ISBN : 978-979-16353-8-7

T-19

PEMANFAATAN NETWORKX UNTUK MENGEKSPLORASI DAN MENGANALISA JARINGAN BESERTA SIFAT/KARAKTERISTIKNYA

Nur Insani, M.Sc

Jurusan Pendidikan Matematika FMIPA UNY
nurinsani2001@yahoo.com

Abstrak

Aplikasi analisis jaringan telah banyak ditemukan di berbagai penelitian ilmiah jaringan transport, jaringan sosial, analisis trafik internet, jaringan penyebaran virus baik di bidang kesehatan maupun teknologi informasi, serta jaringan hubungan protein dan gen di bidang kesehatan dan biologi. Kemajuan teknologi yang sangat pesat telah memotivasi para peneliti di bidang kombinatorial optimisasi untuk mempelajari jaringan dengan skala besar. *NetworkX* merupakan suatu paket perangkat lunak berbahasa Python yang dapat digunakan untuk membentuk, memanipulasi dan mempelajari struktur, dinamika serta fungsi dari jaringan kecil hingga jaringan yang rumit/besar. Makalah ini membahas tentang berbagai bentuk dari jaringan (atau graf) yang dapat dieksplorasi dan dianalisa dengan *NetworkX*, serta algoritma yang dapat diimplementasikan untuk menghitung beberapa sifat atau karakteristik penting dari graf, khususnya pada graf-graf yang digunakan dalam berbagai penelitian optimisasi *(benchmark instances)*.

Kata kunci: *NetworkX*, jaringan, graf, *instance features*.

PENDAHULUAN

Salah satu kemajuan besar dalam teori jaringan pada saat ini, yang dikombinasikan dengan kemampuan untuk mengambil data jaringan berskala besar, telah meningkatkan minat untuk mengeksplorasi dan menganalisis jaringan berskala besar. Aplikasi analisis jaringan telah banyak ditemukan di berbagai penelitian ilmiah dan berteknologi seperti jaringan hubungan protein dan gen di bidang ilmu biologi, struktur Web Graph, analisis trafik internet, dan jaringan sosial dan berkolaboratif termasuk jaringan kontak untuk penyebaran virus. Pada area-area diatas dan area lainnya, perangkat lunak spesial telah tersedia untuk menyelesaikan masalah-masalah komputasi tersebut, akan tetapi hanya beberapa yang tersedia secara bebas dan terbatas untuk suat masalah tertentu.

Pada tahun 2002, Harberg dkk [1] mulai mengembangkan membangun suatu perangkat lunak, *NetworkX*, untuk menganalisa data dan strategi intervensi untuk epidemic penyebaran penyakit [2]. Tujuan awal mereka adalah untuk membangun suatu alat *open source* yang mudah untuk dikembangkan bagi pengguna multidisiplin dimana pengguna atau peneliti tidak harus ahli dan menguasai bahasa pemrograman yang rumit. Selanjutnya, *NetworkX* dikembangkan untuk mengisi kebutuhan akan perangkat lunak yang dapat menganalisa jaringan atau graf secara umum.

Makalah dipresentasikan dalam Seminar Nasional Matema ka dan Pendidikan Matema ka dengan tema " ***Kontribusi Pendidikan Matematika dan Matematika dalam Membangun Karakter Guru dan Siswa***" pada tanggal 10 November 2012 di Jurusan Pendidikan Matema ka FMIPA UNY

NetworkX merupakan suatu alat untuk menganalisa jaringan yang bersifat fleksibel dan ditulis dalam bahasa pemrograman Python. Program ini menyediakan struktur dasar jaringan atau graf untuk representasi suatu jaringan/graf dalam berbagai bentuk seperti graf sederhana, graf berarah maupun graf dengan loop dan busur-busur yang parallel. Struktur jaringan direpresentasikan dalam kumpulan busur (koneksi/hubungan/ikatan/*edges*) yang menghubungkan kumpulan simpul (titik/tempat/actor/*nodes*). Dalam hal ini, sebarang obyek dapat dianggap sebagai simpul dan dapat dihubungkan dengan sebarang obyek lain dengan busur-busur. Hal ini merupakan suatu keuntungan bagi pengguna karena struktur jaringan dapat dapat dimodifikasi dengan obyek dan struktur data yang sesuai dengan kebutuhan pengguna, dengan cara melengkapi kode yang sudah ada sebelumnya dan memungkinkan analisa jaringan tanpa harus mengembangkan lebih lanjut perangkat lunak yang digunakan. Setelah jaringan terbentuk sebagai obyek dalam *NetworkX*, maka peneliti dapat menganalisa struktur jaringan menggunakan algoritma-algoritma yang tersedia dalam *NetworkX*, seperti mencari distribusi derajat (jumlah busur yang menghubungi suatu simpul), properti atau sifat dari suatu jaringan/graf (seperti koefisien kluster, kepadatan, keterhubungan), jalur terpendek (*shortest path*) dan sebagainya.

Dalam makalah ini, penulis mendeskripsikan secara singkat tentang *NetworkX* beserta beberapa contoh untuk mendemonstrasikan beberapa struktur jaringan yang dapat direpresentasikan oleh *NetworkX*. Pada bagian selanjutnya akan dibahas beberapa algoritma yang dapat diimplementasikan untuk menghitung beberapa properti atau sifat penting dari suatu jaringan/graf yang dapat dieksplor menggunakan *NetworkX*, yang kemudian dapat properti-properti tersebut dapat digunakan untuk menganalisa jaringan.

PEMBAHASAN

NetworkX

NetworkX merupakan suatu paket dari Python untuk membentuk dan memanipulasi segala bentuk graf dan jaringan. Adapun paket program, tutorial beserta dokumentasi dari program ini tersedia di http://networkx.lanl.gov. Untuk mendukung analisa jaringan/graf, maka *NetworkX* berinteraksi dengan sejumlah paket Python lainnya seperti NumPy, SciPy dan Matplotlib, dimana paket-paket tersebut juga merupakan paket yang tersedia secara bebas (*open source*).

NetworkX menyediakan bermacam kelas untuk merepresentasikan jaringan/graf berarah maupun tidak berarah, berbobot atau dengan *self loops*. Program ini juga mengijinkan untuk membangun graf yang memuat busur ganda (*multiple edges*). Manipulasi dasar pada jaringan/graf seperti menambah simpul atau busur tersedia dalam metode-metode kelas, sedangkan untuk operasi statistik dan algortima yang lebih kompleks seperti properti/sifat (*clustering*, *eigenvector centrality*), jarak terpendek, dan visualisasi tersedia pada *NetworkX* dalam bentuk fungsi paket.

Suatu jaringan/graf yang paling sederhana dapat dibentuk dengan menggunakan kelas *Graph*. Untuk menambahkan simpul-simpul sesuai dengan kebutuhan, maka dapat digunakan algoritma sebagai berikut:

```
>>> import networkx as nx
>>> G = nx. Graph()
>>> G.add_node(1)
>>> G.add_nodes_from([1,2])
>>> G.add_node('Yogya')
>>> print G.nodes()
[1, 2, 3, 4, 'Yogya']
```

Sedangkan untuk menambah suatu busur (atau beberapa busur sekaligus) yang menghubungkan satu simpul ke simpul lainnya, dapat digunakan algoritma sebagai berikut:

```
>>> G.add_edge(1, 2)
>>> G.add_edge(4, 'Yogya')
>>> G.add_edges_from([(2,4),(1,3), (3,5), ('Yogya',5)])
>>> print G.edges()
[(1, 2), (1,3),(2, 4), (3, 5), (4, 'Yogya'), (5, 'Yogya')]
```

Jika terdapat busur yang menghubungkan simpul yang belum terdefinisi sebelumnya, maka *NetworkX* secara otomatis akan menambahkan simpul tersebut pada jaringan atau graf yang ada. Untuk mengetahui jumlah simpul atau jumlah busur dari jaringan/graf yang dibentuk, maka dapat digunakan perintah berikut:

```
>>> G.nodes()
[1, 2, 3, 4, 5, 'Yogya']
>>> G.edges()
[(1, 2), (1,3),(2, 4), (3, 5), (4, 'Yogya'), (5, 'Yogya')]
>>> G.number_of_nodes()
6
>>> G.number_of_edges()
4
```

Dengan cara yang sama, kita dapat pula menghapus suatu simpul ataupun busur dari jaringan/graf yang telah dibentuk:

```
>>> G. remove_node (1)
>>> G. remove_nodes_from ([1 ,2])
>>> G. remove_edge (1 ,2)
>>> G. remove_edges_from ([(1 ,2) , (2 ,3) ])
```

Untuk mengetahui derajat tiap simpul dari jaringan yang dibentuk, maka dapat digunakan metode *degree()* sebagai berikut:

```
>>> G.degree()
[1, 2, 2, 1, 1, 1]
```

Jika kita ingin mengetahui derajat tiap simpul beserta rata-rata derajat dari simpul tetanga yang terdekat dari simpul, maka dapat digunakan algoritma berikut:

```
>>> for node in G.nodes():
    if G.degree(node)>0:
        cdeg = 0
        for neighbor in G.neighbors(node):
            cdeg += G.degree(neighbor)
        print node, G.degree(node), float(cdeg)/G.degree(node)
```

dan hasilnya yaitu:

```
1 1 2.0
2 2 1.5
3 2 1.5
4 1 1.0
5 1 2.0
Yogya 1 1.0
```

NetworkX telah menyediakan pula beberapa generator atau pembangkit untuk bermacam-macam model graf klasik (seperti graf bintang, graf *Null*, graf *Lollipop*, graf roda, dsb), graf kecil (seperti graf *Chvatal*, *Krackhardt Kite Social Network*, graf *Petersen*, dsb), grad random (seperti graf *Newman-Watts-Strogatz*, graf *Watts-Strogatz*, graf binomial, graf *Erdős-Rényi*, graf berarah, graf bipartite, graf ego, graf stokastik, dan jaringan sosial. Generator-generator tersebut dapat pula dimodifikasi sesuai dengan kebutuhan dalam penelitian.

Berikut merupakan suatu contoh algoritma untuk membangun suatu model jaringan yang telah tersedia di dalam *NetworkX* dengan ukuran jaringan yang cukup besar:

```
>>> N = 1000  #jumlah simpul
>>> k = 12    # jumlah simpul tetangga yang terdekat
>>> p = 0.05  #peluang busur terhubung
>>> G_ws = nx.watts_strogatz_graph(N,k,p)
>>> G_er = nx.erdos_renyi_graph(100, 0.1)
```

Dalam penelitian optimisasi, terdapat banyak algoritma untuk menentukan jalur terpendek pada jaringan/graf berbobot (*weighted*). Berikut merupakan suatu contoh perhitungan untuk menentukan jalur terpendek berbobot dari suatu jaringan sederhana dengan 4 busur menggunakan algoritma Dijkstra:

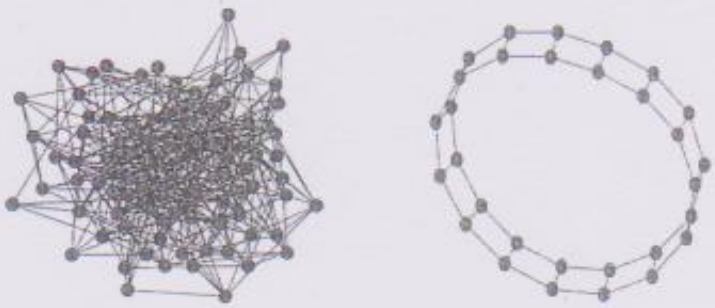
```
>>> G2 = nx.Graph()
>>> e = [('a','b',0.3),('b','c',0.9),('a','c',0.5),('c','d',1.2)]
>>> G2.add_edges_from(e)
>>> print nx.dijsktra_path(G2,'a','d')
['a', 'c', 'd']
```

NetworkX menggunakan paket *Matplotlib* untuk menvisualisasi jaringan/graf yang terbentuk. Dari contoh diatas, maka dengan algoritma berikut akan diperoleh visualisasi graf:

```
import matplotlib.pyplot as plt
>>> plt.figure(figsize =(10,10))
<matplotlib.figure.Figure object at 0x02874E70>
>>> nx.draw(G)
>>> plt.axis("tight")
(-0.088157533407211316, 0.9277530896663666, -0.10500000000000001, 1.105)
>>> plt.show()
```

Gambar 1: Graf G dengan 6 simpul

Dengan cara yang sama, maka akan diperoleh visualisasi dari graf *Erdős-Rényi* dengan jumlah simpul 100 dan graf tangga dengan jumlah simpul 30.



Gambar 2. Graf *Erdős-Rényi* dengan 100 simpul dan graf tangga dengan 30 simpul

Analisa Jaringan/Graf dengan NetworkX

Dalam analisis sebuah jaringan/graf dengan menggunakan terdapat beberapa ukuran dasar yang menjadi titik tolak perhitungan matematis untuk mengetahui pola keterhubungan dalam jaringan/graf tersebut. Ukuran dasar ini merupakan properti atau sifat dari suatu jaringan/graf, antara lain: besar jaringan/graf (*network size*), derajat (*degree*), kepadatan (*density*), ketergapaian (*reachability)*, keterhubungan (*connectivity*), jarak (*distance*), dan jalur (*flow*) informasi. Besarnya jaringan/graf sangat penting untuk mengetahui lingkup penelitian yang dilakukan dan memberikan batasan-batasan kepada

kesimpulan yang dapat dihasilkan. Analisa dari property/sifat jaringan/graf ini telah diteliti oleh penulis dalam [3].

Beberapa properti/sifat jaringan yang dapat dihitung dengan menggunakan *NetworkX* yaitu:

- **Jumlah simpul** dalam suatu jaringan/graf.
- **Jumlah busur** dalam suatu jaringan/graf.
- **Diameter** dari suatu jaringan/graf: jarak terjauh diantara pasangan simpul yang mungkin.
- **Kepadatan (*density*)**: rasio antara jumlah busur dan jumlah busur yang mungkin.
- **Panjang jalur rata-rata (*average path length*)**: rata-rata panjang jalur yang terpendek diantara semua pasangan simpul yang mungkin didalam graf. Efiesiensi perjalanan diantara simpul dapat dihitung menggunakan sifat ini.
- ***Girth***: ukuran dari *cycle* terkecil didalam jaringan/graf.
- **Rata-rata**: rata-rata derajat dari simpul-simpul didalam jaringan/graf.
- **Standar Deviasi**: standar deviasi dari derajat simpul dapat memberikan kita informasi sebaik apakah sutau jaringan/graf terhubungkan.
- **Koefisien kluster**: koefisien ini mengukur derajat bagaimana suatu simpul terhubung dengan simpul yang lain dan membentuk kluster.
- **Sentralitas vektor eigen.** Ini adalah ukuran pentingnya suatu simpul dalam suatu jaringan/graf. Derajat ini memberi nilai relatif pada suatu simpul berdasarkan prinsip bahwa koneksi ke simpul-simpul yang memiliki skor tinggi lebih berkontribusi pada skor simpul yang ingin kita ukur dibandingkan koneksi ke simpul yang memiliki skor kecil.
- **Keantaraan (*Betweeness*)**. Keantaraan identik dengan kekuatan atau pengaruh suatu simpul dalam suatu jaringan/graf. Simpul yang berada diantara simpul lainnya mempunyai pengaruh yang sangat besar dalam menyampaikan informasi. Adapun macam keantaraan yaitu: Sentralitas Derajat (*Degree Centrality*), Sentralitas Kedekatan (*Closeness Centrality*), dan Sentralitas Perantara (*Betweenness Centrality*).

Adapun algoritma yang dapat diterapkan untuk menghitung properti-properti diatas dengan menggunakan *NetworkX* yaitu sebagai berikut [3]:

```
import igraph
import os
import scipy

def NodeDegree(lap):
  nodeDegreeList = scipy.zeros(len(lap))
  for index in range(len(lap)):
    nodeDegreeList[index] = lap[index][index]
  mean = scipy.mean(nodeDegreeList)
  STD = scipy.std(nodeDegreeList)
  return mean, STD

def adjEigen(eigen):
  myLen = len(eigen[0])
  myMat = scipy.zeros((myLen,myLen))
```

```
    for index in range(myLen):
        for index2 in range (myLen):
            myMat[index][index2] = eigen[index][index2]
    spectrum = scipy.linalg.eigvals(myMat)
    spectrum2 = scipy.zeros(len(spectrum))
    for index in range(len(spectrum)):
        spectrum2[index] = scipy.absolute(spectrum[index])
    spectMean = scipy.mean(spectrum2)
    spectSTD = scipy.std(spectrum2)
    return spectMean, spectSTD

def Betweenness(between):
    mean = scipy.mean(between)
    STD = scipy.std(between)
    return mean, STD

def eigenvectorCentrality(eigen):
    mean = scipy.mean(eigen)
    STD = scipy.std(eigen)
    return mean, STD

    #summary(myGraph)
    f = open('instances.txt', "a")
    filename = fileList[files]
    numOfVertices = myGraph.vcount()
    numOfEdges = myGraph.ecount()
    isDirected = myGraph.is_directed()
    diameter = myGraph.diameter()
    density = myGraph.density()
    averagePathLength = myGraph.average_path_length()
    girth = myGraph.girth()
    lap = myGraph.laplacian(normalized=False)
    mean, STD = NodeDegree(lap)
    betweenMean, betweenSTD = Betweenness(myGraph.betweenness())
    algConnect = lapEigen(myGraph.laplacian(normalized=False))
    eigenMean, eigenSTD = eigenvectorCentrality(myGraph.eigenvector_centrality())
    spectMean, spectSTD = adjEigen(myGraph.get_adjacency(type=2, attribute=None, default=None))
    clusteringCoeff = myGraph.transitivity_undirected()
    f.write(filename + "\t" + repr(numOfVertices) + "\t" + repr(numOfEdges) + "\t" + repr(isDirected) + "\t"
+
    repr(diameter) + "\t" + repr(density) + "\t" + repr(averagePathLength) + "\t" + repr(girth) + "\t" +
    str(mean) + "\t" + str(STD) + "\t" + str(clusteringCoeff) + "\t" + str(betweenMean) + "\t" +
    str(betweenSTD) + "\t" + str(eigenMean) + "\t" + str(eigenSTD) + "\t" + "\t" + "\n")
    print files
    f.close()
```

Hasil yang diperoleh dari perhitungan diatas diharapkan dapat membantu menganalisa suatu jaringan didalam penelitian optimisasi lebih lanjut.

KESIMPULAN

NetworkX merupakan suatu paket perangkat lunak berbahasa Python yang dapat digunakan untuk membentuk, memanipulasi dan mempelajari struktur, dinamika serta fungsi dari jaringan kecil hingga jaringan yang rumit/besar. Berbagai bentuk dari jaringan

(atau graf) dapat dieksplorasi dan dianalisa dengan *NetworkX*, menggunakan suatu algoritma yang dapat diimplementasikan untuk menghitung properti atau sifat penting dari jaringan/graf tersebut. Sebagai perangkat lunak yang bersifat fleksibel dan *open source*, diharapkan *NetworkX* dapat bermanfaat untuk pengembangan ilmu matematika khususnya di bidang graf dan optimisasi.

DAFTAR PUSTAKA

[1] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. *Exploring Network Structure, Dynamics, and Function using NetworkX.* Proceedings of the 7th Python in Science Conference (Scipy 08).

[2] J. Ellson, E.R. Gansner, E. Koutsofios, S.C. North, and G. Woodhull. 2004. *Graphviz and dynagraph – static and dynamic graph drawing tools*. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, halaman 127–148. Springer-Verlag.

[3] W. B. L. L. Smith-Miles, K. and N. Insani. 2012. *Predicting metaheuristic performance on a graph coloring problems using data mining*, in Hybrid Metaheuristics. E.-G. Talbi. Ed. Springer. p. in press.