IJCSS International Journal of Complex Systems in Science

Int. J. Complex Systems in Science
vol. **5(1)** (2015), pp. 9–11

A Java API for the description of large complex networks under the object-oriented paradigm

A. Niño[1,†], C. Muñoz-Caro[1], S. Reyes[1] and M. Castillo[1]

[1] *SciCom Research Group. Escuela Superior de Informática de Ciudad Real, Universidad de Castilla-La Mancha. Paseo de la Universidad 4. 13004 Ciudad Real, Spain*

Abstract. At present, the network paradigm is of great interest for the structural and dynamic study of complex systems. Thus, systems are described as a set of, possibly, heterogeneous entities (nodes) interacting in, possibly, different ways (edges). The network approach is especially promising for the treatment of large sets of data (big data). Depending on the size and type of the problem, networks may require different computational approaches. For this reason, we have designed an object-oriented application programming interface, APINetworks, allowing modeling and treatment of general networks in arbitrary computational environments. In this work, we present a Java implementation of APINetworks. We determine its relative performance, for building and handling large networks, against a C++ APINetworks version and two standard tools in the field: NetworkX and JGraphT.

Keywords: Complex networks, API, Network generation, BFS
MSC 2000: 68N19, 05C85, 6804

† **Corresponding author:** alfonso.nino@uclm.es
Received: September 1st, 2015
Published: September 30th, 2015

1. Introduction

The last decade witnessed a great development of the structural and dynamic study of complex systems defined as a network of elements [1]. Therefore, systems can be described as a set of, possibly, heterogeneous entities (nodes) interacting in, possibly, different ways (edges). Depending on the size and type of the problem, networks may require different computational approaches. In addition, the current flood of data has triggered a huge interest in the treatment of very large networks. To tackle these problems a plethora of

software tools has been developed. However, no single solution exists allowing easy development of new algorithms for heterogeneous networks of varying size on all computational environments [2].

To deal with such situations, we have designed an Application Programming Interface (APINetworks) for the modeling and treatment of general networks in arbitrary computational environments [2]. The design uses an object-oriented approach. Thus, we apply inheritance and polymorphism for the modeling of static and dynamic networks and for the use of heterogeneous elements in the nodes, and heterogeneous interactions in the edges. In addition, this approach permits a unified treatment, transparent to the user, of different computational environments. In this work, we present a Java implementation of APINetworks. We determine its relative performance, for building and handling large networks, against the previous C++ APINetworks version and two standard tools in the field: NetworkX and JGraphT.

2. Performance results

The performance of APINetworks Java is tested against APINetworks C++ [2], NetworkX [3], in Python, and JGraphT [4], in Java. The tests are applied in two limiting cases. The first is the linear network, where each node is linked only to the previous one. The other case is the fully connected network. As tests, we use two different network operations. The first is a basic one: the construction of the network. The second operation is the Breadth First Search (BFS) [5] traversal of a network.

The system used in the tests is a Quad-Core AMD Opteron 2376 (2.3 GHz) with16 GB of core memory. This is a compute node of a cluster running under the Rocks 6.1.1 distribution (based on CentOs 6.5). The Java tests have been run with a maximum heap memory of 14GB and the concurrent garbage collector provided by the JVM.

For linear networks construction, APINetworks Java is the most efficient package followed by APINetworks C++, JGraphT, and, with great difference, NetworkX. Our Java API can build networks with as much as 75 million nodes (in 57 seconds) versus the 30 million of APINetworks C++ and the 60 million of JGraphT. For the largest APINetworks C++ and JGraphT networks built, our Java API is about 1.1 and 3 times faster. For the complete network case, APINetworks Java is again the most efficient tool followed by APINetworks C++, NetworkX, and, with very large difference, JgraphT. APINetworks Java handles networks with up to 20 thousand nodes (in 106 seconds) versus the 9 thousand of APINetworks C++ and NetworkX, and the 8 thousand of JGraphT. For the largest APINetworks C++ and NetworkX networks, our Java API is about 5 and 7 times faster, respectively.

With respect to the BFS traversal, in the linear case, APINetworks Java is the most efficient followed by JGraphT, APINetworks C++, and, last by a great margin, NetworkX. APINetworks Java needs 46 seconds to traverse the 75 million nodes network. Our Java API is about 4 and 7 times faster than JGraphT and APINetworks C++ for their largest networks. Finally, for the BFS traversal of complete networks the efficiency, in decreasing order, is: JGraphT, APINetworks Java, NetworkX, and APINetworks C++. Our Java API needs only 210 seconds to traverse the 20 thousand nodes complete network. In the case of the largest JGraphT network, 8 thousand nodes, this is about 4 times faster than APINetworks Java.

Acknowledgements

This work has been co-financed by FEDER funds and the Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha (grant # PEII-2014-020-A). The economic support of the Universidad de Castilla-La Mancha is also acknowledged.

References

[1] M. Newman, Networks: An Introduction. Oxford Univ. Press (2010).

[2] A. Niño,C. Muñoz-Caro and S. Reyes, *Comput. Phys. Commun.* doi: 10.1016/j.cpc.2015.05.017 (2015).

[3] http://networkx.github.io/; last access September, 2015

[4] https://github.com/jgrapht/jgrapht; last access September, 2015

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, 3td edition, The MIT Press, Cambridge, Massachusets, (2009).