International Journal of Scientific & Engineering Research, Volume 4, Issue 8, August 2013
ISSN 2229-5518

NetworkX and Matplotlib an Analysis

Yedhu Sastri, Kuttyamma A.J

Abstract— Today most of the networking computation and functions performed by the network administrators are very complex and requires high degree of programming skill. Python is a very powerful, open source, user friendly language that is widely popular today. NetwokX and Matplotlib are two packages in Python that provides wide range of functionalities and operations. NetworkX as its name indicates is very helpful in networking side. Matplotlib is mainly used for data visualization purpose. This paper is to give a basic idea about how the features of Python, NetworkX and Matplotlib can be used in solving Networking operations and Algorithms in an user friendly manner by reducing its complexity.

Index Terms—Python, NetworkX, Matplotlib, BFS.

1 Introduction

Network administrators and Network programmers have to face a wide variety of complex network operations and algorithms while dealing with computer networks. Solving these complex tasks is difficult and requires very good programming skill. It is very difficult for a person with less programming skill to do these tasks. Python is a good solution for this because it is a good user friendly language. It contains huge amount of inbuilt libraries and packages for making it more user friendly than other languages. In this paper Iam trying to familiarise you with Python and its packages NetworkX, Matplotlib and show how it can be used to make networking tasks and algorithms more user friendly in less complex manner. This paper is organised in such a way that First, familiarisation of Python and its various tools. Then a breif introduction to NetworkX its builtin models and functions.Then an introduction to Matplotlib and its functions. Next a case study by implementing BFS algorithm using Python-NetworkX, Matplotlib and evaluate it with BFS implementation in C. Aim of case study is to compare both language's in various aspects. By comparison iam trying to evaluate the features and functions that Python packages provides with respect to that of C.

2 Python

Python in an open source high-level programming language which is becoming widely popular day by day. There are so many reasons for its popularity. It is a simple and very user friendly language. Those who doesn't have very good programming knowledge, that is a beginner in programming can handle Python very easily. It contains a large amount of standard inbuilt libraries and packages. Almost for every purpose there exist a Python package, libraries or Python supporting applications. The main thing that makes Python more popular is its user friendly nature. It is well documented and supports proper indentation mechanism. User doesn't have to worry about the bracket intricacies. Python supports large data types and data structures. The usage of these data types and data structures are very simple while comparing to other languages.

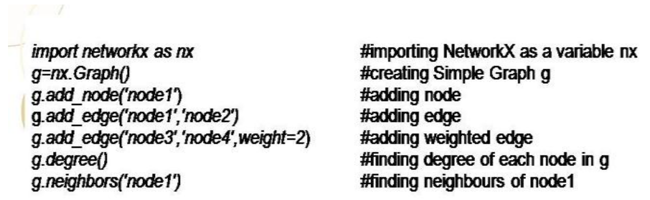
As mentioned earlier Python have numerous packages and tools that can be useful in various fields. Some important packages are; for Scientific computing NumPy is there. SciPy library which depends on NumPy is a tool for mathematical computation and operations. For networking operations and functions NetworkX is there. For visualisation of data Matplotlib package is there. For a Python interactive environment IPython is there. Web application frameworks like Pylon and Django are also very popular. Panda3D game engine also uses Python coding for game development. Like this a huge amount of libraries and packages are available in Python for various purposes. These packages enhances the power of programming and user friendly capabilities of Python. These features makes Python a very popular programming language. Major Python users today include Google, NASA, Disney, Youtube, Facebook, Bitbucket, Tabblo, Reddit etc. In this paper iam concentrating mainly on networking side so that I choose two tools NetworkX and Matplotlib for my analysis.

3 NetworkX

Today Network administrators and network programmers are facing various challenges in networking fields. They have to deal with huge datasets which are in different formats. They have to face new and complex algorithms. Integration of various networking tools is an important challenge they face today. Interfacing their networking applications with hardware's is another difficult task they face today. For solving majority of these problems they need a simple and powerful programming tool. Python is the best answer to their need. NetworkX in Python has huge support for network related operations and algorithms. It contains a large collection of network related Data structures, built in graph algorithms and graph generators. It also contains majority of the network related graphs as their inbuilt attributes. It may include Simple graph, Directed graph, Complete graph, Complete Bi-Partite graph, Petersen Graph etc. It also have

- *Yedhu Sastri is currently pursuing masters degree program in Network Engineering from MG University Kottayam, India, E-mail: yedhutechpaper @gmail.com*
- *Kuttyamma A J is Professor/Head of Department at Rajagiri School Of Engineering, Kakkanad,Kochi, India.*

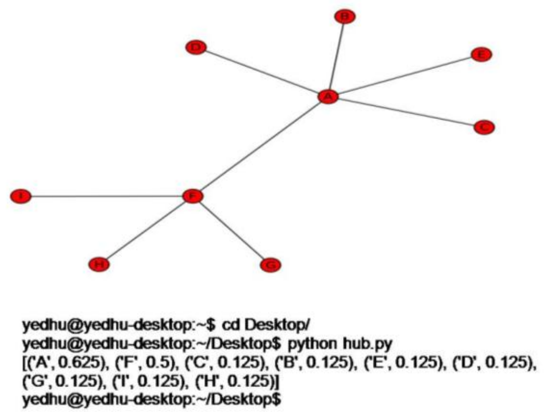
inbuilt support for Complex random Graph generators like Erdos-Renyi, Barbasi-Albert, Small world etc.



```
import networkx as nx                   #importing NetworkX as a variable nx
g=nx.Graph()                            #creating Simple Graph g
g.add_node('node1')                     #adding node
g.add_edge('node1','node2')             #adding edge
g.add_edge('node3','node4',weight=2)    #adding weighted edge
g.degree()                              #finding degree of each node in g
g.neighbors('node1')                    #finding neighbours of node1
```

Fig. 1. NetworkX Graph operations.

Above fig shows some basic graph operations using NetworkX. From the figure itself the user friendly nature of Python-NetworkX is evident. From this we understand that NetworkX operations are very simple.



```
yedhu@yedhu-desktop:~$ cd Desktop/
yedhu@yedhu-desktop:~/Desktop$ python hub.py
[('A', 0.625), ('F', 0.5), ('C', 0.125), ('B', 0.125), ('E', 0.125), ('D', 0.125),
('G', 0.125), ('I', 0.125), ('H', 0.125)]
yedhu@yedhu-desktop:~/Desktop$
```

Fig. 2. NetworkX Graph centrality function.

Figure 2 shows a graph implemented using NetworkX and the Degree centrality of each node in that graph. Hub in a network is usually node with high degrees. This Degree centrality function can be used to identify the hub in a network. There is a function in NetworkX to calculate the centrality of nodes in a graph. Degree Centrality of a node is nothing but number of nodes incident to that particular node. In NetworkX Degree centrality can be calculated as

>> nx.degree_centrality(G)

where G is the graph defined using NetworkX. If G is the graph in figure.2 then its output is as shown in figure. 2 with node A has highest degree of centrality. That is node A is acting as a hub in that network. Like this there are so many functions in NetworkX. By properly using these functions we can do different complex tasks in networks very easily.

4 Matplotlib

By using NetworkX we can create different types of graphs. But in order to view the created graphs another Python tool known as Matplotlib package is needed. Matplotlib is actually a Python library that is used for representing 2D graphs. Matplotlib is similar to that of Matlab and is mainly used for picture and data visualisation. Matplotlib.pyplot is used for plotting different figures and graphs. Matplotlib tool is a very useful tool for network administrators and network programmers to plot various data's related to networks. For example if we consider traffic monitoring in networks Matplotlib can be used for plotting network traffic in different nodes in a graphical manner. Matplotlib provides support for visualising data in different formats like Regular Plots, Scattered Plots, Bar Plots, Grid Plots, 3D Plots, Pie Charts, Polar Axis, Texts etc. Basic way of displaying a graph created by NetworkX, using Matplotlib is as shown below

>> import matplotlib.pyplot as plt
>> networkx.draw(G)
>> plt.show()

Here first we import matplotlib.pyplot as plt a varaiable. This is actually the procedure of importing matplotlib package into Python. Then we will draw the graph G by using the draw function of NetworkX. Then by using the show function of matplotlib.pyplot we will display the graph G in output screen. The above code shows the simple and user friendly way of Python code, instead of writing so many codes for representing a graph it can be done in limited steps by using Python.

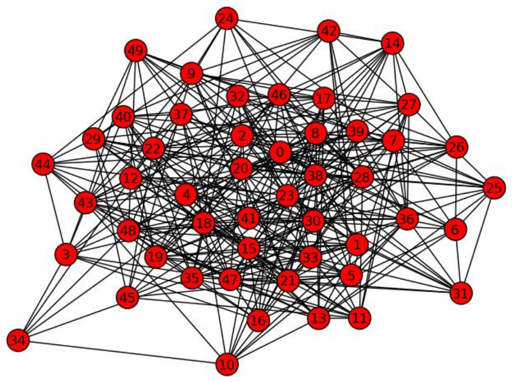


Fig. 3. Erdos_Renyi_Graph

Figure 3 shows a random graph generated by Erdos_Renyi model. Think how many lines of code is needed to generate this complex graph in normal scenario using C language. We know its a very complex task to generate this graph using C. But with Python-NetworkX and Matplotlib it requires only a code of 5 lines as shown below

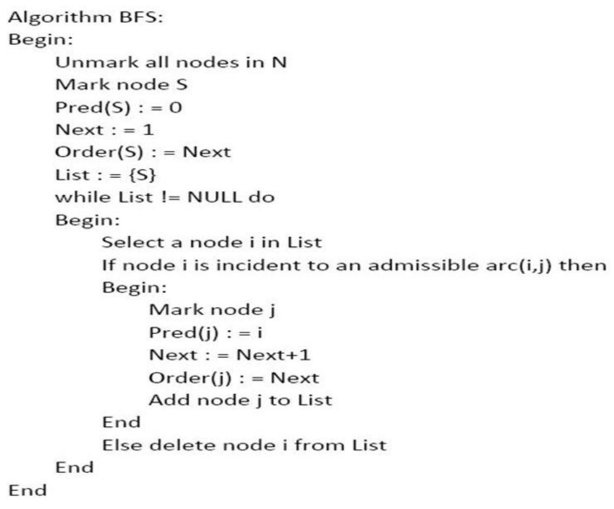
>> import networkx as nx
>> import matplotlib.pyplot as plt
>> G=nx.erdos_reny_graph(50,0.3)
>> nx.draw(G)

>> plt.show()

This is actually the simplicity of Python. In NetworkX there exists a built in function erdos_reny_graph for generating random graphs using method of erdos_reny model. Thus by using NetworkX and Matplotlib packages in Python most of the complex tasks and algorithms in networks can be easily implemented in a user friendly way. This is the reason why these two packages are considered as an important networking tools. It is better for a network admin or network programmers if they are familiar with Python and these tools in Python. Because it will reduce the complexity of their tasks and save their time to a great extend.

5 CASE STUDY : EVALUATION

Aim of this case study is to compare and evaluate the features of Python implemented networking tasks with that implemented by other popular language like C. For this case study we choose BFS algorithm, implemented it by using Python-NetworkX-Matplotlib and also by using C and then compare their features with respect to various aspects. Through this evaluation my aim is to identify how Python enhances the implementation of a network related task and how it reduces the complex nature of that implementation by comparing with other popular language like C. BFS is nothing but Breadth First Search and it is a popular technique used for searching in a graph. In BFS we will first visit a node then we visit its neighbouring nodes and proceeds like that. We choose BFS implementation for this evaluation mainly because it is widely used in networking for finding shortest path between

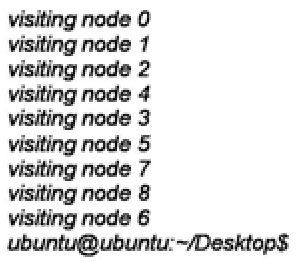


```
Algorithm BFS:
Begin:
    Unmark all nodes in N
    Mark node S
    Pred(S) : = 0
    Next : = 1
    Order(S) : = Next
    List : = {S}
    while List != NULL do
    Begin:
        Select a node i in List
        If node i is incident to an admissible arc(i,j) then
        Begin:
            Mark node j
            Pred(j) : = i
            Next : = Next+1
            Order(j) : = Next
            Add node j to List
        End
        Else delete node i from List
    End
End
```

Fig. 4. BFS algorithm

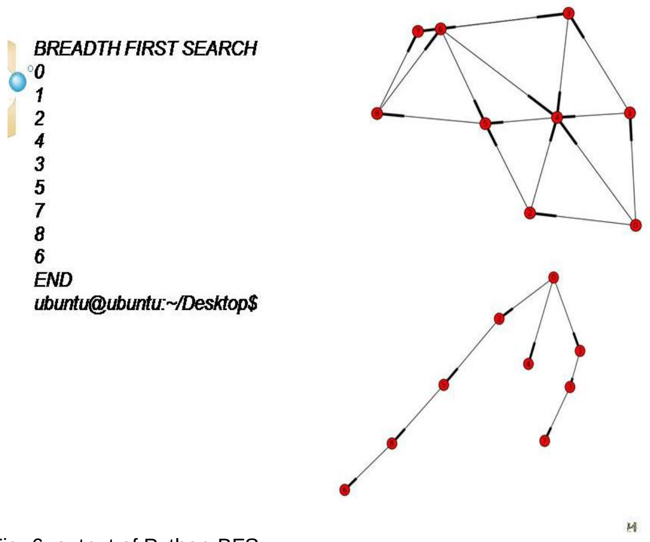
two nodes, maximum flow in a network, finding all nodes with one connected component etc.

Figure 4 shows the BFS algorithm with time complexity O(m) where m is the number of arcs. We implemented this algorithm by using both Python-NetworkX-Matplotlib and by using C. Implemented BFS is for a graph with 9 vertices and 17 edges. Then we made a detail evaluation based on different aspects of these to programs.



```
visiting node 0
visiting node 1
visiting node 2
visiting node 4
visiting node 3
visiting node 5
visiting node 7
visiting node 8
visiting node 6
ubuntu@ubuntu:~/Desktop$
```

Fig. 5. output of C-BFS



```
BREADTH FIRST SEARCH
0
1
2
4
3
5
7
8
6
END
ubuntu@ubuntu:~/Desktop$
```

Fig. 6. output of Python-BFS

First evaluation is based on their outputs, from figure.5 and figure.6 it is very clear that BFS implemented with Python is very user friendly and it is very easy to understand. Figure. 6 shows both the input graph and BFS traversed output graph along with the breadth first search nodes. From this we can understand that Python implemented network algorithms are very simple and user friendly. By viewing the graphs the programmer can easily visualise the networking architecture similar to the graph and easily find solutions. Then Second evaluation is based on length of code. For Python implementation we took about 41 line code but for C implementation we took about 96 lines. The length of the program varies according to your program skills. But one thing iam sure is that Python requires code with less number of lines than C. This shows that Python coding is less time consuming compared to C. Thus it saves the valuable time of network programmers while coding various tasks.

TABLE 1
GRAPH CREATION TIME

No:of nodes	***Python-BFS***	***C-BFS***
10	0.000056982 s	0.000102 s
20	0.00007987 s	0.000105 s
30	0.000094891 s	0.000109 s
50	0.000133991 s	0.000111 s
100	0.000240087 s	0.000138 s

Third evaluation is based on node creation time. Table 1 shows the graph creation time in BFS using Python and C. From the table we can see the node creation time of Python is lower than C when number of nodes is less. But for large number of nodes Python creation time is higher than C. This is because Python creates nodes of graph by loading the package NetworkX and for higher nodes this takes higher time than that of C, but C creates graph by using inbuilt data type structure and it doesnt take that much time in the case of large number of nodes. Python is actually working as an interpreted language but we can compile the Python code and can generate a compiled Python code which doesn't require compilation during each time of its execution. The extension of that code is pyc instead of normal py. This compiled code took very little time compared to the normal execution of Python. Here we executed pyc code.

From this case study it is clear that programming or implementing algorithms in Python is very easy, user friendly and saves our coding time compared to that of other language like C. One drawback it points is Python running time is slightly higher than that of C. If network programmers practise to use the Python language it is very useful and very advantageous for them. Python can be used for implementing very complex algorithms in an easy manner, Python reduces the complexity of programming, Python scripting language can be used for integration of different tools like mashups. Thus Python and Python networking tools like NetworkX and Matplotlib is a best option for network administrators and network programmers.

6 Conclusion

This paper is to give a brief idea about how Python and Python packages like NetworkX and Matplotlib is useful for network programmers and administrators. Python is a rapidly growing language and big corporates in the world are changing towards Python. Thus in future Python have a very good scope in different fields especially in the networking field. As it is an open source language we can also contribute different tools and libraries to enrich its vault. Thus by using the user-friendly, simple, less complex, easy to understand features of Python very complex and time consuming tasks and codes in networking can easily be implemented.

References

[1] Fernando Perez, "*Python: An Ecosystem for Scientific Computing*," Computing in Science & Eng. 2011

[2] T.Oliphant, "*Python for Scientific Computing*," Computing in Science & Eng., vol. 9, no. 3, 2007, pp. 10–20

[3] A.A Hagberg, D.A. Schult, and P.J. Swart, "*Exploring Network Structure, Dynamics, and Function Using NetworkX*," Proc. 7th Python in Science Conf., SciPy Community, 2008; http://conference.scipy.org/proceedings/SciPy2008/paper_2..

[4] F. Pérez and B.E. Granger, "*IPython: A System for Interactive Scientific Computing*," Computing in Science & Eng., vol. 9, no. 3, 2007, pp. 21–29.

[5] Dr. Derek Greene, "*Graph and Network Analysis*",Clique research cluster, UCD 2011.

[6] Carlos Raniery Paula dos Santos, Rafael Santos Bezerra, João Marcelo Ceron,Lisandro Zambenedetti Granville, and Liane Margarida Rockenbach Tarouco, "*On Using Mashups for Composing Network Management Applications*" Federal University of Rio Grande do Sul, 2010.

[7] D.M. Beazley, "*Automated Scientific Software Scripting with SWIG*," Future Generation Computing Systems, vol. 19, no. 5, 2003, pp. 599–609.

[8] D.M. Beazley, "*Python Essential Reference*", 4th ed.,Addison-Wesley, 2009

[9] D. Ascher et al., Numerical Python, tech. report UCRL-MA-128569, Lawrence Livermore Nat'l Lab., 2001; http://numpy.scipy.org.