Journal of Physics:
Complexity

Large Network Generator: a simple, efficient, and flexible graph formation algorithm

Journal of Physics: Complexity

OPEN ACCESS

RECEIVED
18 January 2026

REVISED
29 April 2026

ACCEPTED FOR PUBLICATION
15 May 2026

PUBLISHED
27 May 2026

PAPER

Large Network Generator: a simple, efficient, and flexible graph formation algorithm

João Pedro C Morais[1], **Celso C Ribeiro**[2] and **Ruben Interian**[1,*]

[1] Institute of Computing, University of Campinas (Unicamp) Campinas, SP 13083-852, Brazil
[2] Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-240, Brazil
[*] Author to whom any correspondence should be addressed.

E-mail: ruben@ic.unicamp.br and celso@ic.uff.br

Keywords: real-world network, scale-free network, random walk, small-world, clustering coefficient

Abstract

This study introduces the Large Network Generator, an algorithm capable of creating undirected graphs with three main characteristics of real-world networks: long-tailed degree distribution, short distances between nodes (small-world phenomenon), and large clustering coefficients. The main idea is to add one node to the network at each step, perform random walks on the existing nodes, and select a subset of marked nodes to connect to the new node. Additionally, with an adjustable probability, edges may be created between the marked nodes themselves. Key advantages of our algorithm are its simplicity, efficiency, and flexibility in creating networks with different characteristics without using global information about network topology. The parameters can be adjusted to generate networks with specific characteristics, producing a wide range of values for each parameter, including high clustering (up to $\overline{C} \approx 0.7$), markedly short average distances (as low as $\overline{L} \approx 2.8$ for $N = 200\,000$), and a strong presence of hubs. Additionally, the algorithm's near-linear time complexity allows the creation of networks with one million nodes in less than 60 s. The implementation of our algorithm is publicly available on a GitHub repository.

1. Introduction

The study of complex networks has gained prominence across various scientific disciplines, such as sociology, physics, biology, and computer science [1–5]. Real-world networks often exhibit both scale-free characteristics and surprising proximity among network nodes. These features commonly observed in real-world networks are the so-called Matthew effect and the small-world phenomenon.

The Matthew effect (also known as 'rich-get-richer effect', or accumulated advantage) reflects a preferential attachment dynamic [6], leading to the emergence of high-degree hubs in a network and long-tailed degree distributions [7]. This behavior can be easily observed in social networks, where most people have few connections, and few people hold a very large number of connections, showing a long-tail pattern [8].

Social networks are also examples of networks with high clustering, where individuals tend to form tightly connected groups, such as friend circles, family groups, or professional communities. In these clusters, the likelihood of any two friends of some individual also being friends is higher than in random networks, resulting in a high clustering coefficient.

Finally, the small-world property characterizes networks with short average path lengths. Collaboration networks of actors and the neural network of the nematode worm *C. elegans* are examples of small-world networks [9].

Modeling networks with these three characteristics can be challenging, as the mechanisms underlying each of them tend to drive network structures in different directions. Random walk models [10] were previously used to generate networks with long-tailed degree distributions, capturing real-world systems'

natural growth and preferential attachment. A key advantage is their ability to create structures without global information, aligning with the organic development of real networks. While effective at generating hubs and modeling the Matthew effect, traditional random walks often overlook reducing path lengths between nodes (which are a crucial feature of small-world networks), and generate large clustering coefficients. These limitations make it challenging to model real systems that have long-tailed degree distributions, high clustering, and short average distances.

In this paper, we present the Large Network Generator: a simple, intuitive, and efficient random walk algorithm that generates small-world networks exhibiting high local clustering and long-tailed degree distributions. It does not require any global information about the entire network, such as the node degrees or their coordinates in some Euclidean space. The algorithm represents a significant improvement over the one proposed by Morais and Interian [11]. Our algorithm is efficient, i.e. linear in the number of network nodes, and flexible, generating networks with different clustering coefficients and a range of average distances between nodes. Additionally, we provide the full implementation in a publicly accessible GitHub repository [12]. The Large Network Generator's PyPI package is also available [13].

The paper is organized as follows. Section 2 presents previous works that inspired and contributed with ideas that led to the approach proposed in this work. Section 3 describes the new algorithm. The results and characteristics of the generated networks are presented in section 4. The conclusions are detailed in the last section.

2. Previous works

Previous works have studied the generation of networks with some specific combinations of the features considered in this study.

Arguing that real-world networks are often highly clustered while showing small average distances between nodes, Watts and Strogatz [9] proposed a model to reproduce these characteristics. Starting with a set of N nodes in circular order, where each node is connected by undirected edges to k neighbors, the authors proposed an algorithm rewiring each edge with some fixed and small probability p. The average clustering coefficient remained quite high, while the average distance dropped to a small value approximately proportional to $\log N$.

Latent-space network models have also been employed to investigate small-world networks with non-vanishing clustering [14]. In its simplest version, also called random geometric graph model, nodes are distributed uniformly at random in some metric space, and two nodes i and j are connected if and only if the distance x_{ij} between them is less than some parameter μ, leading to high clustering and large average shortest paths. However, the model can become a small-world by introducing a probability p_{ij} of the existence of a link between the nodes. For example, in $\mathbb{R}^d$ space, choosing $p_{ij} \propto x_{ij}^{-\beta}$ with $\beta \in (d, 2d)$ results in non-vanishing clustering coefficients and small-world networks, with average distances scaling proportionally to $\log N$ [15].

On the other hand, real-world networks also have high-degree nodes called hubs, which are absent in the above models. Barabási and Albert [6] proposed a preferential attachment process that generates such long-tailed degree distributions (BA model). Starting from some small graph, one new node v is added at each iteration with k new edges linking v to k different nodes chosen with probabilities proportional to the node degrees. That is, the likelihood of node v choosing node w is proportional to the degree of w at that iteration, generating scale-free networks.

To increase the clustering coefficient of the networks generated by the above BA model, Holme and Kim [16] introduced a triad formation step performed with probability P_t after a node and its edges are added to the network. When a new edge is added linking the new node v to a node w, an edge is added linking v to a randomly selected neighbor of w, thus creating a triad between the three nodes and increasing the clustering of the network.

In the same context, Klemm and Eguíluz [17] proposed a memory-based growth model in which nodes are either active or inactive, and new connections are made exclusively to the active ones. At each time step, a new active node is added to the network, and one of the currently active nodes is deactivated with a probability inversely proportional to its degree—modeling a form of collective forgetting, where highly connected ('famous') nodes are less likely to be forgotten. As a result, nodes are unlikely to remain active and, therefore, grow older unless they have a high degree. The model is able to generate networks with a power-law degree distribution and high clustering coefficients.

Although previous models have presented solid results in scale-free network construction, a significant concern we raise is that they require global information at each step (e.g. the degrees of all nodes to

calculate the preferential attachment probabilities, or the coordinates of all nodes), which may be unrealistic since, in real-world networks, links emerge naturally, without global information about the network topology.

A different approach was proposed by Davidsen *et al* [18]. Their model maintains a constant population of nodes, each connected by undirected links representing mutual acquaintance. At each time step, a randomly selected individual introduces two of their acquaintances to each other, forming a new link. This 'transitive linking' mechanism leads to the formation of triangles. Additionally, with a small probability, one node is removed from the network along with all its connections and replaced by a newcomer who starts with a single random acquaintance. These two alternating steps, triangle formation and node turnover, drive the network toward a stationary state. The resulting networks exhibit both high clustering and short average path lengths, and, for some parameter values, they also may have power-law degree distributions. However, Davidsen *et al* [18] do not investigate the number of steps required for the network to reach the stationary state. This represents a significant limitation for larger networks, where convergence could be slow or computationally expensive, and where a nonlinear time complexity may be impractical.

Saramäki and Kaski (S–K) [10] used random walkers to generate undirected scale-free networks. They showed that it is not necessary to have global information about node degrees at each step to achieve these results. Herrera and Zufiria [19] improved this process by using the number of steps in the random walks to guide triangle generation, introducing a way to control the network's clustering coefficient using only local information.

The random walk algorithm proposed by Saramäki and Kaski [10] begins from a typically small initial graph with n_0 nodes. At each iteration, a new node v is added to the graph, linking v to existing nodes chosen using random walks. These chosen nodes (called 'marked nodes' from now on) are identified as follows: beginning from a randomly selected node w, ℓ random steps are taken from w, allowing to revisit previous nodes. After each walk, the endpoint is marked, and the process continues until m nodes are marked. The new node v is then connected to the marked nodes. The algorithm terminates after adding N new nodes to the graph.

Herrera and Zufiria [19] noted that by changing the value of the parameter ℓ, the number of steps in the random walk, it is possible to control the network's clustering coefficient. If $l = 1$, the neighbor of a marked node will also be marked, generating a triangle between these two neighbors and the added node, thus affecting the clustering coefficient. Each node v has an associated value p_v, the probability of $l = 1$ if the random walk starts from that node.

To reduce the average distances in the random walk networks, Morais and Interian [11] introduced a new phase of the algorithm after the random walk, adding a special edge at the end of each iteration. After the random walk marking and linking is finished, a random node s of the network is chosen, a distance value d is selected based on some probability distribution, and then another node t with distance d from s is linked to s. With this simple enhancement, they were able to significantly reduce the average shortest path lengths in the networks, while keeping the clustering coefficients' high values and the scale-free properties of the previous models. However, there are downsides to their approach: the step of finding another node at a distance exactly d from s could make their algorithm computationally costly, and they have not studied the controllability of the parameters of the generated networks.

Random walks controlling the walk length ℓ proved an efficient way to create power-law networks while regulating the clustering coefficient. Nevertheless, the efficient and flexible generation of large realistic networks [20] that simultaneously exhibit all three main features (small-world, high local clustering, and long-tailed degree distributions), while keeping the algorithm time complexity low and relying only on local information, remains a challenge, justifying the search for innovative algorithms. Relying only on local information is crucial, since the mechanisms underlying the formation of real-world networks never depend on the global structure of the entire network. These gaps will be addressed in the next section.

3. Network generation algorithm

The previous algorithm by Morais and Interian [11] managed to reduce the average distances in the networks generated by the algorithm of Herrera and Zufiria by introducing a new step, in which an edge is added linking two random nodes with distance d from each other, where d is drawn from a probability distribution. The step-by-step description of the algorithm is given below:

1. Start with a small graph $G(V, E)$ with n_0 nodes.

2. Create a new, disconnected node v and add it to the graph.
3. Pick a random node $w \in V, w \neq v$. Perform a random walk from w, marking each node reached after every ℓ steps. Stop when m nodes are marked, and connect them to v.
4. Choose an integer $d > 1$ with some probability $P(d)$ and a random node s. Find a node t at a distance d from s, and connect s and t with an edge.
5. Repeat N times steps 2 to 4.

Although the new edge reduces distances, finding two nodes that are exactly at distance d is computationally expensive, making the overall algorithm quadratic in the number of nodes N. More precisely, the time complexity is $O(N^2)$, because the node-finding step runs in linear time, and is executed exactly N times, which can be a limitation when generating large networks.

Additionally, the extra step that adds the new edge is heterogeneous with respect to the algorithm's core. The new edge is unrelated to the random walk, making the overall procedure inconsistent or artificial.

More importantly, the extra edge decreases the clustering coefficient. Carelessly connecting distant nodes in the network can open new triads that may remain unclosed, thus limiting the maximum clustering coefficients measured in networks generated by the algorithm.

3.1. Description of the new algorithm

The new Large Network Generator algorithm proposed in this work starts with a small initial graph $G = (V, E)$ with n_0 nodes. At each iteration, a new node v is added to the graph, and a random walk is performed, marking one node every ℓ steps until m nodes are marked. Then, all the marked nodes are connected to the new node v.

Suppose now that ℓ can take values between 1 and k, based on a truncated geometric distribution. For example, if $k = 10$:

$$P(l = x) = \frac{p\,(1-p)^{x-1}}{1-(1-p)^{10}}.$$

We can easily control the clustering coefficient of the generated network through the parameter p.

Moreover, the average distances are reduced without creating additional edges. Although the probability of obtaining a large value of ℓ is low, a small fraction of the walks will reach distant nodes, and the new node v will act as a shortcut that reduces the average distances. This rationale motivated the choice of the decreasing truncated geometric distribution, but other decreasing distributions may also be considered.

We introduce a new step to the algorithm, connecting pairs of marked nodes with probability f_p, a new parameter that represents friendship probability. This step can be interpreted straightforwardly. Friends of a common friend, the newly added node v, may form connections among themselves. This follows the *Triadic Closure* [21–24] principle, a feature of real-world networks that states that if two nodes are both connected to a third one, there is an increased likelihood that they will also become directly connected. These new connections between marked nodes can emerge in the future, as the triadic closure principle suggests. In our implementation, we generate these links when the new node is added.

The step-by-step of the **Large Network Generator** algorithm is the following:

1. Start with a small graph $G(V, E)$ with n_0 nodes.
2. Perform steps 3 to 5 a total of N times.
3. Pick a random node $w \in V$, and perform a random walk from w. Mark a node at every ℓ steps, each time choosing a new value of $\ell \in \{1, \ldots, k\}$ from some probability distribution P. Stop when m nodes have been marked.
4. Add a new node v to the graph. Connect each marked node to v.
5. For each pair of marked nodes, connect them with probability f_p.

Figure 1 shows a flowchart of the proposed procedure, highlighting its key components. The diagram intuitively illustrates the full workflow and key components of the proposed approach, including

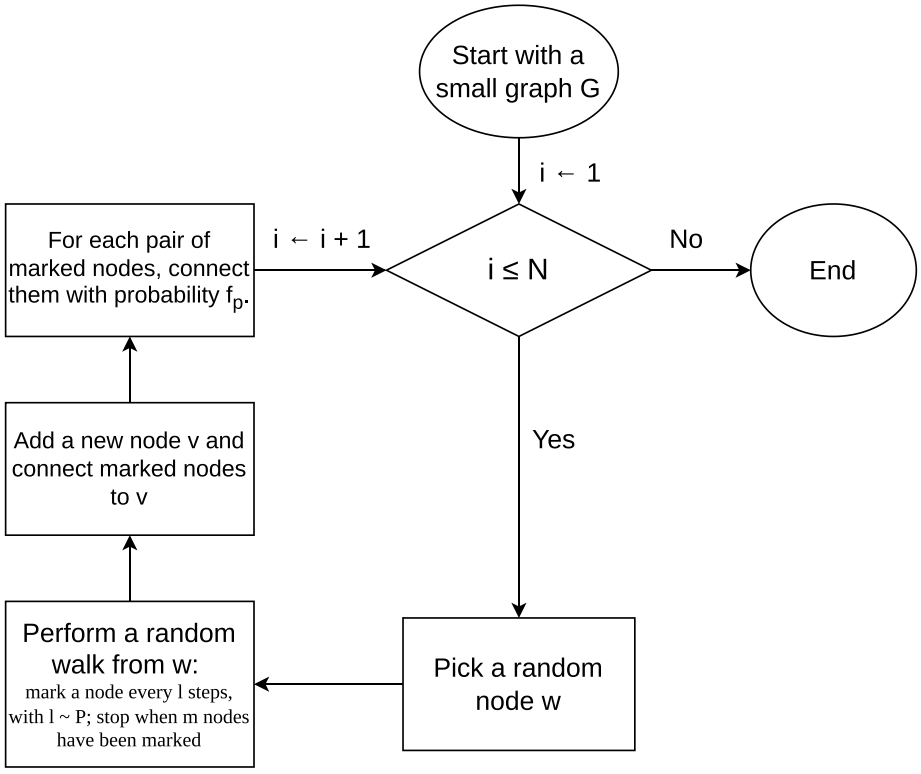

Figure 1. The flowchart diagram of the proposed *Large Network Generator* algorithm.

the main loop, the random walk, and the addition of edges between marked nodes with an adjustable probability.

The steps of the algorithm are also illustrated in figures 2–4. In figure 2, the random walk in step 3 marks the nodes in yellow, following the path indicated by the arrows. In figure 3, the new node is added and linked to the nodes marked in the previous step. Finally, in figure 4, edges are created between the marked nodes with probability f_p (step 5).

In summary, the algorithm consists of two main phases, which can be interpreted as follows. First, a new node is added to the network and connected to a set of existing nodes, which can be seen as a new person joining a community and forming a few initial acquaintances. Second, with a given probability, links are created among these neighbors of the new node, just as this person might introduce their acquaintances to one another.

3.2. Specification of the Large Network Generator

Algorithms 1 and 2 describe the implementation of the Large Network Generator in more detail.

Algorithm 1 describes the random walk process, while algorithm 2 corresponds to the network generation. In line 1, the RandomWalk algorithm initializes the list of marked nodes with the starting node. In line 2, it sets the *current* variable to track the current position during the walk, initially set to *start*. The main loop (lines 3–8) continues until the number of marked nodes reaches m. In each iteration of the loop, line 4 samples an integer value ℓ from a given probability distribution P over the set $\{1,\ldots,k\}$ (for example, the truncated geometric distribution), which determines how many steps the walk will take in the current phase. Lines 5–7 perform ℓ random walk steps: In each step, the algorithm retrieves the neighbors of the current node (line 6) and updates the current node by randomly choosing one of its neighbors (line 7). After completing the ℓ steps, the final node reached is added to the list of marked nodes (line 8). Once m nodes have been marked, the algorithm returns the list of marked nodes in line 9.

Algorithm 2 outlines the network generation process. It takes as input an initial graph G, typically small and connected, the number of nodes N to be added, a probability distribution P used by the random walk, the number of edges m to be added from each new node, and a probability f_p that controls the addition of edges between selected neighbors.

The algorithm iteratively repeats the following steps N times: In line 2, it selects a random node from the current graph as the starting point of the random walk. In line 3, it performs a random walk using

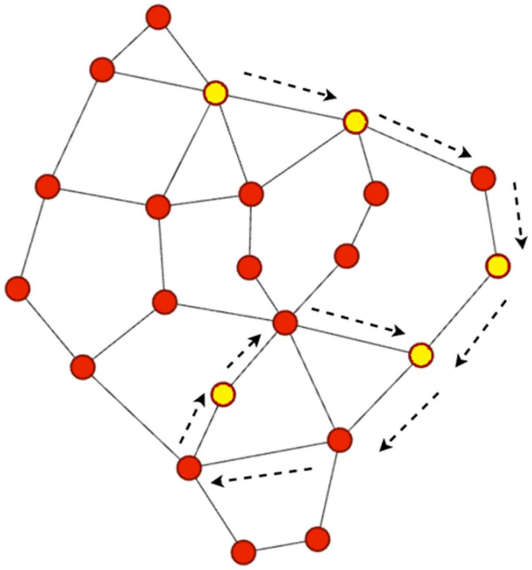

Figure 2. The random walk, following the arrows, generates a set of yellow-marked nodes (step 3 of the algorithm).

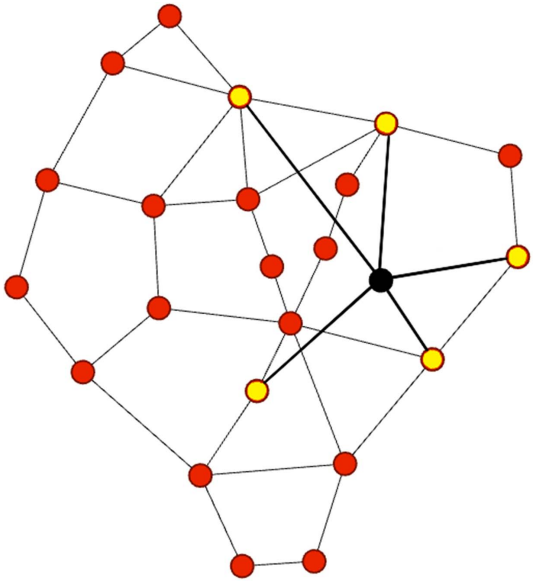

Figure 3. A new node is added to the graph, connecting this node to the yellow-marked nodes (step 4 of the algorithm).

the **RandomWalk** procedure (algorithm 1), which returns a set of m marked nodes that will serve as the attachment points. Line 4 adds a new node v to the graph. Then, in lines 5–6, the algorithm connects v to each of the marked nodes returned by the random walk, creating m new edges. Lines 7–8 introduce the extra edges linking marked nodes. For each pair of distinct nodes u_1, u_2 in the set of marked nodes, an edge between them is added with probability f_{p}, resulting in an expected value of $f_{\mathrm{p}} \cdot \binom{m}{2}$ additional local edges created per algorithm step. Finally, after all iterations are completed, line 9 returns the resulting graph G.

3.3. Time complexity

Assuming that the maximum number of steps in each random walk is a small constant (in our implementation and all tests, we used the value 10), the time complexity of algorithm 1 is $O(m)$, where m is the number of nodes to be marked. The outer while loop runs until m nodes have been visited, and in each iteration, at most one new node is added to the marked set. Inside the loop, a number $\ell \in \{1, \ldots, k\}$ is sampled from a fixed probability distribution, which determines how many steps the random walk performs in that iteration. Although the value of ℓ varies, it is always bounded above by

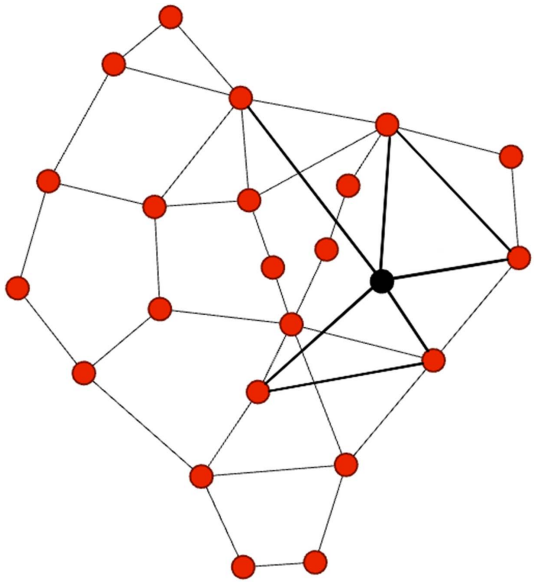

Figure 4. Each pair of marked nodes is connected with a fixed probability f_{p} (step 5 of the algorithm).

Algorithm 1. RandomWalk$(G, start, P, m)$.

marked $\leftarrow$ {*start*}
current $\leftarrow$ *start*
while $|marked| < m$ **do**
  Sample $\ell \in \{1, \ldots, k\}$ from the probability distribution P
  for $i = 1$ to ℓ **do**
    neighbors $\leftarrow$ G[*current*].*neighbors*
    current $\leftarrow$ random element from *neighbors*
  marked $\leftarrow$ *marked* $\cup$ {*current*}
return *marked*

Algorithm 2. GenerateNetwork$(G, N, P, m, f_{\mathrm{p}})$.

repeat N **times**
  start $\leftarrow$ random node from G
  marked $\leftarrow$ **RandomWalk**$(G, start, P, m)$
  $v \leftarrow G$.**add_node**()
  for $u \in$ *marked* **do**
    G.**add_edge**(v, u)
  for all $\{u_1, u_2\} \subset$ *marked* **do**
    With probability f_{p}, run G.**add_edge**(u_1, u_2)
return G

a small value k, so the inner loop performs at most k constant-time operations, making its time complexity constant $O(k)$ per iteration. Therefore, the total time complexity is $O(km) = O(m)$ for small values of k.

Based on this result, the time complexity of the network generation algorithm can be inferred as follows. The main loop runs N times, and in each iteration, it performs a random walk through the RandomWalk subroutine, which runs in $O(m)$ time. After that, a new node is added to the graph and connected to each of the m nodes visited during the walk, which takes $O(m)$ time. Then, for each of the $O(m^2)$ pairs of nodes in the marked set, the algorithm may add an edge with probability f_{p}, which still results in $O(m^2)$ operations in the worst case. Alternatively, we can add $f_{\mathrm{p}} \cdot \binom{m}{2}$ edges (the expected number of edge additions), maintaining the same behavior in the worst case, but improving its performance for small values of f_{p}. Therefore, the overall time complexity of the algorithm is $O(N \cdot (m + m + m^2)) = O(Nm^2)$.

The time complexity is linear in N and quadratic in m. However, m should be small, and much smaller than N, when generating sparse networks. Generating large dense networks is outside the scope of our algorithm, since empirical studies show that large real-world networks are sparse [25]. Therefore, the current time complexity achieves a significant improvement in the time required to generate networks when compared to previous versions of random-walk-based algorithms such as the one proposed by Morais and Interian [11], which has $O(N(N+m)) = O(N^2)$ time complexity, and N may grow to much larger values, reaching millions of nodes, as we show in the next section.

4. Results

In this section, we analyze the key properties of the networks produced by the algorithm. We show that it can generate a diverse range of networks whose characteristics resemble those observed in real-world systems. For the results shown below, the initial graph G was a cycle of length 10 (C_{10}), and $k = 10$. Each value reported in the tables corresponds to the average of the metric calculated over thirty networks generated by the algorithm.

For each network measure in the tables, we start reporting the characteristics of the networks generated using all possible combinations of the following extremal parameter values:

- Number of nodes $N \in \{1000,\ 200\,000\}$;
- Number of edges added for each new node $m \in \{2,\ 5,\ 10\}$;
- Probability associated with the random walk step $p \in \{0.1,\ 0.9\}$;
- Probability of forming edges between marked nodes $f_{\mathrm{p}} \in \{0.1,\ 0.9\}$.

Note that setting $m = 1$ produces a very specific type of network. In this configuration, only one node, chosen at random, is marked at each round. Consequently, no random walks occur, meaning that p and f_{p} have no influence at any step of the algorithm. At each iteration, a new node simply connects to a randomly chosen existing node. The resulting structure is a network with near-zero clustering that preserves the initial ring, while each original node becomes the root of a tree-like structure. Since the starting ring structure has the same number of nodes and edges, and at each step one node and one edge are added, the generated network will always have an equal number of nodes and edges.

On the other hand, very high f_{p} values may produce dense networks, since almost all edges between marked nodes are generated. Empirical studies show that large real-world networks are sparse [25]. Note that this behavior occurs only if p is not too close to one (lower values of p favor larger random walks).

We can achieve fine-grained control over $|E|$, the number of edges in the resulting networks, whose expected value is given by the expression

$$N\left(m + f_{\mathrm{p}} \cdot \binom{m}{2}\right) = Nm\left(1 + \frac{(m-1)}{2} f_{\mathrm{p}}\right).$$

This value can be easily adjusted by tuning m and f_{p}. For each m, the expected number of edges in the network is in the range: $[Nm, Nm(\frac{m+1}{2})]$. Consecutive ranges overlap for $m >= 2$, guaranteeing that any value between $2N$ and the maximum number of edges for a given graph can be achieved by an appropriate choice of m and f_{p}.

Our analysis is focused on three network measures. First, the average local clustering coefficient, $\overline{C}$, which quantifies the likelihood that two neighbors of a given node are also connected, averaged over all nodes in the network. Second, the average shortest path length, $\overline{L}$, which represents the average distances between all pairs of nodes. Finally, the estimated power-law coefficient γ, which characterizes the power-law degree distribution. The exponent γ is estimated by computing the slope of the least-squares linear approximation to the complementary cumulative distribution function of degrees on a log–log scale.

Each table entry is the average of the coefficients of 30 different networks generated with the given combination of parameters, except for the average distance values of the networks with $N = 200\,000$, which are the average of five networks.

Network generation is fast, with an average runtime of 30.74 s for $N = 1\,000\,000$, measured across 30 iterations for each $m \in \{2, 3, 4, 5\}$. However, computing all node-node distances in these large networks is time-consuming, which is why we used the average across five networks to calculate $\overline{L}$ values in

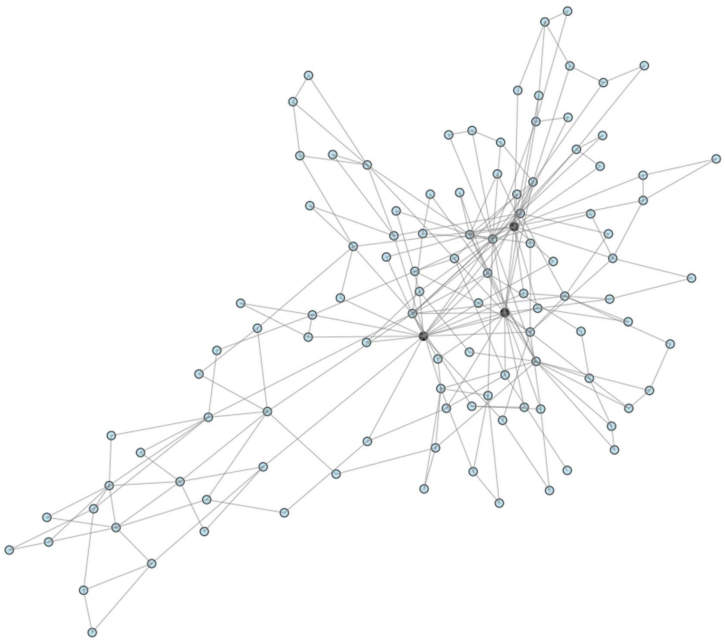

Figure 5. Network generated by the algorithm with $N = 100, m = 2, p = 0.1, f_{\mathrm{p}} = 0.1$. The three nodes with the highest degree are highlighted in black.

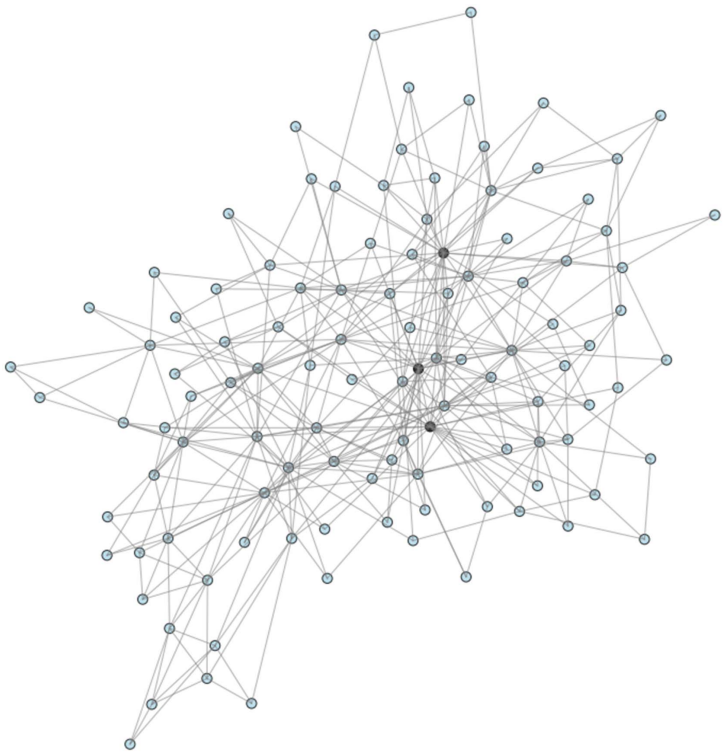

Figure 6. Network generated by the algorithm with $N = 100, m = 3, p = 0.5, f_{\mathrm{p}} = 0.5$. The three nodes with the highest degree are highlighted in black.

table 2. Additionally, we estimated the average distances for larger networks using approximate methods and evaluated the quality of these estimates.

Figures 5 and 6 illustrate two networks generated by the algorithm with two different combinations of parameters, keeping $N = 100$. In both figures, the three nodes with the highest degree (hubs) are highlighted in black.

Table 1. Average clustering coefficients $\overline{C}$ and their standard deviations σ_C for different parameter combinations. The highest and lowest values are highlighted in bold.

m	p	f_p	N	$\overline{C}$	σ_C
2	0.1	0.1	1000	0.2537	0.0157
			200 000	0.2508	0.0010
		0.9	1000	0.5620	0.0078
			200 000	0.5586	0.0007
	0.9	0.1	1000	0.6506	0.0082
			200 000	0.6515	0.0006
		0.9	1000	**0.6813**	0.0058
			200 000	**0.6806**	0.0004
5	0.1	0.1	1000	0.1545	0.0047
			200 000	0.1200	0.0002
		0.9	1000	0.5585	0.0041
			200 000	0.4986	0.0004
	0.9	0.1	1000	0.4199	0.0056
			200 000	0.4058	0.0004
		0.9	1000	0.6071	0.0049
			200 000	0.5807	0.0002
10	0.1	0.1	1000	**0.1783**	0.0035
			200 000	**0.0866**	0.0001
		0.9	1000	0.6213	0.0071
			200 000	0.5070	0.0050
	0.9	0.1	1000	0.2621	0.0028
			200 000	0.1951	0.0001
		0.9	1000	0.6185	0.0056
			200 000	0.5233	0.0040

4.1. Clustering coefficients

Table 1 shows the values obtained for the average clustering coefficient $\overline{C}$. As shown in this table, the results range from a minimum of 0.09 to a maximum of 0.68, which is substantially higher than the largest clustering coefficients reported for the algorithm proposed by Morais and Interian [11]. The network with the smallest clustering coefficient is generated using the parameter combination $N = 200\,000$, $m = 10$, and $p = f_p = 0.1$, while the largest values are obtained for $m = 2$ and $p = f_p = 0.9$. The clustering coefficient values do not depend significantly on N.

These trends are further illustrated in the parallel coordinates plot shown in figure 7. In this visualization, each vertical line or axis represents a different input parameter. The plotted configurations were selected from simulations spanning $N \in [2500, 200\,000]$, $m \in [2, 10]$, and probabilities $p, f_p \in [0.1, 0.9]$. From this parameter space, we retained the 5 configurations with the highest average clustering coefficient $\overline{C}$, the 5 with the lowest $\overline{C}$, and 10 other configurations chosen at random. The value of the clustering coefficient $\overline{C}$ obtained from one combination of algorithm parameters is represented as a connected line segment (a polyline). Each polyline is colored according to the $\overline{C}$ value.

Higher clustering values are associated with the largest values of p and f_p, revealing a clear positive relationship between these parameters and $\overline{C}$ across different choices of N and m. This behavior is expected, since both parameters directly increase the formation of local triangles: higher values of p favor shorter random walks, increasing the likelihood that the new node attaches to neighboring nodes and closes triangles, while larger values of f_p increase the number of edges added among marked nodes, further contributing to the number of closed triplets involving the newly added node.

4.2. Average distances

Table 2 reports the values obtained for the average shortest path length $\overline{L}$. The maximum average shortest path length, 12.55, is observed for the parameter combination $N = 200\,000$, $m = 2$, $p = 0.9$, and $f_p = 0.1$, while the minimum value, 2.14, occurs for $N = 1000$, $m = 10$, $p = 0.1$, and $f_p = 0.9$. As expected, increasing the network size N increases the distances between nodes. With $N = 200\,000$, the algorithm is

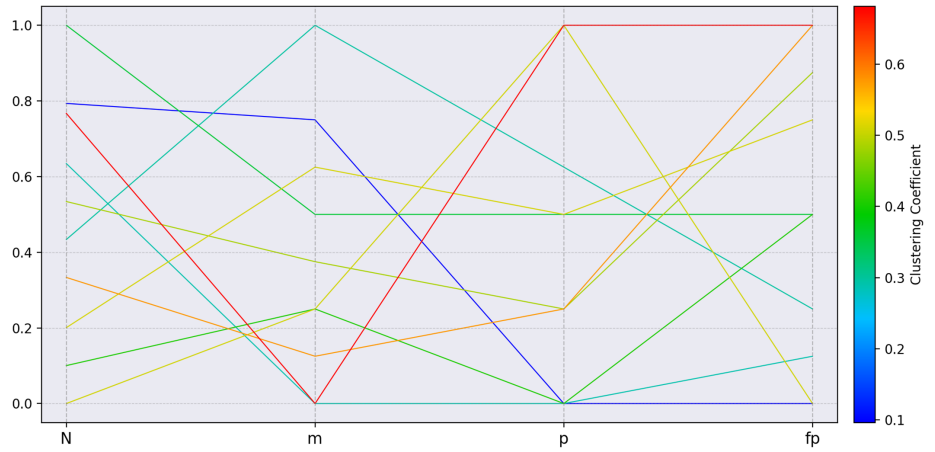

Figure 7. Parallel coordinates plot of N, m, p, f_{p} versus the clustering coefficient $\overline{C}$. The color shifts from blue to red as the clustering coefficient increases. The values shown in each axis are normalized, with m varying from 2 to 10, and p and f_{p} from 0.1 to 0.9. N is on a logarithmic scale and varies from 2500 to 250 000 nodes.

capable of going as low as 2.83, a substantial improvement when compared with the algorithm proposed by M–I [11], which could not achieve values lower than 4.00, even for a network with $N = 50\,000$.

These trends are further clarified by the parallel coordinates plot shown in figure 8, where each parameter configuration is represented by a polyline, which is colored according to the corresponding $\overline{L}$ value. The plotted configurations were selected from simulations covering $N \in [2500, 100\,000]$, $m \in [2, 10]$, and probabilities $p, f_{\mathrm{p}} \in [0.1, 0.9]$. From this ensemble, we included the 10 configurations with the highest $\overline{L}$ values, the 10 with the lowest $\overline{L}$ values, and 10 configurations randomly sampled from the full parameter space. In this visualization, we can see that increasing m systematically shifts the curves toward lower values of $\overline{L}$.

In contrast, $\overline{L}$ is observed to decrease as p decreases. Smaller values of p favor longer random walks, increasing the probability that the marked nodes are farther apart when the new node is attached, thereby promoting the formation of long-range connections. As shown in the parallel coordinates plot, this mechanism enables even large networks to exhibit small average distances with appropriate parameter selection.

To examine how the average shortest path length scales in very large networks, we generated 30 networks with $N = 5\,000\,000$, using parameter values $m = 2$, $p = 0.1$, and $f_{\mathrm{p}} = 0.9$. Because computing the exact average shortest path length for networks of this size is computationally prohibitive, we employed the *random vertex sampling method* [26]. Specifically, we selected 1000 nodes uniformly at random, computed the average distance from each selected node to every other node, and then averaged these values to obtain an estimate for the network. Across the 30 generated networks, the average distance was $\overline{L} = 7.9925$, with a standard deviation of $\sigma_L = 0.1494$. This result shows that small-world behavior is maintained across different large generated networks, with reduced variability. In these simulations, the average network construction time remained below 40 seconds under these specific parameter settings.

4.3. Presence of hubs

Finally, we evaluated the presence of hubs (high-degree nodes) by estimating the power-law exponent γ of each generated network. Smaller values of γ indicate the presence of more hubs in the network.

Table 3 presents the values of γ, the power-law exponent, while the parallel coordinates plot in figure 9 illustrates how they behave with different combinations of the parameters. The plotted configurations were chosen from simulations spanning $N \in [1000, 200\,000]$, $m \in [2, 5, 10]$, and probabilities $p, f_{\mathrm{p}} \in [0.1, 0.9]$. From this set, we chose the 5 configurations yielding the highest γ values, the 5 yielding the lowest, and 10 random configurations from the same ranges.

The values range from a minimum of approximately 2.31 (for $N = 1000,\ m = 10,\ p = 0.1,\ f_{\mathrm{p}} = 0.9$) to a maximum of 4.26 (for $N = 200\,000,\ m = 2,\ p = 0.9,\ f_{\mathrm{p}} = 0.1$). Note that, in each case, these extrema occur at opposite ends of the parameter ranges. The results indicate that higher values of m increase the proportion of hubs, nodes with a large number of connections.

The parameters p and f_{p} exhibit a relationship with γ: as p increases, γ tends to increase, whereas larger values of f_{p} drive γ downward. Lower values of p produce longer random walks, making the proportion of marked nodes increasingly reflect the random-walk stationary distribution corresponding to

Table 2. Average distances $\overline{L}$ and their standard deviations σ_L for different parameter combinations. The highest and lowest values are highlighted in bold.

m	N	p	f_p	$\overline{L}$	σ_L
2	1000	0.1	0.1	5.3640	0.1130
			0.9	4.4622	0.1278
		0.9	0.1	7.1677	0.1003
			0.9	6.9229	0.3235
	200 000	0.1	0.1	9.0370	0.1926
			0.9	6.7508	0.0181
		0.9	0.1	**12.5504**	0.1611
			0.9	**11.7811**	1.3129
5	1000	0.1	0.1	2.9591	0.0113
			0.9	2.5478	0.0045
		0.9	0.1	3.6018	0.1622
			0.9	2.8073	0.1314
	200 000	0.1	0.1	4.3353	0.0012
			0.9	3.3015	0.0022
		0.9	0.1	5.1996	0.1351
			0.9	3.6703	0.0592
10	1000	0.1	0.1	2.4250	0.0050
			0.9	**2.1416**	0.0054
		0.9	0.1	2.4771	0.0059
			0.9	**2.1726**	0.0017
	200 000	0.1	0.1	3.2941	0.0029
			0.9	2.8359	0.0001
		0.9	0.1	3.4174	0.0198
			0.9	2.8588	0.0004

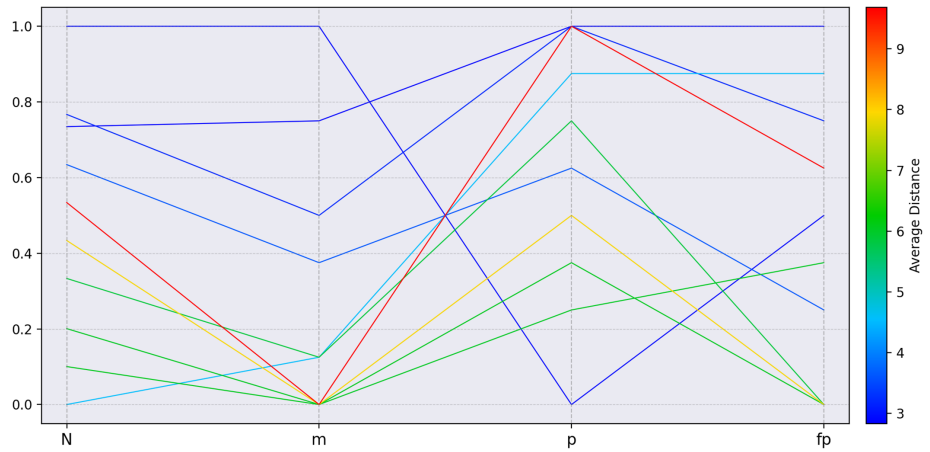

Figure 8. Parallel coordinates plot of N, m, p, f_p versus the average shortest path length. The color shifts from blue to red as the average distance increases. The values shown in each axis are normalized, with m varying from 2 to 10, and p and f_p from 0.1 to 0.9. N is on a logarithmic scale and varies from 250 to 250 000 nodes. It can be seen that even networks with large values of N can lead to small-distance networks with the correct choice of the other parameters.

node degrees [27]. For an undirected graph, $\pi(v) \propto \deg(v)$, therefore high-degree nodes are visited and marked more often, leading to a higher frequency of hubs and to smaller values of γ.

In contrast, increasing f_p generates more connections among marked nodes, boosting node degrees, and lowering γ.

Taken together, these parameter combinations provide substantial control over the power-law behavior of the generated networks, a feature absent in the previous Morais and Interian [11] model.

Table 3. Values of γ and their standard deviations σ_γ for different parameter combinations. The highest and lowest value are highlighted in bold.

m	N	f_p	p	γ	σ_γ
2	1000	0.1	0.1	3.3225	0.1231
			0.9	3.3316	0.1340
		0.9	0.1	3.1448	0.0944
			0.9	3.2893	0.1651
	200 000	0.1	0.1	4.2028	0.0752
			0.9	**4.2600**	0.1223
		0.9	0.1	3.8634	0.0839
			0.9	4.2423	0.1106
5	1000	0.1	0.1	2.9642	0.1085
			0.9	3.0496	0.1102
		0.9	0.1	2.5944	0.0990
			0.9	2.7451	0.1183
	200 000	0.1	0.1	3.3730	0.0901
			0.9	3.4835	0.0737
		0.9	0.1	3.0652	0.0986
			0.9	3.1796	0.0974
10	1000	0.1	0.1	2.6797	0.0936
			0.9	2.7412	0.1069
		0.9	0.1	**2.3009**	0.0958
			0.9	2.3128	0.1302
	200 000	0.1	0.1	3.0845	0.0815
			0.9	3.1413	0.0965
		0.9	0.1	2.7871	0.0981
			0.9	2.8314	0.0840

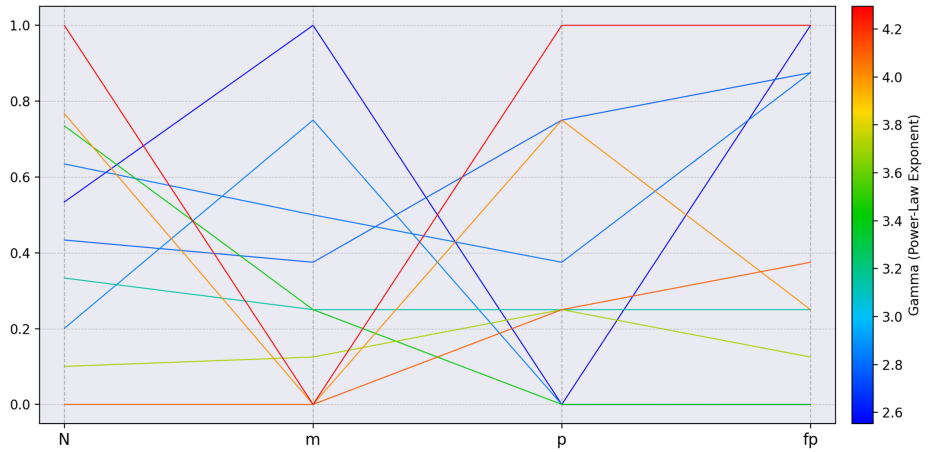

Figure 9. Parallel coordinates plot of N, m, p, f_p versus gamma. The color shifts from blue to red as the power-law coefficient increases. The values shown in each axis are normalized, with m varying from 2 to 10, and p and f_p from 0.1 to 0.9. N is on a logarithmic scale and varies from 2500 to 250 000 nodes.

4.4. Generating networks with specific features

To illustrate how the proposed algorithm can generate networks with some target structural properties, we show how to fit the networks produced by our model to a reference network. This enables the construction of collections of realistic synthetic networks that can be used, e.g. to simulate processes (diffusion, robustness, intervention strategies) or to test optimization algorithms in real-world scenarios. In the first place, we match exactly the reference network's number of nodes. Then, we consider the number of edges $|E|$, the average clustering coefficient $\overline{C}$, the estimated power-law exponent γ, and the average shortest-path length $\overline{L}$ as all possible tunable target features.

Ignoring the number of nodes N, which is in one-to-one correspondence with the size of the resulting network's node set, the number of the algorithm's input parameters (three: m, p, f_p) is smaller than the number of all target features of the resulting network (four: $|E|$, $\overline{L}$, $\overline{C}$, γ). We also note that the set of four network features considered in this study is not exhaustive, and more potentially relevant features may be considered (for example, the network's modularity). Therefore, it becomes more difficult to generate all possible elements of a four-dimensional feature space using three parameters. This task is even more difficult for any graph generation algorithm if we want to keep it simple and interpretable, as the first graph generation challenge identified by Bonifati *et al* [20] ('simple usage, simple parameters' principle).

On the other hand, it is valid to expect a consistent generation of a network with fewer target features. There is a substantial number of target feature subsets (four individual parameters, six pairs, four triples), and different combinations may require specific fitting procedures. In this section, we illustrate how to generate a network with two chosen features, and analyze the ability to generate networks by fixing one feature at a time. In future work, we are interested in studying these parameter-fitting procedures in a more systematic way.

As a first example of a reference network, we chose the Internet topology graph [25], having 10 697 nodes and 31 992 edges. In this example, two features of the empirical network are chosen as targets: the number of edges and the average clustering coefficient $\overline{C} = 0.39$.

First, the number of nodes the algorithm will add to the initial network is fixed at $N = 10\,687$. We then focus on m and f_p, the parameters that influence the number of edges, to identify parameter values that yield the desired edge count. Given N, m, and f_p it follows that the expected number of edges $|E|$ in the generated network is roughly $|E| = N\left(m + f_p \cdot \binom{m}{2}\right)$. Therefore, we consider the potential values of m (starting with $m = 1$) for which f_p may lie within the range $(0, 1)$ according to the equation. Fixing m, we then gradually increase f_p from close-to-zero values until the desired number of edges is reached. This procedure led to the selection of one of the possible (m, f_p) pairs $(m = 3, f_p = 0.1)$, which produces a number of edges $|E| = 31\,906$ that is almost identical to that of the reference network. The number of edges is slightly lower than $3N$ due to the random nature of the walk: sometimes it can mark the same node twice, that is, not all marked nodes are necessarily distinct.

Holding the other parameters fixed, we then increase p from a value close to zero until the average clustering coefficient of the generated network matches its empirical value $\overline{C} = 0.39$ at $p = 0.47$. We recall that higher clustering values are associated with the largest values of p.

The resulting configuration ($N = 10\,687$, $m = 3$, $p = 0.47$, and $f_p = 0.1$) yields $|E| = 31\,906$ and $\overline{C} = 0.392$ (averages over 30 generated networks). The other two features maintained realistic values: $\overline{L} = 5.487$ (small-world) and the estimated $\gamma = 3.562$. As expected, the procedure achieves an excellent agreement for the two prioritized features: $|E|$ and $\overline{C}$.

We extended this analysis to several real-world networks from a seminal article by Mark Newman [25], including the physics coauthorship network, with $\overline{C} = 0.56$, the biology coauthorship network, with $\overline{C} = 0.6$, and the metabolic network, with $\overline{C} = 0.67$. We obtained the following $\overline{C}$ values: 0.559, 0.597, and 0.668 for physics coauthorship, biology coauthorship, and metabolic network, respectively, and closely replicated the number of edges in each case.

On the other hand, we are also able to generate a network with the same power-law exponent of the Internet topology network by numerically exploring the parameter space of (m, f_p, p) on the synthetic graphs, while also maintaining the average distances relatively close to the empirical value. We leveraged the effects of the parameters (f_p and m decrease γ, while p increases it) to guide the search: for each integer m, f_p and p can be adjusted to bring γ closer to the empirical target. Using this approach, we were able to closely replicate the reported exponent of 2.5, obtaining $\gamma = 2.54$ and $\overline{L} = 2.52$ in the generated network.

Overall, the Large Network Generator can also reproduce structural properties of a wide range of real-world networks individually (considering one feature at a time). We based our analysis on the undirected networks investigated by Mark Newman [25]. First, 13 of the 15 reported clustering coefficients fall within the range produced by our method. For the power-law exponents, 6 of the 9 reported values fall within the algorithm's achievable range. Finally, considering undirected networks with at most 200 000 nodes, our algorithm can generate networks with similar or smaller average distances for 8 of 10 networks reported in [25]. The next section presents these ranges of values for each structural parameter, compared with previous network generation approaches.

4.5. Comparison with other network generation methods

To illustrate the advantages of the proposed algorithm, we compare the new Large Network Generator with five models: Barabási-Albert [6], Holme and Kim [16], Saramäki and Kaski [10], Herrera and

Zufiria [19] and Morais and Interian [11]. For each model, we evaluate several structural characteristics of the generated networks and the model's time complexity. Furthermore, we analyze the degree of control they have over each network feature.

The Barabási-Albert model captures the emergence of power-law degree distributions through the mechanisms of network growth and preferential attachment, but has several limitations. It only generates networks with low clustering coefficients. The total number of edges is constrained to be a multiple of N. The model lacks mechanisms to effectively control distances between nodes. It produces power-law degree distributions with exponents confined to a narrow and non-tunable range. Finally, the model needs global information.

Holme and Kim proposed an improvement by adding a triad formation step to the previous model, a mechanism to increase and control the clustering coefficient. The new model achieves high clustering coefficients while keeping the scale-free property of its predecessor. Yet the other limitations remained: a lack of a mechanism to control average distances and the power-law coefficients, and reliance on global information.

The Saramäki-Kaski model is capable of generating scale-free networks using only local information. However, it also provides limited control over the key structural properties. The number of edges can only be changed in multiples of N. The model can produce clustering coefficients as high as 0.75, but it does not provide a direct mechanism for controlling this property. Average distances can be controlled by varying the parameter l, which reduces average path lengths through longer random walks, but this effect is coarse-grained, making it difficult to achieve specific target distances. The model does not provide an explicit mechanism for tuning the power-law exponent.

The model proposed by Herrera and Zufiria is an enhancement over the Saramäki-Kaski model by introducing a mechanism to control the clustering coefficient. However, the model still exhibits limitations. The total number of edges in the generated networks can only be adjusted in multiples of N. The model also allows setting the parameter l to only two possible values (1 and 2), considerably reducing the control over the average distances. Furthermore, the model does not provide a clear mechanism for controlling the power-law exponent.

The previous Morais and Interian [11] algorithm extends the Herrera-Zufiria model by incorporating an additional mechanism to reduce average path lengths while preserving other structural properties and maintaining the use of only local information. In addition to connecting newly added nodes through random walks, the model introduces shortcut edges between pairs of nodes selected according to a distance-dependent probability distribution. This mechanism reduces the average distances of the Herrera-Zufiria model. The model allows control over clustering via the random walk parameter. The power-law exponent is not directly tunable.

The model proposed in this work addresses the aforementioned limitations. It provides a fine-grained control over $|E|$, the number of edges in the resulting networks. It also provides strong control over the clustering coefficient, as shown in the previous section. The average distance can be regulated by p, which determines the length of the random walks: as p decreases, longer walks are encouraged, thus reducing the average distances. Additionally, the model provides control over the power-law exponent through appropriate parameter tuning: increasing m and f_p reduces the exponent, whereas increasing p increases it. Finally, the algorithm operates without requiring any global information, consistent with the decentralized nature of real-world networks.

We implemented the comparison analysis of the models considering networks with $N \in [1000, 200\,000]$, fixing, for each model, the number of edges to be added for each node $m \in [2, 10]$. We used the same initial graph with $m_0 = 10$ nodes as defined by each algorithm (rings for B–A, H–Z, and M–I; disconnected nodes for H–K; clique for S–K). For the H–K model, the probability of the triad formation step is varied within the interval $[0.1, 0.9]$, while the same range is used for the clustering coefficient parameter in the H–Z model and for the p_1 parameter in the M–I model. In the S–K model, the parameter l is varied in the interval $[1, 10]$. The average distance analysis is performed for networks with $N = 200\,000$.

Table 4 summarizes the comparisons between the models, showing whether they have control over the output parameter, and the range of each measured parameter for each model. We consider a model to have control over some parameter if it provides a specific procedure for reducing or increasing it. The control over the number of edges ($|E|$) is evaluated differently: we consider that a model has *weak control* if it can only reach a small set of possible number of edges values (generally, multiples of N), without covering intermediate values between them, and *strong control* if it can achieve intermediate values.

The Large Network Generator can achieve a wider interval of values for average distances than its predecessors. It achieves a wider power-law exponent range than all other methods except for one lower bound of a much more costly method. Finally, it also keeps the upper limit of the generated network's

Table 4. Comparison of network generation models, considering $N \in [1000, 200\,000]$. Measures: number of edges $|E|$, average local clustering coefficient $\overline{C}$, average shortest path length $\overline{L}$, estimated power-law exponent γ.

Model	Global information	$\|E\|$ control	$\overline{C}$: control, range	$\overline{L}$: control, range	γ: control, range	Time complexity
Barabási and Albert [6]	Yes	Weak	No, [0.00, 0.06]	No, [3.7, 6.2]	No, [2.7, 3.2]	$O(Nm)$
Holme and Kim [16]	Yes	Weak	Yes, [0.00, 0.68]	No, [3.7, 7.7]	No, [2.8, 3.2]	$O(Nm)$
Saramäki and Kaski [10]	No	Weak	No, [0.00, 0.75]	Yes, [3.8, 8.2]	No, [2.4, 3.5]	$O(Nml)$
Herrera and Zufiria [19]	No	Weak	Yes, [0.00, 0.71]	No, [3.8, 9.7]	No, [2.7, 3.4]	$O(Nm)$
Morais and Interian [11]	No	Weak	Yes, [0.00, 0.55]	No, [3.5, 4.0]	No, [1.7, 3.5]	$O(N^2)$
Large Network Generator	No	Strong	Yes, [0.00, 0.68]	Yes, [2.8, 12.6]	Yes, [2.3, 4.3]	$O(Nm^2)$

clustering coefficients very close to those of other models. We note that clustering coefficients greater than 0.7 are highly uncommon in real networks [25].

In terms of time complexity, most predecessor methods run in $O(Nm)$, with two more costly exceptions. The Large Network Generator, on the other hand, has a time complexity of $O(Nm^2)$. At first glance, it may seem slightly less efficient than the other approaches due to the squared m term. However, as discussed earlier, the parameter m should not exceed 10 in practical applications, as is the case with all the results reported in this study. Under this constraint, the additional cost remains limited, and the Large Network Generator, in the worst case ($m = 10$), is more costly than other methods by only a constant factor of $\binom{10}{2} \div 10 = 4.5$. Nevertheless, this shortcoming is addressed by our efficient implementation, enabling the generation of networks with one million nodes in less than 60 s.

5. Conclusions

We proposed the *Large Network Generator*, a random-walk-based growth algorithm designed to create large, sparse graphs with structural signatures typically observed in real-world networks. The algorithm is intentionally simple: at each iteration, a new node is connected to a set of existing nodes selected by a random walk, and then additional edges are created among the selected nodes, implementing a triadic-closure mechanism [21, 24]. Because both steps rely solely on local neighbor information, the model avoids global topology requirements (e.g. computing attachment probabilities based on the degrees of all nodes, or on their coordinates in some Euclidean space), a common assumption in other growth models [6]. Our implementation is available in a publicly accessible GitHub repository [12]. A PyPI package of the Large Network Generator is also available [13].

The experiments presented in this paper indicate that this unified mechanism is sufficient to reproduce three key properties simultaneously: long-tailed degree distributions, high clustering, and short average distances, yielding a wide range of values for each parameter.

Regarding clustering, the proposed algorithm provides an ample, controllable range of average local clustering coefficients. The parameter f_p directly increases the number of closed triplets by creating additional edges among the nodes selected during the random walk. This mechanism enables the generation of networks with clustering coefficients spanning from low to very high values, exceeding those obtained by previous random-walk-based models.

The average shortest path length is mainly influenced by the interaction between the random-walk step length distribution and the growth process. Although most random-walk steps remain local, occasional longer walks allow the newly added node to connect distant regions of the network, effectively acting as a shortcut without introducing explicit long-range edge creation steps. As a result, the generated networks display small-world behavior, with average distances remaining short and growing slowly with network size. This provides a natural and efficient mechanism for controlling distances using only local information, addressing a limitation of earlier random-walk growth models, which produced networks with large diameters.

Finally, the algorithm offers substantial control over the degree distribution and its associated power-law exponent γ. As p decreases, node selection increasingly reflects the stationary distribution of the random walks, which is proportional to node degrees in undirected graphs [27]. Consequently, as p decreases, high-degree nodes are more likely to be revisited and selected as attachment points, reinforcing the emergence of hubs, and leading to smaller values of γ. The exponent γ varies across a broad range while maintaining a long-tailed degree distribution. Among the limitations of our approach, we can mention the difficulty of achieving gamma values below 2.7 for very large networks, although larger gamma values are easier to reach and control.

The proposed algorithm addresses the two key challenges highlighted by Bonifati *et al* [20]: simple use with simple parameters, and the generation of large graphs with realistic structures. The model relies on a small set of parameters (N, m, p, f_p), each with a clear interpretation. Despite this simplicity, the algorithm can generate networks that simultaneously exhibit the main structural signatures of real-world graphs, including high clustering, short average path lengths (small-world behavior), and long-tailed degree distributions.

Compared to previous models, our algorithm offers substantially greater flexibility and performance on several fronts. For example, it can produce much higher clustering (up to $\overline{C} \approx 0.7$), compared to those reported in [11], while also achieving markedly short average distances (we observe values as low as $\overline{L} \approx 2.8$ for $N = 200\,000$), outperforming several existing methods. At the same time, the algorithm leads to the presence of hubs, enabling the simultaneous emergence of long-tailed degree distributions, high clustering, and small-world distances without relying on any form of global information.

In addition to its flexibility, the proposed algorithm is computationally efficient and scalable in practice: networks with $N = 200\,000$ nodes are generated in less than 30 s, and networks with $N = 1\,000\,000$ nodes in less than 60 s on average, properties that make the model much more suitable for large-scale numerical experiments and systematic parameter studies.

As a next step, we intend to investigate the precise mathematical relationships between the algorithm parameters and the resulting network features. While the present work demonstrates that these parameters provide effective qualitative control over the generated structures, a deeper understanding of their interactions may enable more precise quantitative predictions of the network properties produced by a given parameter configuration, rather than relying solely on empirical analysis. Such an analysis could lead to functional mappings that enable the selection of parameter combinations to generate networks with prescribed characteristics.

We are interested in replicating, in a more systematic way, more than two real-world network features in our synthetic networks. This research would further enhance the usefulness of the proposed generator as a practical tool for synthetic network construction and benchmarking, especially in applications where matching specific structural targets is essential.

We are also interested in exploring other distributions, besides the truncated geometric, to evaluate how this choice affects the considered network properties. It would be valuable to explore different distribution' upper bounds (k), beyond the fixed value of 10, especially when generating larger networks, for example, with tens or hundreds of millions of nodes.

Acknowledgments

This work was supported by Instituto Kunumi. The authors thank the institution for its financial support and commitment to advancing scientific research. This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number 2024/12936-5. Ruben Interian was supported by research grant PIND 2423/24 (Universidade Estadual de Campinas). The work of Celso C. Ribeiro was supported by research grant CNPq 309869/2020-0.

Data availability statement

No new data were created or analysed in this study.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Author contributions

João Pedro C Morais 0009-0001-3155-3164
Conceptualization (equal), Investigation (equal), Methodology (equal), Software (equal), Visualization (equal), Writing – original draft (equal)

Celso C Ribeiro 0000-0002-9478-2351
Methodology (equal), Supervision (equal), Writing – review & editing (equal)

Ruben Interian 0000-0001-5878-6462
Conceptualization (equal), Funding acquisition (equal), Investigation (equal), Methodology (equal), Supervision (equal), Writing – original draft (equal), Writing – review & editing (equal)

References

[1] de Arruda G F, Rodrigues F A and Moreno Y 2018 Fundamentals of spreading processes in single and multilayer complex networks *Phys. Rep.* **756** 1–59
[2] Interian R, Marzo R G, Mendoza I and Ribeiro C C 2023 Network polarization, filter bubbles and echo chambers: an annotated review of measures and reduction methods *Int. Trans. Oper. Res.* **30** 3122–58
[3] Seguin C, Sporns O and Zalesky A 2023 Brain network communication: concepts, models and applications *Nat. Rev. Neurosci.* **24** 557–74
[4] Zhou B, Holme P, Gong Z, Zhan C, Huang Y, Lu X and Meng X 2023 The nature and nurture of network evolution *Nat. Commun.* **14** 7031
[5] Interian R 2024 A political radicalization framework based on Moral Foundations Theory *Mathematics* **12** 2121
[6] Barabási A-L and Albert R 1999 Emergence of scaling in random networks *Science* **286** 509–12
[7] Broido A and Clauset A 2019 Scale-free networks are rare *Nat. Commun.* **10** 1017
[8] Bhattacharya S, Sinha S and Roy S 2020 Impact of structural properties on network structure for online social networks *Proc. Comput. Sci.* **167** 1200–9
[9] Watts D and Strogatz S 1998 Collective dynamics of 'small-world' networks *Nature* **393** 440–2
[10] Saramäki J and Kaski K 2004 Scale-free networks generated by random walkers *Physica A* **341** 80–86
[11] Morais J P CandInterian R 2025 A simple and flexible algorithm to generate real-world networks *Complex Networks XVI(CompleNet)*,(Springer) pp 1–12
[12] Morais J P C and Interian R 2026 GitHub repository: real-world networks algorithm (available at: https://github.com/jpcmorais16/real-world-network-algorithm)
[13] Morais J P C and Interian R 2026 Large network generator (PyPI package) (available at: https://pypi.org/project/large-network-generator/)
[14] Boguñá M, Bonamassa I, de Domenico M, Havlin S, Krioukov D and Serrano M A 2021 Network geometry *Nat. Rev. Phys.* **3** 114–35
[15] Boguñá M, Krioukov D, Almagro P and Serrano M A 2020 Small worlds and clustering in spatial networks *Phys. Rev. Res.* **2** 023040
[16] Holme P and Kim B J 2002 Growing scale-free networks with tunable clustering *Phys. Rev. E* **65** 026107
[17] Klemm K and Eguíluz V M 2002 Highly clustered scale-free networks *Phys. Rev. E* **65** 036123
[18] Davidsen J, Ebel H and Bornholdt S 2002 Emergence of a small world from local interactions: modeling acquaintance networks *Phys. Rev. Lett.* **88** 128701
[19] Herrera C and Zufiria P J 2011 Generating scale-free networks with adjustable clustering coefficient via random walks *2011 IEEE Network Science Workshop* pp 167–72
[20] Bonifati A, Holubová I, Prat-Pérez A and Sakr S 2020 Graph generators: state of the art and open challenges *ACM Comput. Surv.* **53** 1–30
[21] Granovetter M S 1973 The strength of weak ties *Am. J. Sociol.* **78** 1360–80
[22] Newman M E J 2001 The structure of scientific collaboration networks *Proc. Natl Acad. Sci.* **98** 404–9
[23] Snijders T A B 2001 The statistical evaluation of social network dynamics *Sociological Methodology* **31** 361–95
[24] Easley D and Kleinberg J 2010 *Networks, Crowds and Markets: Reasoning About a Highly Connected World* (Cambridge University Press)
[25] Newman M E J 2003 The structure and function of complex networks *SIAM Rev.* **45** 167–256
[26] Ye Q, Wu B and Wang B 2010 Distance distribution and average shortest path length estimation in real-world networks *Advanced Data Mining and Applications*, ed L Cao, Y Feng and J Zhong (Springer) pp 322–33
[27] Lovász L 1993 Random walks on graphs: a survey *Combinatorics, Paul Erdos is Eighty* **2** 1–46