2 \log n,\ m^2))$ where n is the number of nodes and $m \times m$ is the dimension of the underlying map. We detail the cost of the different steps of the algorithm.

- The generation of the underlying map with dimensions $m \times m$ corresponds to the cost of the FFT, which is $O(m^2 \log(m^2))$. Note that the generation of X, matrix H and the computation of $F^{'}$ is $O(m^2)$.
- The generation of the n nodes has cost $O(m^2)$ as n random values are produced and then need to be located in the $m \times m$ normalized matrix. At the same time that nodes are generated the matrix of distances between pairs of nodes are computed. Assuming $n < m$, the generation of the matrix of distances has cost $O(n^2) \leq O(m^2)$. In general, we have that the cost is $O(\max(n^2,\ m^2))$.
- Next, we generate e edges for the n nodes. This is done iteratively for each node. At each step, we need to order the list of missing edges (at most n values). Therefore, this ordering step has cost $O(n \log n)$. We select for each node an average of v/n edges, although this number is not much relevant. Edges are generated using v/n random numbers that are checked against a distribution built for all other $n-1$ nodes. To build the distribution and select the nodes, we need $O(n)$. As the ordering and the generation and selection of edges is done for every node, the overall cost is $(n^2 \log n)$.

We do not have an exact bound for the algorithm to transform the multigraph (if there are loops and multiedges) to a graph. This would depend on the number of edges already present and the difficulty of randomly constructing a valid edge when a valid edge and a repeated one are swapped.

An upper bound obtained directly from Theorem 2 for Algorithm 3.1 is $e^{'}(L/2 - 1)$ where $e^{'}$ is the number of repeated edges and L is the length of the largest alternating circuit in the graph. In the experiments presented below, the number of swaps needed is less than this bound. We discuss the performance of the approach for a few graphs in the next section.

4 | EXPERIMENTS

We have considered the generation of graphs for the degree sequences of the datasets in Table 1. The size of the graphs range from 34 to 1133 nodes (denoted as N. Nodes), and from 78 to 5451 edges (denoted as N. Edges). For each graph, we also give the average degree (denoted as M. Degree) and the standard deviation of the degrees of the nodes (denoted as σ(degree)).

TABLE 1 Description of the datasets considered in our experiments

Name	N. Nodes	N. Edges	M. Degree	σ(degree)	N. Complete	Density
Jazz[39]	198	2742	27.7	17.5	19503	0.140
Karate[40]	34	78	4.6	3.9	561	0.139
Football[41]	115	613	10.7	0.9	6555	0.093
Erdos971[42]	472	1314	5.6	6.7	111156	0.011
Urvemail[43]	1133	5451	9.6	9.3	641278	0.008
CElegans[44]	453	2025	8.9	16.7	102378	0.019

For each set, we give the number of nodes, the number of edges, the mean degree, the standard deviation, the number of edges for a complete graph, and the degree of completeness (i.e., number of edges/number possible edges).

TABLE 2 Average results for 10 executions in seconds

Name	Times multigraph	No of multiedges	Steps	Time
Jazz[39]	10/10	14.9	26.5	5.84
Karate[40]	0/10	0	0	0.26
Football[41]	10/10	3.9	4.8	0.95
Erdos971[42]	7/10	1.2	1.2	7.92
Urvemail[43]	10/10	3.0	3.5	102.83
CElegans[44]	1/10	0.1	0.1	9.97

The number of edges if the graphs were complete (ie, $n^*(n-1)/2$ where n is the number of nodes), and the proportion of edges with respect to a complete graph (ie, $|E|/(n^*(n-1)/2)$) with the same number of nodes is also given in the table (denoted as N. Complete and Density, respectively).

We have used Algorithm 3.1 to remove the multiedges. Each algorithm has been applied 10 times to compute averages.

The maximum number of multiedges has been obtained for the Jazz data set, and it was 23. In fact, this is the data set that gets on average the maximum number of multiedges. It is also the graph with higher density. Nevertheless, Karate with almost the same density has not generated graphs with multiedges. Table 2 summarizes the number of times we have obtained a multigraph and the number of multiedges obtained. We also display the number of steps required to remove these multiedges using the second algorithm, and the average time (in seconds) to build the graph for each data set. This time is the average time of building the map and applying both the first and the second algorithms. It can be seen from the table that the average number of steps is less than twice the number of multiedges. In one of the tests, it was slightly larger (with $r = 20$ multiple edges and 47 steps in the Jazz case). Figures 4 and 5 display two examples of graphs generated from the Karate data set. Figure 4 uses as its underlying map a graph generated with our map generation function. We applied for this example the approach based on the FFT and the filtering matrix that approximates a Gaussian distribution. The other approach for generating a map leads to similar results. Figure 5 uses as its underlying map the map of Catalonia and a function of its population density. The straightforward use of the

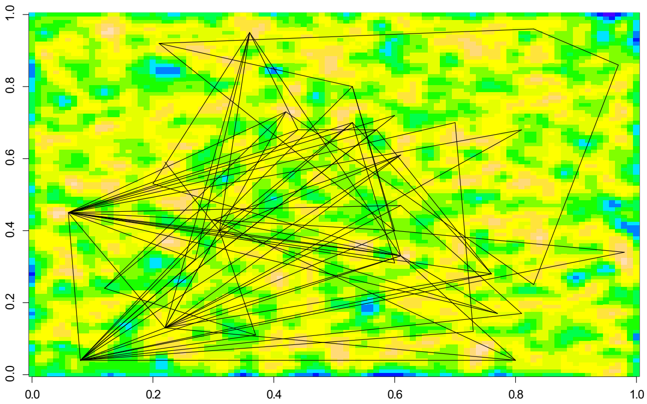


FIGURE 4 Graph built from karate data set using as its underlying map a random map [Color figure can be viewed at wileyonlinelibrary.com]

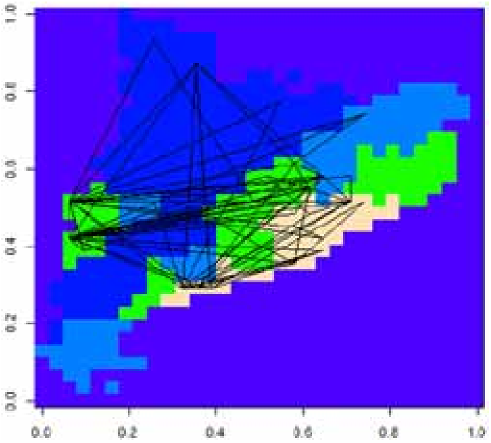


FIGURE 5 Graph built from karate data set using as its underlying map a map of Catalonia and its population density [Color figure can be viewed at wileyonlinelibrary.com]

population density assigns almost all nodes to the capital. See for details http://www.ppdm.cat/links.php.

5 | CONCLUSIONS AND FUTURE WORK

In this paper, we propose a method for the generation of graphs in which nodes are located according to an underlying probability distribution. We have also shown how to generate these distributions when the whole graph is synthetic. Future work includes the development of other algorithms that include some additional constraints into the synthetic graph, for example, adding constraints on centrality,[45] girth, diameter, etc.

We have proven in Theorem 2 that for a given degree sequence it is always possible to transform a multigraph into a graph by means of swaps. Our heuristic approach based on Theorem 2 forces that all graphs in this path do not have multiedges. In the experiments considered, this heuristic approach is shown to be valid and computationally feasible. A probabilistic analysis of this approach would be a valuable future analysis.

The application of the FFT in the construction of the map from an initial random map makes the graph smooth, something that it is required for completely synthetic graphs. This approach can also be applied for anonymization of geolocalized maps which would be a useful future development of the algorithm.

The construction of a partially synthetic graph starts with the map. Then, when a given region has a high density, the probability that several nodes are assigned to that region is high. This also implies that there will be links between the nodes of the region. This approach can be used to generate community structures. This is left to future work.

ACKNOWLEDGMENTS

Julián Salas acknowledges the support of a UOC postdoctoral fellowship. This study was partially supported by Swedish VR, Swedish Research Council, Sweden (project VR 2016-03346), and Spanish MINECO, Ministry of Economy and Enterprise, Spain (projects TIN2014-57364-C2-2-R SMARTGLACIS and TIN2014-55243-P).

REFERENCES

1. Barabasi A-L, Albert R. Emergence of scaling in random networks. *Science*. 1999;286:509-512.
2. Csardi G, Nepusz T. The igraph software package for complex network research. *Int J Complex Syst*. 2006;1695(5):1-9. https://doi.org/igraph.sf.net
3. Bender E, Canfield E. The asymptotic number of labelled graphs with given degree sequences. *J Comb Th A*. 1978;24:296-307.
4. Bollobas B. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *Euro J Combin*. 1980;1:311-316.
5. Jerrum M, Sinclair A. Fast uniform generation of regular graphs. *Theoret Comput Sci*. 1990;73:91-100.
6. Jerrum M, McKay BD, Sinclair A. When is a graphical sequence stable? In: Frieze A, Luczak T, eds. *Random Graphs*. 2. New York: Wiley-Interscience; 1992:101-115.
7. Salas J, Torra V. Improving the characterization of P-stability for applications in network privacy. *Discrete Appl Math*. 2016;206:109-114.
8. Salas J, Torra V. Graphic sequences, distances and k-degree anonymity. *Discrete Appl Math*. 2015;188:25-31.
9. Gkantsidis C, Mihail M, Zegura E. The markov chain simulation method for generating connected power law random graphs., 2003 16-25.
10. Viger F, Latapy M. Fast generation of random connected graphs with prescribed degrees. 2005; arXiv:cs/0502085
11. Ying X, Wu X. Graph generation with prescribed feature constraints. Proceedings of the 2009 SIAM International Conference on Data Mining. April 2009. 966-977, Reno-Sparks-Tahoe, NV.
12. Cooper C, Dyer M, Greenhill C. Sampling regular graphs and a peer-to-peer network. *Comb Prob Comp*. 2007;16(4):557-593.
13. Kannan R, Tetali P, Vempala S. Simple markov-chain algorithms for generating bipartite graphs and tournaments. Proceeding of ACM Symposium on Discrete Algorithms. January 1997. 193–200, New Orleans, LA.
14. Miklós I, Erdös PL, Soukup L. Towards random uniform sampling of bipartite graphs with given degree sequence. *Electr J Comb*. 2013;20(1):P16.
15. Milo R, Kashtan N, Itzkovitz S, Newman MEJ, Alon U. On the uniform generation of random graphs with prescribed degree sequences.2004; arXiv:cond-mat/0312028v2.
16. Britton T, Deijfen M, Martin-Löf A. Generating simple random graphs with prescribed degree distribution. *J Statist Phys*. 2006;124:1377.
17. Aiello W, Chung F, Lu L. A random graph model for massive graphs. Proc. STOC 2000; 171-180, Portland, OR.
18. Molloy M, Reed B. A critical point for random graphs with a given degree sequence. *Random Struct Algor*. 1995;6(2-3):161-180.
19. Molloy M, Reed B. The size of the giant component of a random graph with a given degree sequence. *Comb Probab Comput*. 1998;7:295-305.

20. Erdös P, Gallai T. Graphen mit punkten vorgeschriebenen grades. *Mat Lapok*. 1960;11:264-274.
21. Havel V. A remark on the existence of finite graphs. *Časopis ProPěstovánímatematiky*. 1955;80:477-480.
22. Hakimi SL. On realizability of a set of integers as degrees of the vertices of a linear graph I. *J SIAM*. 1962;10:496-506.
23. Casas-Roma J, Herrera-Joancomartí J, Torra V. Anonymizing graphs: Measuring quality for clustering. *Knowledge and Information Systems*. 2015;44(3):507-528.
24. Nagle F. Privacy breach analysis in social networks. In: Özyer T, Erdem Z, Rokne J, Khoury S, eds. *Mining social networks and security informatics, lecture notes in social networks*. The Netherlands: Springer, 2013:63-77.
25. Nettleton DF. Generating synthetic online social network graph data and topologies. 3rd Workshop on Graph-based Technologies and Applications (Graph-TA), March 2015, Barcelona.
26. Drechsler J. *Synthetic datasets for statistical disclosure control: Theory and implementation*. New York: Springer; 2011.
27. Stokes K, Torra V. Reidentification and k-anonymity: A model for disc losure risk in graphs. *Soft Comput*. 2012;16(10):1657-1670.
28. Torra V, Nin J. Record linkage for database integration using fuzzy integrals. *Int J Intel Sys*. 2008;23:715-734.
29. Samarati P, Sweeney L. Protecting privacy when disclosing information: *k*-anonymity and its enforcement through generalization and suppression. SRI Intl Tech Rep No. SRI-CSL-98-04. 1998; Menlo Park, CA: SRI International.
30. Liu K, Terzi E.Towards identity anonymization on graphs. Proceedings of SIGMOD 2008. 2008; 93-106, Vancouver, Canada.
31. Campan A, Truta TM. Data and structural k-anonymity in social networks. *Lect Notes Comput Sci*. 2009;5456:33-54.
32. Hay M, Miklau G, Jensen D, Towsley D, Weis P. Resisting structural reidentification in anonymized social networks. *Proc VLDB*. 2008;1(1):102-114.
33. Stokes K, Torra V. On some clustering approaches for graphs. Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011) (ISBN 978-1-4244- 7315-1), Taipei, Taiwan. 2011; 409-415.
34. Torra V, Shafie T, Salas J. Data protection for online social networks and P-stability for graphs. *IEEE Trans Emerging Top Comput*. 2016;4(3):374-381.
35. Barthélemy M. Spatial networks. *Phys Rep*. 2011;499(1-3):1-101.
36. Blitzstein J, Diaconis P. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Math*. 2011;6(4):489-522.
37. Håkanson N, Jonsson A, Lennartsson J, Lindström T, Wennergren U. Generating structure specific networks. *Adv Complex Sys*. 2010;13(2):239-250.
38. Lennartsson J, Hąnsson U, Wennergren U, Jonsson A. SpecNet: A spatial network algorithm that generates a wide range of specific structures. *PLOS One*. 2012;7(8):e42679.
39. Gleiser P, Danon L. Community structure in jazz. *Adv Complex Sys*. 2003;6(4):565-573.
40. Zachary WW. An information flow model for conflict and fission in small groups. *J Anthropol Res*. 1977;33:452-473.
41. Girvan M, Newman MEJ. Community structure in social and biological networks. *Proc Natl Acad Sci USA*. 2002;99(12):7821-7826.
42. Networks/pajek, package for large network analysis. http://vlado.fmf.uni-lj.si/pub/networks/pajek/data/gphs.htm
43. Guimera R, Danon L, Diaz-Guilera A, Giralt F, Arenas A. Self-similar community structure in a network of human interactions. *Phys Rev E*. 2003;68:065103(R). http://deim.urv.cat/ aarenas/data/welcome.htm
44. Duch J, Arenas A. Community identification using extremal optimization. *Phys Rev E*. 2005;72:027-104.
45. Grassi R, Stefani S, Torriero A. Centrality in organizational networks. *Int J Intel Sys*. 2010;25:253-265.

How to cite this article: Torra V, Jonsson A, Navarro-Arribas G, Salas J. Synthetic generation of spatial graphs. *Int J Intell Syst*. 2018;33:2364-2378. https://doi.org/10.1002/int.22034