Journal of Complex Networks (2013) **1**, 63–71
doi:10.1093/comnet/cnt005
Advance Access publication on 3 April 2013

Predicting efficiency in master–slave grid computing systems

G. Travieso*, C. A. Ruggiero, O. M. Bruno and L. da F. Costa
Instituto de Física de São Carlos, Universidade de São Paulo, Brazil
*Corresponding author: gonzalo@ifsc.usp.br

Edited by: Ernesto Estrada

[Received on 17 January 2013; revised on 1 March 2013; accepted on 4 March 2013]

This work reports a quantitative analysis to predicting the efficiency of distributed computing running in three models of complex networks: Barabási–Albert, Erdős–Rényi and Watts–Strogatz. A master–slave computing model is simulated. A node is selected as master and distributes tasks among the other nodes (the clients). Topological measurements associated with the master node (e.g. its degree or [betwenness] betweenness centrality) are extracted and considered as predictors of the total execution time. It is found that the closeness centrality provides the best alternative. The effect of network size was also investigated.

Keywords: grid computing; efficiency; topology; prediction.

1. Introduction

Despite all the scientific and technological advances in scientific computing made in the last decades, the demand on intensive number crunching remains as high as ever. This has been caused by at least the following three factors: (i) increasing amount of data being continuously produced in virtually all areas; (ii) the higher resolutions of such data, often involving 3D images and movies; and (iii) the focus on complex, non-linear systems that has underlain much of the basic and applied sciences. Given that the individual processing power has saturated [1], ever increasing attention has been placed on distributed and networked systems. One particularly attractive possibility, known as *grid-computing* involves the use of a large number of computers, duly connected to the Internet, as a massive computing resource [2]. The immediate advantage of such an approach is to provide the access to a large mass of already existent machines, without much additional cost for hardware or software. However, the performance of a grid system will depend strongly on the protocol for distributing the tasks, the topology of the interconnections, and the type of processing required. As a matter of fact, the execution of distributed tasks in a grid network is largely a complex problem directly related to the structure/function paradigm in complex networks [3]. Previous approaches at trying to understand the efficiency in grid computing include a study of the effect of network connectivity [4] and of geographical constraints [5] on a simple grid computing model and how topology affects applications with real-time constraints [6]. Despite such recent advances, the investigation of how the topology of the interconnections on the respective performance remains an open problem. In particular, it would be interesting to devise means to predict the execution time of grid computing systems from simpler network measurements, without the need of undergoing the whole process of real computations. The present work aims precisely at investigating how several types of networks measurements [7] can be used to foresee the execution time in grid computing systems as modelled and simulated in terms of model complex networks of varying topology.

2. Methodology

2.1 *Grid representation as a complex network*

In this work, computing grids are modelled as graphs where the nodes are computers and the edges are two-directional internet-like connections (e.g. TCP/IP). This gives rise to an undirected complex network with N nodes, which can be represented by a respective *adjacency matrix*, A. The element of the ith row and jth column of the matrix, denoted by A_{ij}, has value 1 if and only if there is a link between the ith and the jth node, and the value 0, otherwise.

The main issue in the current work is to choose a computer that will distribute the tasks to all other computers in such a way as to minimize the total execution time. It is natural to suppose that this optimum node should be, in some sense, *central* to the grid topology. With this in mind, the following metrics have been chosen in order to characterize the centrality of each node: *degree centrality*, *betweenness centrality*, *eigenvector centrality* and *closeness centrality* (related with the *average distance*). Another important measure was included: the local clustering coefficient. All these metrics can be derived from the adjacency matrix A.

Consider a node i. The degree centrality (also known as *node degree*) gives the number of adjacent links to i and is easily calculated by taking the number of non-zero elements in row i:

$$\sum_j A_{ij}. \tag{2.1}$$

The betweenness centrality tries to quantify how much a node is in the path connecting two other nodes. Considering all shortest paths connecting all pairs of nodes, the betweenness centrality of i is related to the number of these paths passing through i and defined as

$$\sum_{j,k \neq i} \frac{s_{jk}(i)}{s_{jk}}, \tag{2.2}$$

where s_{jk} is the total number of shortest paths between nodes j and k, and $s_{jk}(i)$ is the number of those paths that pass through node i. In this work, we used the algorithm of Brandes [8] for its computation. The eigenvector centrality quantifies the influence of a node as given by the influence of its neighbors. If λ_1 is the largest eigenvalue of the adjacency matrix A and v_1 is the corresponding eigenvector

$$Av_1 = \lambda_1 v_1 \tag{2.3}$$

the eigenvector centrality of node i is the ith component of v_1. The closeness centrality of node i quantifies how close it is to the other nodes in the network and is taken as the reciprocal of the mean of the distances from i to all other nodes:

$$\frac{N}{\sum_j d_{ij}}, \tag{2.4}$$

where d_{ij} is the distance between nodes i and j and is taken as the number of edges of the shortest path connecting them. Finally, the local clustering coefficient of i is defined as the number of links connecting its neighbors divided by the number of links necessary to fully connect all those neighbors (the number of links necessary to connect each and every pair of neighboring nodes), leading to a number between 0 and 1:

$$\frac{2e_i}{k_i(k_i-1)}, \tag{2.5}$$

where e_i is the number of links among neighbors of node i.

2.2 *Network models*

Three complex network models were chosen for this work: Erdős–Rényi (ER), Barabási–Albert (BA) and Watts–Strogatz (WS) [9].

The ER model uses a fixed number N of network nodes and probability p of creating a link between any two nodes.

The ER model is interesting for comparison and analysis but it lacks some properties found in real computing networks. Preferential attachment models such as the BA are more realistic, because they generate networks with degree distributions following a power law, very similar to those found in the real Internet [10]. Considering that many grid computing infrastructures are built upon the Internet or upon an Internet like network, the BA model is included in our analysis. BA networks are built by starting with a few unconnected nodes; for each new node created, m connections are generated from this node to others in such a way that nodes with higher degree are more likely to receive one of the m connections. The resulting network average degree will be equal to $2m$.

As another example of a network model that has different properties than the aforementioned, we chose the WS model which implements the 'small-world effect.' Short distances are frequently desirable in grids in order to minimize the communication latency among nodes. This model combines clustering coefficients that are higher than those in the BA and ER models with small distances. It starts with a regular structure such as a ring, with average degree k and then rewires some edges with uniform probability r. Larger values of r decrease clustering and average distances, but distances fall much faster. It is therefore possible, by adjusting r, to see the effect of high clustering and small distances simultaneously.

2.3 *Computational model*

The computational model used is the same introduced in Ref. [4]. The application works as master–slave, with a master having a set of tasks that are distributed to the slaves for processing. All tasks require the same amount of time to compute. After choosing one of the nodes as the master, all other nodes in the network are slaves. Figure 1 shows a diagram of the computational model, which illustrates the time necessary for the slaves to send a message to the master exemplified in Fig. 1(a) using dashed lines. The processing starts with the slaves sending a task request to the master (shown as a question mark in Fig. 1(b)). For each request received, the master sends a task to be computed (represented by the empty triangle). After the slave finishes computing the task, the result (black triangle) is returned to

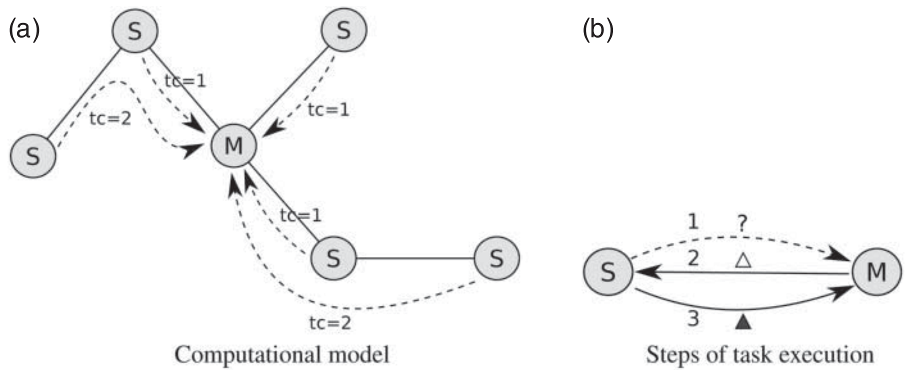

FIG. 1. (a) Diagram showing the communication cost in the computation model, each connection requires the same amount of time (one unit). (b) Steps of the model, (1) slave sends a task request to the master (question mark), (2) master sends a task to be computed (empty triangle) and (3) slave returns the result to the master (black triangle).

the master, which sends another task to the slave, if there are yet tasks to compute (see Fig. 1). When the results of all tasks are received by the master, the application stops.

We consider that the time to send a task or result through each link is the same, and use this as the time unit. We also assume that the time taken by the master to choose a task to send and by each node to route packets to a neighbor node are negligible. Also, all nodes have the same computational power.

At the start of the execution, a slave that lies at a distance d from the master will need to wait for a task for $2d$ time units (d units for its request to reach the master and another d for the task to arrive). Suppose the node takes c time units to compute the task. The master will then receive the result for the first task from this slave at $2d + c + d = 3d + c$. Subsequent task results will arrive from this slave at intervals of $2d + c$ (because the arrival of a result is a sign that the slave is idle, and the master can send a new task). If the total number of tasks is sufficiently large, all slaves at distance d from the master will receive approximately the same number of task to compute; we represent this number of tasks as $m(d)$, as it depends on the distance. They will therefore take a total time of

$$T(d) = d + m(d)(2d + c). \tag{2.6}$$

Again if the total number of tasks is sufficiently large, all slaves will end processing approximately at the same time, such that we can consider $T(d)$ in Equation (2.6) as the total execution time, but we will further simplify the expression because in this case $m(d)(2d + c)$ is large with respect to d, giving:

$$T = m(d)(2d + c). \tag{2.7}$$

If we have a total of M tasks to execute, we get

$$M = \sum_{d=1}^{d_{\max}} n(d)m(d) = \sum_{d=1}^{d_{\max}} \frac{n(d)T}{2d + c}, \tag{2.8}$$

where $n(d)$ is the number of nodes at distance d from the master and $d_{\max}$ is the maximum distance from the master for all nodes. Isolating T in Equation (2.8), we get

$$T = \frac{M}{\sum_{d=1}^{d_{\max}} (n(d)/(2d + c))}. \tag{2.9}$$

If we have $c \gg 2d$, Equation (2.9) reduces to

$$T = \frac{Mc}{N - 1}, \tag{2.10}$$

where N is the number of nodes in the network. This is the best possible result for this application, and is achieved independently from network topology. On the other hand, if $c \ll 2d$, most time is taken sending and receiving tasks, and it will make no sense to use a grid computing system (better results would be achieved by local execution of the tasks). The interesting values are then for $c \approx 2d$, where total execution time depends on topology. More specifically, the results depend on the hierarchical structure of the network [11]. The case $c \approx 2d$ is relevant because in order to employ the largest possible number of workers, it is necessary to divide the total work to be done in as small as possible tasks, while avoiding the case $c \ll 2d$ as discussed above.

3. Results and discussion

In order to quantify the overall processing performance, we adopt the concept of *efficiency* [12]. This measurement is calculated dividing the achieved speed-up by the ideal speed-up. Speed-up refers to the ratio between the total execution time with one processor and the total execution time using all the processors in the network. Therefore, the ideal speed-up is N, i.e. the number of available processors (nodes).

The performed experiments proceed in three stages. First, we compare the potential of several network topology measurements in predicting the grid efficiency for a BA configuration. Then, we investigate the effect of network size in the prediction. Finally, we consider other network models, more specifically the ER and WS models.

3.1 *Preliminary analysis*

In order to have an initial indication of how several measurements of the topology of the networks influence execution time, we considered a BA networks with $N = 1000$ and $\langle k \rangle = 6$ (the latter reflects the trend observed in the real internet [9]). The individual tasks to be performed were assumed to be 10 (in the chosen time unit that corresponds to the time to send a task through a network link). Each simulation involves 10000 tasks. We obtained the execution times for 30 network configurations, which had their topology characterized in terms of their average degree, clustering coefficient, betweeness, eigenvector and closeness centralities. Figure 2 shows the processing efficiency in terms of each of these measurements, with respective Pearson correlation coefficients collected in Table 1. It is clear that the closeness exhibited a significant correlation with efficiency and the largest Pearson coefficient when compared with the other measurements, suggesting that it can be used to provide accurate predictions of the processing efficiency. For this reason, we henceforth focus on the closeness in this work.

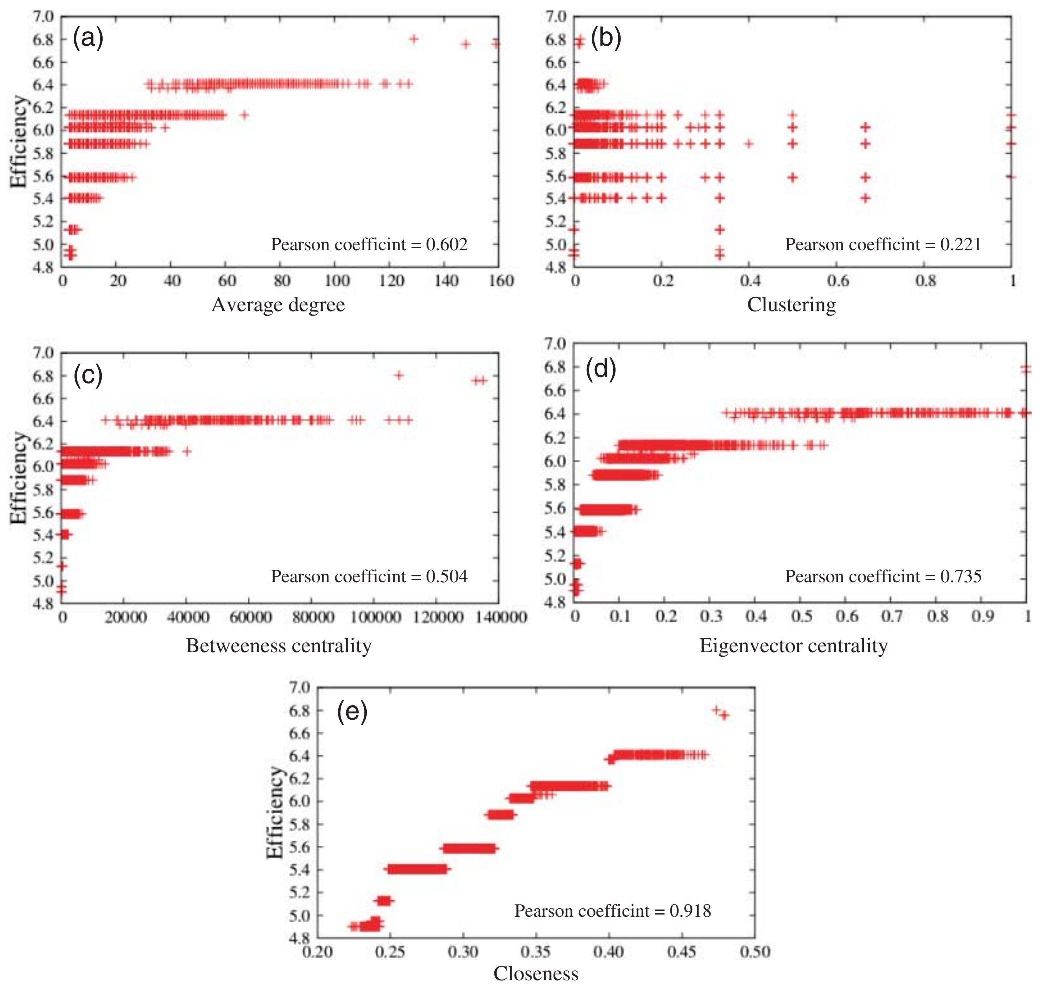

FIG. 2. Efficiency × various measurements for the BA model with 1000 nodes and average degree 6. Each graph shows the execution efficiency against a topological measurement considering each node (cross) as master. (a) Efficiency × average degree, (b) efficiency × clustering coefficient, (c) efficiency × betweeness, (d) efficiency × eigenvector centrality and (e) efficiency × closeness.

TABLE 1 *Average (30 samples) Pearson coefficient value*

Measure	Average Pearson
Degree	0.60
Clustering	0.22
Betweenness	0.50
Eigenvector	0.74
Closeness	0.92

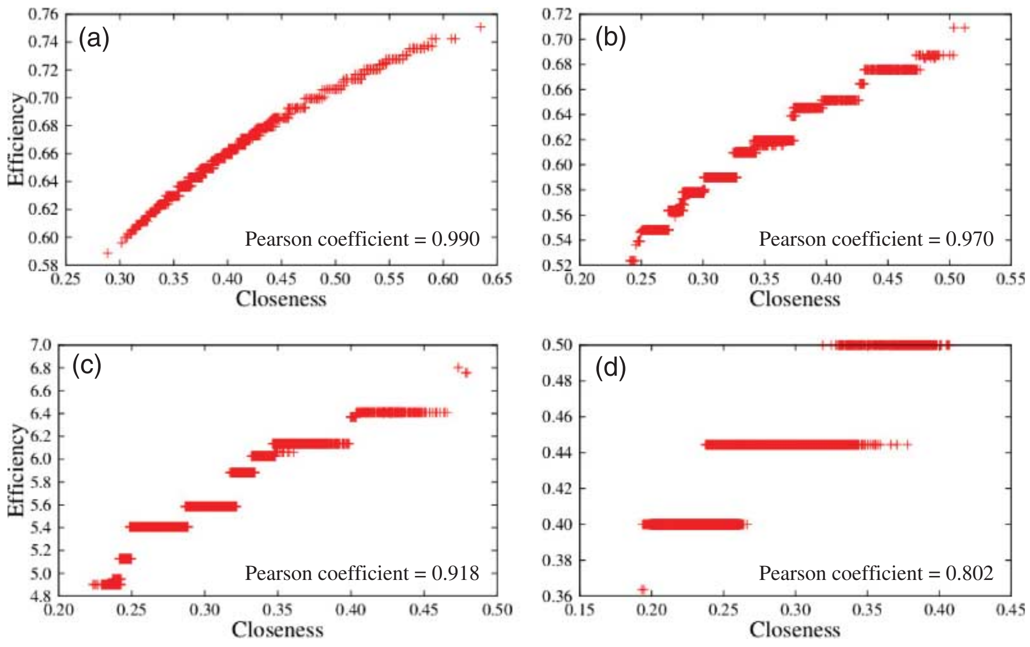

FIG. 3. Efficiency × closeness for various network sizes (BA model, $m = 3$). The graphs are constructed as explained in the caption of Fig. 2. (a) $N = 100$, (b) $N = 500$, (c) $N = 1000$ and (d) $N = 5000$.

3.2 *Effect of network size*

The previous experiment was performed with respect to BA networks with fixed sizes $N = 1000$. Now, we investigate the effect of N on the predictive power of closeness. Figure 3 shows these results. The Pearson correlation coefficient decreases monotonically with N, but remains high even for $N = 5000$. The quantization effect in these results are an artefact implied because all tasks have the same computational load, all nodes have the same computational power and all links transmit packets at the same rate. With our chosen parameter of 1 time unit for packet transmission and 10 time units for task computation, nodes that are at distance 1 from the master end their tasks at instants 13, 25, 37, 49, etc. while tasks from nodes at distance 2 end at instants 16, 30, 44, 58, etc. and from nodes at distance 3 at instants 19, 35, 51, 67, etc. When the total number of tasks is not large with respect to the number of nodes, only a few values of execution time are possible.

3.3 *Effect of network models*

In order to investigate the effect of network topology on the efficiency, we performed simulations for ER and WS (for several values of p) models for $N = 1000$ and $m = 3$. The results are shown in Fig. 4. The Pearson correlation coefficients resulted very high for all cases. The same quantization effect already commented above appeared for the ER and WS with $r = 0.1$. This is a direct consequence of the fact that these configurations have smaller diameter. The highest efficiency values have been observed for the ER configuration.

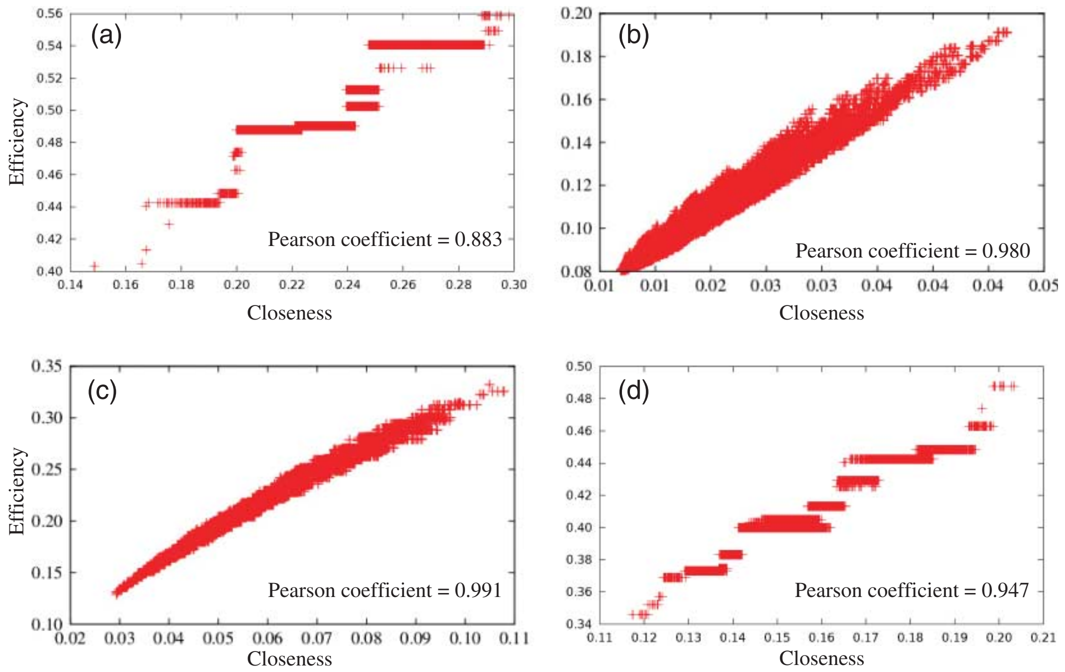

Fig. 4. Efficiency × closeness for ER and WS models. The average degree is 6 for all cases. The graphs are constructed as explained in the caption of Fig. 2. (a) ER $p = 0.006$, (b) WS $r = 0.001$, (c) WS $r = 0.01$ and (d) WS $r = 0.1$.

4. Conclusions

The proliferation of concurrent processing systems in the last decades as an alternative to achieving high processing speeds has emphasized the need not only to quantify the obtained improvements, but also to try to predict the respective performance. Because of its potential to harness available computing resources in a flexible and efficient manner, grid computing has been an important research topic in the last years. In this work, we set out to investigate how the efficiency of grid systems can be predicted while taking into account several topological measurements of the topology of the grid network. First, we quantified, through the Pearson correlation coefficient, the potential of several measurements from complex networks to predict the efficiency of BA networks, finding out that the closeness provided the best alternative. Then, we checked out the influence of the networks size (i.e. number of nodes) on the prediction of the efficiency. Despite a quantization effect appearing for large sizes, all tried situations yielded very high Pearson coefficients. Finally, we also found that extremely good predictions can be obtained even for other network topologies such as ER and WS. The high values of Pearson correlation coefficients obtained for most tested cases allow us to choose the best master node from which to distribute the task, i.e. the node with the lowest average distance to other nodes would provide the best choice for this finality.

It should be observed that the developments and results reported in this work, although related to grid computing, can be extended to many other real-world situations and systems. For instance, basic parts supply request by industries could be modelled by using our approach by understanding the production as a task and the requests as messages. A similar approach could be used for investigating task distribution among employees of a company or office.

Future studies could investigate task distribution schemes other than the master–slave configuration addressed in the present work.

Acknowledgements

G.T. is grateful to CNPq (grant number 308118/2010-3), C.A.R. to FAPESP (grant number 2003/08269-7), O.M.B. to CNPQ (grant numbers 308449/2010-0 and 473893/2010-0) and FAPESP (grant number 2011/01523-1) and L.F.C. to FAPESP (grant number 2011/50761-2) and CNPq (grant number 304351/2009-1) for the financial support.

References

1. Geer, D. (2005) Chip makers turn to multicore processors. *Computer*, **38**, 11–13.

2. Foster, J. & Kesselman, C. (eds) (2003) *The Grid 2: Blueprint for a New Computing Infrastructure*, 2nd edn. The Elsevier Series in Grid Computing. Las Altos, CA: Morgan Kaufmann.

3. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. & Hwang, D.-U. (2006) Complex networks: structure and dynamics. *Phys. Rep.*, **424**, 175–308.

4. da F. Costa, L., Travieso, G. & Ruggiero, C. A. (2005) Complex grid computing. *Eur. Phys. J. B*, **44**, 119–128.

5. de Angelis, A. F., Travieso, G. & Ruggiero, C. A. (2008) On the effects of geographical constraints on task execution in complex. *Int. J. Mod. Phys. C*, **19**, 847–853.

6. Travieso, G. & da F. Costa, L. (2011) Effective networks for real-time distributed processing. *J. Syst. Sci. Complex.*, **24**, 39–50.

7. da F. Costa, L., Rodrigues, F. A., Travieso, G. & Boas, P. R. V. (2007) Characterization of complex networks: a survey of measurements. *Adv. Phys.*, **56**, 167–242.

8. Brandes, U. (2001) A faster algorithm for betweenness centrality. *J. Math. Sociol.*, **25**, 163–177.

9. Newman, M. E. J. (2003) The Structure and Function of Complex Networks. *SIAM Rev.*, **45**, 167–256.

10. Faloutsos, M., Faloutsos, P. & Faloutsos, C. (1999) On power-law relationships of the Internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 251–262. New York: ACM.

11. da F. Costa, L. & Silva, F. N. (2006) Hierarchical characterization of complex networks. *J. Statist. Phys.*, **125**, 845–876.

12. Padua, D. (ed.) (2011) *Encyclopedia of Parallel Computing*. Berlin: Springer.