Physica A 390 (2011) 2172–2180

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

On the efficiency of data representation on the modeling and characterization of complex networks

Carlos Antônio Ruggiero [a], Odemir Martinez Bruno [a], Gonzalo Travieso [a,*], Luciano da Fontoura Costa [a,b]

[a] Instituto de Física de São Carlos, Universidade de São Paulo, Caixa Postal 369, São Carlos, São Paulo, 13560-970, Brazil

[b] National Institute of Science and Technology for Complex Systems, Brazil

ARTICLE INFO

Article history:
Received 27 October 2010
Received in revised form 10 January 2011
Available online 15 February 2011

Keywords:
Complex networks
Representation
Computational efficiency

ABSTRACT

Specific choices about how to represent complex networks can have a substantial impact on the execution time required for the respective construction and analysis of those structures. In this work we report a comparison of the effects of representing complex networks statically by adjacency matrices or dynamically by adjacency lists. Three theoretical models of complex networks are considered: two types of Erdős–Rényi as well as the Barabási–Albert model. We investigated the effect of the different representations with respect to the construction and measurement of several topological properties (i.e. degree, clustering coefficient, shortest path length, and betweenness centrality). We found that different forms of representation generally have a substantial effect on the execution time, with the sparse representation frequently resulting in remarkably superior performance.

1. Introduction

As a consequence of the intrinsic difficulties in achieving analytical approaches for the characterization and modeling of natural systems, a great deal of such investigations must rely on computational methods. The typical case involves the application of numerical methods in order to solve differential equations (e.g. Refs. [1,2]), which is the most frequent situation found in practice in physics. Frequently, such problems involve large amounts of data, as well as data which are very large. Given the importance of effectively tackling these problems, a lot of attention and effort has been invested in developing, implementing and applying numerical methods which are fast and accurate (e.g. Ref. [1]). Indeed, such efforts give rise to the important area of *computational physics*.

Different approaches are needed in complex networks research [3–5], a new multidisciplinary area of physics which has undergone an impressive development in the past decade. Here, the investigations rely not just on numerical solution of differential equations, but on intensive handling of matrices as well as combinatorial or spectral methods as required for the calculation of measurements [6] such as shortest paths, betweenness centrality, and spectra of graphs. Though presenting such a distinctive nature, computational approaches to complex networks also aim at achieving precision and speed. The latter demand often becomes particularly critical as a consequence of the large size of several complex networks of current interest, such as the Internet [7], protein-protein interaction [8], and social interactions [9], to name but a few cases. In other words, as an ever increasing number of large real-world networks are being constructed and made available, it becomes critically important to develop the means not only for the efficient representation of such structures, but also for handling the frequently computationally expensive operations demanded by their analysis. For instance, it can be estimated that

* Corresponding author. Tel.: +55 16 33739857.
E-mail addresses: toto@ifsc.usp.br (C.A. Ruggiero), bruno@ifsc.usp.br (O.M. Bruno), gonzalo@ifsc.usp.br (G. Travieso), luciano@ifsc.usp.br (L. da Fontoura Costa).

0378-4371
doi:10.1016/j.physa.2011.02.011

the calculation of the betweenness centrality of a typical collaboration network (e.g. movie stars) with about 400 000 nodes, represented in terms of its standard adjacency matrix, would require more than three months in a typical personal computer. As shown in the present work, it would be possible to reduce drastically the execution time to about one day simply by representing the complex network in terms of its adjacency list.

Indeed, the effective approach to most of the remaining challenges in complex networks research is immediately related to the ability to effectively represent and process large discrete mathematical structures ([10,11]). This can be done in the two following ways: (i) development of more effective algorithms [12]; and (ii) careful and efficient respective implementation of those algorithms. While much attention has been placed recently on (i), the final performance will ultimately depend critically on the implementation, making step (ii) particularly critical for achieving good results. The current work focuses on important practical implementational aspects related to the use of sparse or full representation of graphs. To our best knowledge, the present article constitutes one of the few works [13] investigating the effect of such important practical choices on the resulting efficiency of the implementation of a set of crucially important operations typically performed in complex networks research, including network generation as well as the estimation of important topological properties such as the degree, clustering coefficient, shortest path length, and betweenness centrality.

This article starts by describing the computational tasks to be performed, namely the estimation of several topological features of the networks, and follows by presenting the adopted network models and the two types of representations of networks to be compared. The work concludes by presenting and discussing the computational efficiency of these two representations as obtained through computational simulations.

2. The methods chosen for the evaluation

It is henceforth assumed that all networks are undirected and unweighted. Full representations of the networks are performed in terms of the respective *adjacency matrices*, K, such that the presence of an edge between nodes i and j imply $K(i, j) = K(j, i) = 1$, with $K(i, j) = K(j, i) = 0$ being otherwise imposed. The total number of nodes (also known as network size) and edges in the networks are respectively abbreviated as N and E. A set of four representative methods/measurements of complex networks have been selected in order to investigate the effect of implementational parameters and choices on the respective performance: degree, clustering coefficient, shortest path and betweenness centrality. Each of these methods are briefly revised in the following.

Degree: The degree of a node i corresponds to the number of edges attached to it. It can be calculated by adding all entries in column i of the adjacency matrix. The degree is an intrinsically local measurement, in the sense of taking into account only the edges directly attached to the node. Usually, the degree is calculated for all the nodes of a given network.

Clustering coefficient: The clustering coefficient is also a local measurement, specific to each node i. However, it also considers the interconnectivity between the neighbors of that node. In the case of full representation in terms of the adjacency matrix, the calculation of these measurements requires access to all the columns corresponding to each of the neighbors of node i.

Shortest path identification: Given two nodes i and j, the shortest topological path between them corresponds to the path which has the smallest number of edges. Note that it is possible to have two or more distinct shortest paths of the same size.

Betweenness centrality: The betweenness centrality is a property associated to a given node or edge. In both cases, it refers to the number of shortest paths, considering all pairs of nodes in the networks, which pass through the given node or edge. The calculation of the betweenness centrality requires the determination of the shortest paths for every pair of distinct nodes.

3. Network models

In the following we use three network models. Two models due to Erdős and Rényi (ER) and the scale free model of Barabási and Albert [14] (BA). The first model, denoted ER (probability), connects each pair of nodes with a fixed probability p. The average degree in this model is $p(N - 1)$, where N is the number of nodes in the network. The second model, denoted ER (edges), uses a fixed number E of edges, and connects each edge to a randomly chosen pair of nodes. The average degree of the network is $2E/N$. These two models have similar statistical properties, but are included here due to their different behavior during network construction: as the first model must consider all pairs of nodes, it is computationally intensive for large networks; the second model has construction time proportional to the number of edges, and is therefore faster for sparse networks.

The Barabási–Albert networks are constructed starting with a small number of nodes, and adding nodes one by one, each new vertex being connected to m existing nodes, chosen using a linear preferential attachment rule, nodes with higher degrees having higher probabilities of being chosen. The resulting average degree is given by $2m$.

4. Full or sparse representation of the networks

Two main representations were used: the adjacency matrix and adjacency lists [15]. The adjacency matrix is a dense representation, in the sense that all possible edges in the network are explicitly included, with a value used to indicate

the presence of each edge, and another value used otherwise. Adjacency lists are sparse, as only the edges present in the network are incorporated. The adjacency matrix is usually implemented as a static structure, like the basic arrays used in most computer languages. On other hand, adjacency lists are implemented as dynamic structures and require pointers (a memory position pointing to another one).

For the elements of the adjacency matrix, we consider five possibilities, depending on the C language data type used for each element: double precision (`double`, 64 bits) and single precision (`float`, 32 bits) floating point number, integer numbers (`int`, 32 bits), Boolean values (which can assume only true and false values, 8 bits) and bits. This last element representation does not have a corresponding type in C, and was implemented using an `int` value to store 32 elements of the matrix, with bit manipulation operations used to access the individual bit values.

In the adjacency lists representation, a list is maintained for each vertex, with the numbers of the nodes that are neighbors to it. This representation uses an integer value for each neighbor and an overhead for list administration. Nevertheless, it spares memory space for sparse networks.

5. Algorithms

In this section we discuss briefly the algorithms used. We use N for the number of nodes and E for the number of edges in the networks. Also, let $\langle k\rangle = 2E/N$ be the average degree and $\langle k^2\rangle$ be the second momentum of the degree distribution.

ER (probability) network generation: For this model, all possible pairs of nodes need to be considered. Therefore, the time complexity is $\mathcal{O}(N^2)$ independent of representation.

ER (edges) network generation: In this case, for each edge to be generated, two endpoints must be chosen. Each generated edge must be tested to avoid multiple edges between the same pair of nodes. For the adjacency matrix representation, testing for the existence of an edge can be done by reading a matrix element, and therefore the complexity is $\mathcal{O}(E) = \mathcal{O}(N\langle k\rangle)$. In the case of adjacency lists, the list of neighbors of one of the endpoints of the possible edge must be searched in order to verify if an edge exists. As these lists have on average $\langle k\rangle$ elements, the complexity is $\mathcal{O}(E\langle k\rangle) = \mathcal{O}(N\langle k\rangle^2)$.

BA network generation: for each vertex added to the network, it is necessary to chose among all existing nodes with a probability proportional to their degrees. To achieve this, a list (represented by an array) is maintained where for a vertex i with degree k_i there are k_i entries with value i. Each time the degree of a vertex increases, a new entry is added. From this list, one value is randomly chosen to determine the vertex that will be connected to the new vertex. For each edge to be added, the amount of time taken to handle this auxiliary list is constant. The time complexity is therefore identical to that of the ER (edges) model.

Degree: The computation of the average degree can be done by counting the number of edges of each vertex. Using adjacency matrices, it is necessary, for each vertex, to scan the entire corresponding line (or column) in the matrix. That is, for each of the N nodes, N matrix elements must be considered, yielding a time complexity of $\mathcal{O}(N^2)$. When using adjacency lists, it is enough to count the number of elements in the list of each vertex. As there are on average $\langle k\rangle$ elements in these lists, the time complexity is $\mathcal{O}(N\langle k\rangle) = \mathcal{O}(E)$. This complexity can be further reduced to $\mathcal{O}(N)$ if the length of the list is stored inside the list during its construction, as done in the present work.

Clustering coefficient: For each vertex, we (i) find its neighbors; and (ii) verify if there is an edge between each pair of neighbors. Step (i) needs $\mathcal{O}(N)$ for the adjacency matrix representation and $\mathcal{O}(\langle k\rangle)$ (in average) for the adjacency lists representation. For step (ii), if the vertex has degree k, $k(k-1)/2$ pairs need to be tested; each pair can be tested in $\mathcal{O}(1)$ with the adjacency matrix and in $\mathcal{O}(\langle k\rangle)$ (in average) for adjacency lists. Therefore, the overall time complexity for computing the average clustering coefficient is $\mathcal{O}(N^2 + N\langle k^2\rangle)$ when using adjacency matrices and $\mathcal{O}(N\langle k^2\rangle\langle k\rangle)$ when using adjacency lists. We note that for the Erdős–Rényi models $\langle k^2\rangle$ is independent of N, while for the Barabási–Albert model it grows with $\log N$ [16]. In both models $\langle k^2\rangle$ grows with $\langle k\rangle^2$.

Average shortest path distances: Each vertex in turn is considered as the source of the paths and the distances from it to all other nodes are computed using the breadth-first search algorithm [15]. In this algorithm, we mark all neighbors of the source as having distance one, then consider the neighbors of the neighbors that have not already been visited and mark them as having distance two, and so on. For each source vertex, all other nodes must be visited once, and their neighbors considered. As computing the neighbors is $\mathcal{O}(N)$ in the adjacency matrix representation and $\mathcal{O}(\langle k\rangle)$ in the adjacency lists representation, the total time complexity of the average shortest distance computation is $\mathcal{O}(N^3 + N^2\langle k\rangle)$ when using adjacency matrices and $\mathcal{O}(N^2\langle k\rangle)$ when using adjacency lists.

Betweenness centrality: The betweenness centrality of all nodes is computed by using the algorithm of Brandes [17]. This algorithm is an extension of the breadth first search algorithm, and therefore has the same time complexity. Further details can be found in Ref. [17].

6. Results and discussion

We study the execution time needed for the generation of the network and for the computation of the following network measurements: average degree, clustering coefficient, all-pairs distances and betweenness centrality. The effect of the various network representations is evaluated as a function of the network size and average degree.

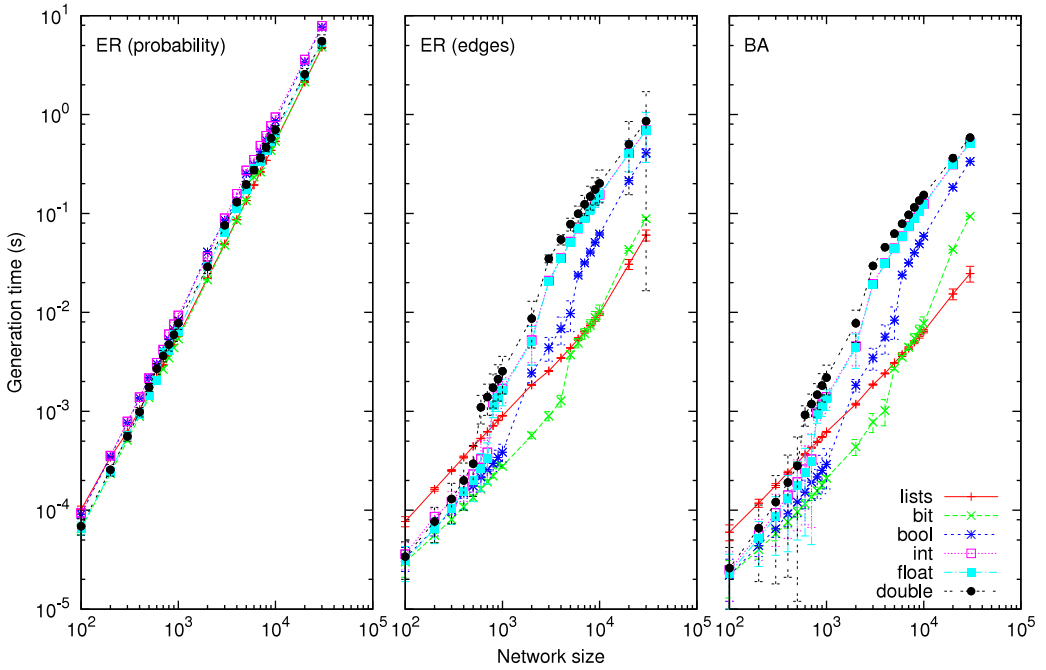

Fig. 1. Time taken to generate networks of different number of nodes (using $\langle k\rangle = 10$).

6.1. Network generation times for different network sizes

We first consider the effect of network size on the execution time needed for the generation of the networks, using different network representations.

6.1.1. ER (probability) model

We start by investigating how the generation of ER networks is affected by the choice of different graph representations. The execution times required to produce ER networks of several sizes, and average degree 10, are shown in Fig. 1.

Generally speaking, the different types of graph representation clearly had little effect on the execution time. This result is expected, considering the identical complexities $\mathcal{O}(N^2)$ of the various representations, and the fact that in all codes most of the computational effort is invested in considering all pairs of nodes to be connected with constant probability. A least-square fitting for the inclination of the lines gives a result of about 1.9 for the list representation and 2.0 for the other representations.

6.1.2. ER (edges) model

Unlike the results obtained previously, the adoption of different types of graph representations has a marked effect on the respective execution times for the ER (edges) model, as can be seen in Fig. 1. In particular, the improvements allowed by the more memory-effective representations (bit and list) are now evident. Interestingly, a sharp change of execution times in the matrix cases is observed at about $N = 1000$. This abrupt increase occurs when the capacity of the cache of the microcomputer is exceeded by larger sizes of graphs. Though the list representation is initially slower than the matrix cases, it becomes faster and faster with the increase of N.

As $\langle k\rangle$ is kept constant, the complexity for all representations is $\mathcal{O}(N)$. For small values of N, the fact that the adjacency lists representation has a complexity constant proportional to $\langle k\rangle^2$, while the constant for the adjacency matrix is proportional to $\langle k\rangle$, is shown by the larger generation times for the adjacency lists representation. As soon as the network size grows, the mentioned architectural effects (for large matrices, matrix representations take more memory space than list representations), not taken into account in a complexity analysis, dominate. This can be seen in the slopes of the power-law segments of the curves. The adjacency list representation curve has a slope of 1.0. For adjacency matrix representations, there are some segments of power-law behavior with fast transitions between them, and the slopes depend on the size of the matrix element. When using bits for the elements, the slope is about 1.0 in the region $N \leq 4000$ and 1.5 for $N \geq 5000$. If the elements are boolean, the approximate values of the slopes of the power-law regions are: 1.1 for $N \leq 1000$, 1.5 for $2000 \leq N \leq 5000$ and 1.8 for $N \geq 6000$. For `int` and `float` elements (both using 32 bits) the results are identical, with slopes of 1.2 for $N \leq 700$, 1.6 for $800 \leq N \leq 2000$ and 1.5 for $N \geq 5000$. When using `double`, the slopes are: 1.3 for $N \leq 500$, 1.7 for $600 \leq N \leq 2000$ and 1.4 for $N \geq 3000$.

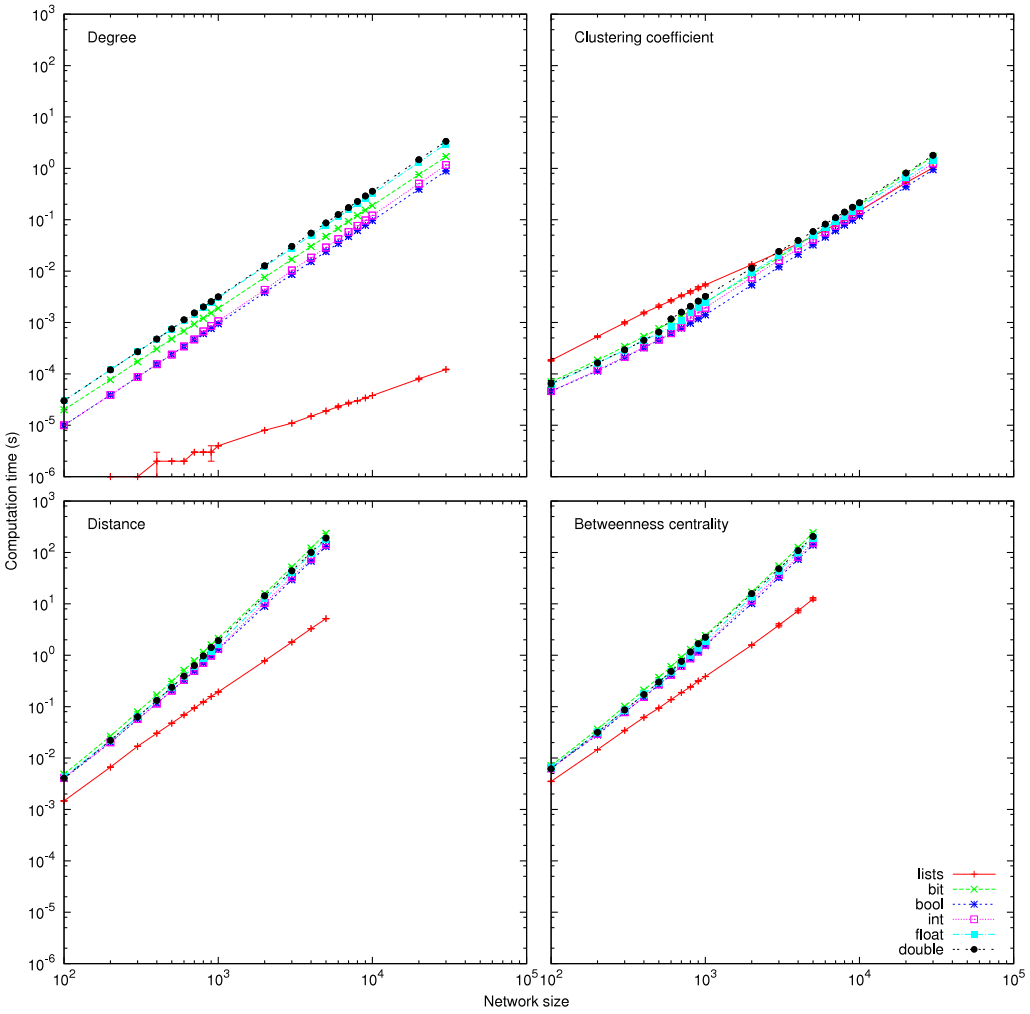

Fig. 2. Computation time for average degree, clustering coefficient, average distances and betweenness centrality as a function of network size for BA networks of average degree 10.

6.1.3. BA model

The generation times obtained for BA networks are also shown in Fig. 1. The results are similar to that of the ER (edges) model, with the list and bit representations providing the fastest execution times for large values of N, and similar architectural effects are displayed. The regions with power-law behavior are identical and the slopes do not differ much. For adjacency lists the slope is 1.0. Using bits: 1.1 in the region $N \leq 4000$ and 1.5 for $N \geq 5000$. Using `bool`: 1.1 for $N \leq 1000$, 1.7 for $2000 \leq N \leq 5000$ and 1.6 for $N \geq 6000$. Using `int` or `float`: 1.3 for $N \leq 700$, 1.7 for $800 \leq N \leq 2000$ and 1.4 for $N \geq 5000$. Using `double`: 1.5 for $N \leq 500$, 1.8 for $600 \leq N \leq 2000$ and 1.3 for $N \geq 3000$. It is interesting to note that, although the generation of ER (edges) and BA networks have identical complexities and the algorithm for the generation of BA networks is a bit more complicated, due to the necessity of maintaining a list of node connection (see Section 5), the execution times are a bit smaller for the BA networks. The reason is discussed in Section 6.3.1.

6.2. Computation time of some measurements for different network sizes

We turn now our attention to the effect of network size on the computation time of some network measurements, for the various network representations.

6.2.1. Average degree

We start by investigating the execution times required for determination of the average degree. Fig. 2 depicts the obtained results for BA networks with varying sizes and average degree equal to 10.

It is clear that the list implementation allows a dramatic reduction of the execution times for all network sizes. The other implementations required similar execution times and, as could be expected, the representation using matrices of

double precision values implied the longest execution times. It is also easy to see that all matrix representation cases have a complexity of $\mathcal{O}(N^2)$, as expected: the slopes of all curves are 2.0. The implementation with adjacency lists shows not only complexity $\mathcal{O}(N)$ (confirmed by the slope of 1.0 assumed for N larger than 1000), but dramatically reduced execution times with respect to the other representations.

6.2.2. Clustering coefficient

Fig. 2 shows the execution times obtained for the calculation of the clustering coefficient of BA networks of several sizes and average degree 10. As a consequence of the fact that this measurement demands more computations than the average degree, the execution times resulted larger. Interestingly, the several tested representations led to similar execution times, with the list and bool implementations providing particularly good efficiency for large values of N. Moreover, by keeping $\langle k\rangle$ fixed, the complexity of these calculations is $\mathcal{O}(N^2)$ for adjacency matrices and $\mathcal{O}(N \log N)$ for adjacency lists (and the BA model). The lower complexity using the adjacency lists representation can be seen on the graph. It also has significantly higher constants, as shown by the larger values for small N. The slope of the adjacency matrices curves is 1.9 for N greater than 500.

6.2.3. All-pairs distances and betweenness centrality

We also investigated how the time required for the calculation of the average distances and betweenness centrality varied with the several adopted implementations. Fig. 2 shows the obtained results for BA networks of several sizes and average degree equal to 10. The results for the two measurements are similar.

The substantially more complex nature of these measurements has been clearly reflected in the larger execution times. While little differences can be noticed for most implementations, the list representation allowed, again, substantially faster execution times, representing the fast option for all values of N. Indeed, the relative improvement obtained with lists clearly seems to increase with the network size. This implies that the use of lists becomes critical for allowing calculation of these measurements in particularly large networks.

These results are compatible with the respective complexity for fixed $\langle k\rangle$, which is $\mathcal{O}(N^3)$ for adjacency matrices (the slopes are of 2.7 for `bool` and `int` and 2.8 for the other representations) and $\mathcal{O}(N^2)$ for adjacency lists (slope of 2.1).

6.3. Network generation times for different average degrees

So far we have probed how the execution time varies with the network size for a fixed average degree (equal to 10). Now, we proceed to investigate how the speed is influenced by different values of average degree. This will allow us to get insights about the generality of the previously observed trends. In principle, it could be expected that the larger the average degree of a network, the smaller would be the benefits provided by the lists, because the matrices would become less sparse. Therefore, special attention is henceforth focused on this potential effect.

6.3.1. Network generation time

Fig. 3 shows the execution times, in terms of the average degree, obtained for generating networks of the three considered models while using the several representations. The network size is henceforth fixed at $N = 10\,000$.

The results are evident and confirm that the use of lists guarantees higher speed for sparse networks, decreasing steeply as the network becomes more dense. Particularly interesting is the behavior of the bits implementation, which overtakes the lists from average degree of the order of 10. This fact suggests that the execution time seems to be strongly affected by the memory which is demanded by each implementation. With the increase of the average degree, the matrix implementations become progressively more effective, while the bits, and particularly the list, implementations lose their effectiveness.

With respect to time complexity, as we are dealing with the constant N case, the ER (probability) model is $\mathcal{O}(1)$ for both representations (confirmed in the results) while the ER (edges) and BA models are $\mathcal{O}(\langle k\rangle^2)$ for adjacency lists (confirmed by a slope of 1.0 in the ER curve and 1.1 in the BA curve) and $\mathcal{O}(\langle k\rangle)$ for adjacency matrices. This last fact could only be confirmed for the curve corresponding to bit elements for the ER model, which has a slope of 2.2. Using larger elements, there is no clear power-law region, but the curves tend to follow the same slope for large $\langle k\rangle$.

Another interesting point shown in the results is that the generation of ER (edges) networks is much slower than that of BA networks for adjacency lists when $\langle k\rangle$ is large. This is somewhat surprising, as the algorithm to generate BA networks is more complicated. Some additional experiments have shown that, for large $\langle k\rangle$, the generation time for these models is dominated by the test to verify if a possible new link already exists. In the adjacency list representation, this involves searching for a node number in the list of neighbors of the other node. As this search is generally not successful, all of the list is searched. In the case of the ER (edges) model, as each link is independent of the other, a new list must be searched every time, implying a poor use of the processor cache. For BA networks, we simply search always the list of the new node that is being added, which results in better locality in the cache, and also in searching through a smaller list of neighbors.

6.4. Computation time of some measurements for different average degrees

Now we consider the effect of the average degree in the computation of some network measurements for different graph representations.

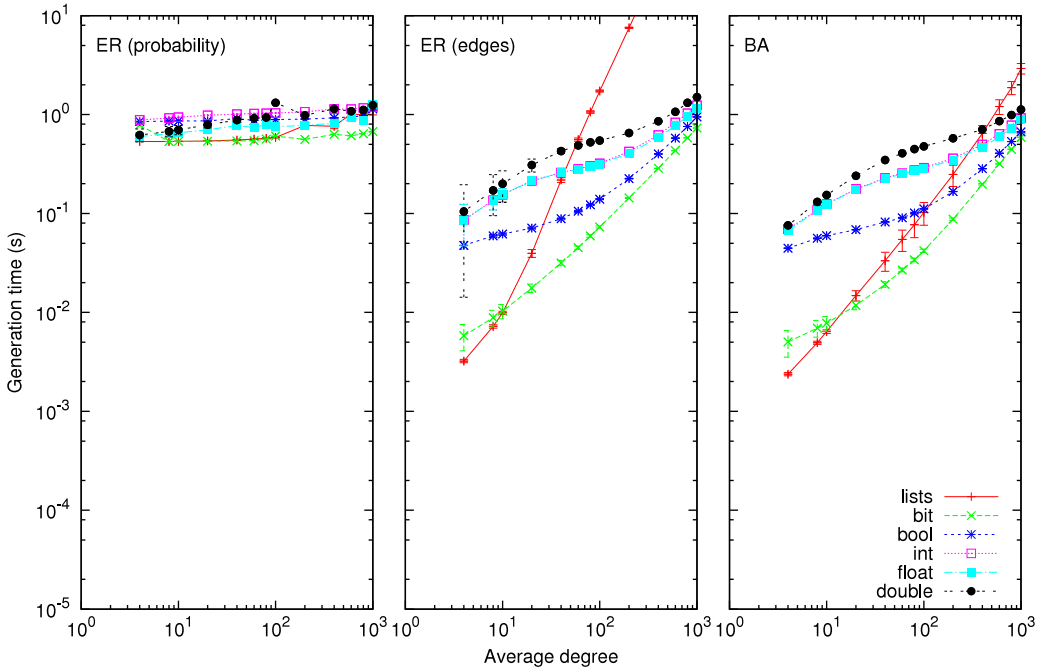

Fig. 3. Time taken to generate networks of different average degrees (using $N = 10\,000$).

6.4.1. Average degree

Fig. 4 depicts the execution time, in terms of the average degree, required to calculate the average degrees of BA networks with size $N = 10\,000$.

As expected by complexity analysis (Section 5), with fixed N all representations have constant time. But the better performance of the list representation is indisputable.

6.4.2. Clustering coefficient

The dependency of the execution times required for calculation of the clustering coefficient is shown in Fig. 4, in terms of several average degrees of BA networks.

The general trends verified in this graph are similar to those obtained for the networks generation and average degree calculation. However, the relative advantage of the list implementation is only present for networks with average degrees smaller than 10.

This behavior is explained by the fact that, for fixed N, the complexity of this computation is $\mathcal{O}(\langle k\rangle^3)$ for the list representation (the curve shows a slope of 3.0) and $\mathcal{O}(\langle k\rangle^2)$ for the matrix representation. For this last case, the curves show no clear power-law region, implying that architectural effects are dominating complexity issues.

6.5. All-pairs distances and betweenness centrality

The estimation of the shortest distances and the betweenness centrality in terms of the average degree are shown in Fig. 4. Both show similar behaviors.

From complexity analysis, it is expected that the execution times grows linearly with $\langle k\rangle$ (for fixed N) for all representations. Nevertheless, we see that for most of the values of $\langle k\rangle$ used, the matrix representations show almost constant time. The explanation is that this constant comes from the (large) N^3 term of the complexity expression, that is only overcome by the linear term when $\langle k\rangle$ is of the order of N. As expected, the list representation is more sensitive to increasing $\langle k\rangle$ and it demands increasing execution times for smaller values of the average degree. No clear power-law regions can be seen in these results. It is interesting to note that the transition from the almost constant time for small $\langle k\rangle$ to linear time for large $\langle k\rangle$ closely follows a q-exponential function of q-statistics [18]. As an example, we fit the execution time for the betweenness centrality computation using the bits representation to the following expression:

$$T(\langle k\rangle) = T_0\left(1 - (1-q)\frac{\chi}{\kappa}\right)^{1/(1-q)}. \tag{1}$$

The fitting can be seen in the inset of the betweenness centrality graph of Fig. 4, where the values $q = 0.56$, $T_0 = 116$, and $\kappa = 358$ were found by the least squares method.

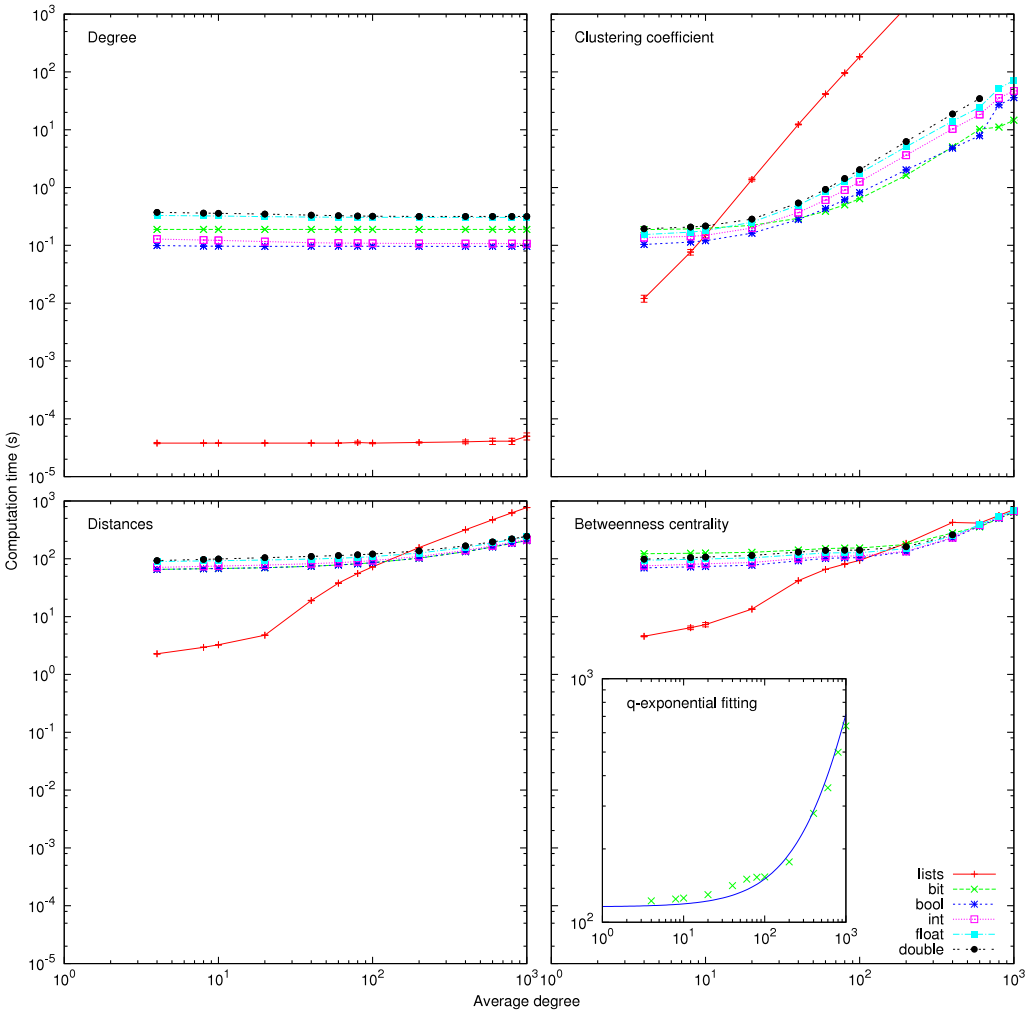

Fig. 4. Computation time for average degree, clustering coefficient, distances and betweenness centrality as a function of average degree for BA networks of size $N = 10\,000$.

7. Concluding remarks

Though the development of more effective algorithms for complex network generation and characterization can lead to great computational savings, we have shown that the choice of adequate network representations can have a major impact on the overall execution time. More specifically, we compared full and sparse schemes for representing the connectivity of the networks while generating networks and calculating several measurement of their topology. The sparse representation resulted as being generally more effective than the full scheme, except for the cases when the networks have very large average degrees. We also investigated the effect of having diverse data types such as byte, integer, float, double and bit. In general, the shorter data types led to superior performance as a consequence of the smaller amount of memory to be accessed.

The obtained results and trends suggest a number of further investigations. For instance, it would be interesting to consider other network models and measurements, as well as to assess the effect of different types of hardware, compilers and operating systems.

Acknowledgements

The authors thank CNPq [grants 301303/06-1 (LFC) and 306628/2007-4 (OMB)] and FAPESP [grants 573583/2008-0 (LFC) and 03/08269-7 (CAR, GT, LFC)] for financial support.

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C, 2nd ed., Cambridge University Press, 1992.
[2] N.J. Giordano, H. Nakanishi, Computational Physics, 2nd ed., Benjamin Cummings, 2005.

[3] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of Modern Physics 74 (2002) 48–98.
[4] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of networks, Advances in Physics 51 (2002) 1079–1187.
[5] M.E.J. Newman, Structure and function of complex networks, SIAM Review 45 (2) (2003) 167–256.
[6] L. da Fontoura Costa, F. Rodrigues, G. Travieso, P. Villas Boas, Characterization of complex networks: a survey of measurements, Advances in Physics 56 (2007) 167–242.
[7] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the Internet topology, Computer Communication Review 29 (4) (1999) 251–262.
[8] H. Jeong, S.P. Mason, A.-L. Barabási, Z.N. Oltvai, Lethality and centrality in protein networks, Nature 411 (6833) (2001) 41–42.
[9] S. Wasserman, K. Faust, Social Network Analysis, Cambridge University Press, Cambridge, 1994.
[10] J.L. Gersting, Mathematical Structures for Computer Science: A Modern Approach to Discrete Mathematics, 6th ed., W.H. Freeman, 2006.
[11] L.R. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics: A Foundation for Computer Science, 2nd ed., Addison-Wesley, 1994.
[12] J. Lerner, D. Wagner, K.A. Zweig (Eds.), Algorithmic of Large and Complex Networks, in: Lecture Notes in Computer Science, Springer, 2009.
[13] H.A. James, K.A. Hawick, Computational data structures for high-performance complex network-based small-world simulations. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.1903.
[14] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1997) 509–512.
[15] T.H. Cormen, C.E. Leierson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.
[16] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of Networks, Oxford, 2003.
[17] U. Brandes, A faster algorithm for betweenness centrality, Journal of Mathematical Sociology 25 (2001) 163–177.
[18] C. Tsallis, Possible generalization of Boltzmann–Gibbs statistics, Journal of Statistical Physics 52 (1988) 479–487.