Penerapan Teori Graf untuk Mendeteksi Keberadaan *Fraud Rings*

Dimas Shidqi Parikesit - 13520087
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13520087@std.stei.itb.ac.id

***Abstract*—Deteksi *fraud rings* dapat dilakukan dengan penerapan teori graf. Dengan merepresentasikan histori transaksi dalam simpul dan sisi, dapat dicari *fraud rings* beserta pihak yang paling berpengaruh di dalamnya. Untuk mencari *fraud rings* dapat menggunakan salah satu dari *Louvain Community Detection, Leiden Community Detection, atau Walktrap Community Detection*. Mencari pihak yang paling berpengaruh dapat menggunakan *PageRank*. Dengan proses deteksi ini akan membantu mengatasi terjadinya *fraud*.**

***Keywords*—Graf, Database, Louvain Method, Centrality.**

I. PENDAHULUAN

Perkembangan teknologi informasi mengakibatkan masyarakat bisa saling berinteraksi tanpa bertemu secara langsung. Perkembangan ini menimbulkan banyak perubahan pada cara hidup masyarakat. Salah satu perubahan yang ditimbulkan ada pada cara jual beli barang dan atau jasa, dimana banyak bermunculan perusahaan atau perseorangan yang menjual barang dan atau jasa melalui media *online*.

Akan tetapi, sifat transaksi online yang tidak bertemu langsung antara penjual dan pembeli menimbulkan permasalahan baru yaitu sulitnya mendeteksi dan mencegah pihak yang ingin melakukan penipuan maupun kecurangan. Terdapat berbagai cara untuk melakukan tindak kecurangan dalam melakukan transaksi online, mulai dari pemalsuan resi transfer, peretasan akun, sampai pembuatan banyak akun oleh satu orang dengan tujuan untuk mensimulasikan banyak orang.

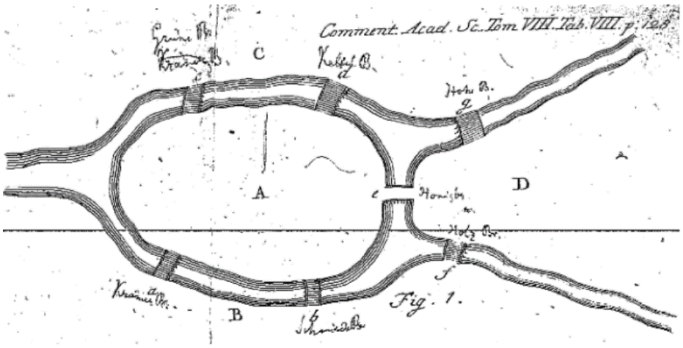
Salah satu cara mendeteksi kecurangan tersebut adalah dengan menggunakan teori graf, dimana setiap histori transaksi antara penjual dan pembeli digambarkan sebagai hubungan antara simpul dan sisi, sehingga apabila terdeteksi kecurangan pada satu transaksi, maka kecurangan lain dapat ikut terdeteksi. Dengan cara ini, pemilik *platform* jual beli dapat mengatasi terjadinya kecurangan dengan lebih akurat.

II. LANDASAN TEORI

A. Teori Graf

1. Sejarah Teori Graf

Teori Graf mulai muncul ketika Leonhard Euler mempublikasikan *paper* pada tahun 1736 yang memuat konsep *eulerian graph* dan penerapannya pada permasalahan tujuh jembatan *Königsberg* yang berada di kota Prussia, yang sekarang menjadi Kaliningrad, Russia.



Sumber: Leonard Euler's Solution to the Konigsberg Bridge Problem

Pada permasalahan tujuh jembatan *Königsberg* ini, ingin diketahui apakah terdapat lintasan tertentu sehingga seseorang dapat melalui ketujuh jembatan yang ada tepat sekali. Euler mendapatkan kesimpulan bahwa masing-masing jembatan *Königsberg* tidak dapat dilalui tepat sekali. Euler menyimpulkan bahwa masing-masing tujuh jembatan *Königsberg* tidak dapat dilalui tepat sekali. Pada paper ini pula, Euler menemukan bahwa permasalahan ini dangkal tapi tidak cukup diselesaikan dengan geometri, aljabar, atau perhitungan biasa.

2. Definisi Teori Graf

Teori graf adalah percabangan dari matematika yang memperhatikan jaringan sebuah simpul atau *node* yang dihubungkan oleh sisi atau *edge.*

3. Jenis Graf

a. Berdasarkan keberadaan sisi ganda atau gelang

Sisi ganda adalah sisi yang menghubungkan dua simpul yang sama dengan sisi yang lain. Gelang adalah sisi yang keluar dari satu simpul kemudian masuk ke simpul yang sama. Berdasarkan keberadaan sisi ganda dan gelang, graf dapat dibagi menjadi dua, yaitu

i. Graf sederhana

Graf sederhana adalah graf yang tidak mengandung gelang ataupun sisi ganda. Graf sederhana dapat disebut juga *simple graph*.

ii. Graf tak sederhana

Graf tak-sederhana adalah graf yang mengandung sisi ganda atau gelang. Graf tak-sederhana dapat disebut juga *unsimple graph*. Graf tak-sederhana dapat dibedakan menjadi dua, yaitu

a. Graf ganda

Graf ganda adalah graf yang mengandung sisi ganda. Dapat disebut juga sebagai *multi-graph*.

b. Graf semu

Graf semu adalah graf yang mengandung gelang. Dapat disebut juga sebagai *pseudo-graph*.

b. Berdasarkan orientasi arah pada sisi

Orientasi arah sebuah sisi menandakan simpul tempat keluar dan masuk dari sebuah sisi. Berdasarkan keberadaan orientasi arah ini, graf dapat dibagi menjadi dua, yaitu

i. Graf tak-berarah

Graf tak-berarah adalah graf yang sisinya tidak mempunya orientasi arah. Graf ini dapat disebut juga sebagai *undirected graph*.

ii. Graf berarah

Graf berarah adalah graf yang sisinya memiliki orientasi arah. Orientasi arah tersebut dapat berupa memasuki simpul atau keluar dari simpul. Graf ini dapat disebut juga sebagai *directed graph* atau *digraph*.

c. Berdasarkan keterhubungan

Keterhubungan pada sebuah graf menandakan apakah sebuah simpul pada graf dapat dicapai dari simpul lain. Cara mencapai simpul tersebut adalah dengan melalui sisi-sisi yang ada pada graf. Berdasarkan keterhubungan ini, sebuah graf dapat dibagi menjadi dua, yaitu

i. Graf tak-terhubung

Graf tak-terhubung adalah graf yang memiliki pasangan simpul yang tidak memiliki lintasan diantara keduanya.

ii. Graf terhubung

Graf terhubung adalah graf yang semua pasangan simpul di dalamnya memiliki lintasan diantara keduanya. Pada graf berarah, keterhubungan dapat dibagi menjadi dua, yaitu

a. Graf terhubung kuat

Graf terhubung kuat adalah graf berarah yang pada setiap pasangan simpul terdapat lintasan menuju keduanya. Graf ini dapat disebut juga *strongly connected*.

b. Graf terhubung lemah

Graf terhubung lemah adalah graf berarah yang memiliki pasangan simpul dimana sisi diantara kedua simpul tersebut hanya menuju salah satu simpul.

d. Berdasarkan jenis sisi

Sebuah sisi pada graf menandakan hubungan antar dua simpul. Hubungan tersebut ada yang dapat diabaikan, ada juga yang tidak dapat diabaikan. Berdasarkan jenis sisi yang ada, graf dapat dibagi dua, yaitu

i. Graf tak-berbobot

Graf tak-berbobot adalah jenis graf yang semua sisi nya tidak ditentukan bobotnya sehingga setiap sisi dapat dianggap identik. Graf ini dapat disebut juga sebagai *unweighted graph.*

ii. Graf berbobot

Graf berbobot adalah jenis graf yang setiap sisinya ditentukan bobotnya. Graf ini dapat disebut juga sebagai *weighted graph*.

e. Berdasarkan susunan

Terdapat enam jenis graf yang memiiliki susunan tersebut. Keenam graf tersebut adalah

i. Graf lengkap

Graf lengkap adalah graf sederhana yang setiap simpulnya terhubung dengan semua simpul lain dalam graf tersebut. Graf lengkap dengan n buah simpul dilambangkan dengan K_n. Jumlah sisi pada graf lengkap yang memiliki n buah simpul adalah $n(n-1)/2$. Graf ini dapat disebut juga sebagai *complete graph*.

ii. Graf lingkaran

Graf lingkaran adalah graf sederhana yang setiap simpul memiliki derajat dua. Graf lingkaran dengan n buah simpul dilambangkan dengan C_n.

iii. Graf teratur

Graf teratur adalah graf yang tiap simpul memiliki derajat yang sama. Jumlah sisi pada graf teratur yang memiliki n buah simpul dengan derajat r adalah $nr/2$. Graf ini dapat disebut juga sebagai *regular graph*.

iv. Graf bipartite

Graf bipartite adalah graf yang apabila himpunan simpulnya dipisah menjadi dua, sisi nya menghubungkan semua simpul diantara kedua hasil pemisahan tersebut. Graf ini dapat disebut juga sebagai *bipartite graph.*

v. Graf planar

Graf planar adalah graf yang dapat digambarkan pada bidang datar dengan sisi-sisi yang tidak bersilangan. Graf ini dapat disebut juga sebagai *planar graph.* Graf yang tidak memenuhi ketentuan ini disebut graf tak-planar.

vi. Graf bidang

Graf bidang adalah hasil dari pengubahan graf planar menjadi bentuk yang tidak bersilangan. Graf ini dapat disebut juga sebagai *plane graph*.

f. Graf kosong

Graf kosong adalah graf yang tidak memiliki sisi, atau semua simpulnya memiliki derajat nol. Graf ini dapat disebut juga sebagai *null graph* atau *empty graph.*

B. Basis Data

1. Definisi basis data

Basis data adalah kumpulan informasi terstruktur yang disimpan pada sebuah komputer. Pada sebuah aplikasi, basis data biasanya digunakan untuk menyimpan informasi pengguna atau aksi yang dilakukan pengguna tersebut. Sebuah basis data biasanya dikendalikan oleh sistem manajemen basis data, atau yang dapat disebut juga sebagai *database management system (DBMS).*

2. Jenis basis data

Berdasarkan struktur penyimpanan informasi, basis data dapat dibagi menjadi beberapa, yaitu

a. Basis data relasional

Pada basis data relasional, informasi disimpan dalam bentuk kumpulan tabel yang memiliki kolom dan baris. Basis data ini disebut juga basis data SQL.

b. Basis data non-relasional

Pada basis data non-relasional, informasi disimpan dalam bentuk pasangan *key* dan *value*. Basis data ini disebut juga sebagai basis data NoSQL.

c. Basis data graf

Pada basis data graf, informasi disimpan sebagai kumpulan entitas, dan hubungan antar entitas.

d. Basis data berbasis objek

Pada basis data berbasis objek, informasi disimpan dalam bentuk objek. Objek yang dimaksud adalah objek yang sama seperti pada pemrograman berbasis objek.

e. Basis data multi model

Pada basis data multi model, informasi dapat disimpan dalam bentuk yang bermacam-macam, seperti tabel, objek, ataupun dokumen dalam sebuah basis data.

Berdasarkan letak penyimpanan, basis data dapat dibagi menjadi tiga jenis, yaitu

a. Basis data terpusat

Pada basis data terpusat, informasi disimpan pada sebuah server terpusat yang dimiliki oleh pembuat aplikasi. Jenis ini dapat disebut juga sebagai *centralized database*.

b. Basis data terdistribusi

Pada basis data terdistribusi, informasi tidak disimpan pada pada satu server saja, tetapi pada setiap aplikasi. Salah satu contoh penggunaan basis data terdistribusi ada pada *blockchain*.

c. Basis data awan

Pada basis data awan, informasi disimpan pada layanan server milik perusahaan lain, seperti pada AWS, *Google Cloud Platform,* atau *Microsoft Azure.*

III. PENERAPAN TEORI GRAF UNTUK MENDETEKSI KEBERADAAN *FRAUD RINGS*

A. Tahap Persiapan

Dalam melakukan deteksi keberadaan *fraud rings* dalam suatu aplikasi, harus dilakukan ekstraksi data yang ingin dianalisis. Sebagai contoh, apabila ingin dilakukan deteksi keberadaan *fraud rings* pada seluruh transaksi yang dilakukan semua pengguna, maka perlu diekstrak histori transaksi dari basis data aplikasi. Sangat disarankan melakukan ekstraksi dan tidak melakukan analisis langsung pada basis data aplikasi agar tidak mengganggu performa aplikasi. Data hasil ekstraksi tersebut memiliki bentuk sesuai dengan jenis basis data yang digunakan. Apabila aplikasi menggunakan basis data relasional, maka hasil ekstraksi akan berupa tabel. Begitu pula dengan tipe basis data lainnya.

Setelah dilakukan ekstraksi, data tersebut dikonversi menjadi basis data bertipe graf. Beberapa contoh *tool* yang dapat digunakan adalah Amazon Neptune, Neo4j, dan OrientDB. Setelah konversi selesai, data siap digunakan untuk tahapan selanjutnya.

B. Tahap Deteksi

Terdapat beberapa tahap dalam mendeteksi keberadaan *fraud rings*, yang pertama adalah deteksi individual pelaku penipuan, kemudian deteksi keberadaan kelompok pelaku penipuan tersebut, setelah itu deteksi pemimpin kelompok tersebut, dan terakhir mengaplikasikan hasil temuan untuk mencari kelompok lain yang serupa.

1. Deteksi individu pelaku penipuan

Deteksi individu pelaku penipuan memiliki indikator yang berbeda pada tiap kasus. Indikator yang paling umum adalah laporan pihak tertentu secara manual, penggunaan identitas yang sama pada transaksi dalam akun yang berbeda, transaksi menggunakan alamat IP yang sama untuk banyak akun yang berbeda, transaksi menggunakan id perangkat keras (*hardware ID)* yang sama untuk akun yang berbeda, lonjakan jumlah transaksi secara signifikan dalam waktu yang singkat, dan lain-lain. Dalam melakukan deteksi individu ini, cara yang paling aman adalah dengan berkolaborasi dengan ahli pada bidang investigasi penipuan.

2. Deteksi keberadaan kelompok pelaku penipuan

Deteksi keberadaan kelompok ini dilakukan dengan menggunakan *Community Detection Algorithm*. Algoritma ini mirip dengan *clustering* pada pembelajaran mesin, dimana keduanya merupakan teknik untuk mengelompokkan data berdasarkan atribut yang dimiliki. Akan tetapi, *Community Detection* ini memiliki perbedaan utama pada penggunaannya, dimana algoritma ini didesain khusus untuk analisis jaringan yang bergantung pada sisi atau *edge*.

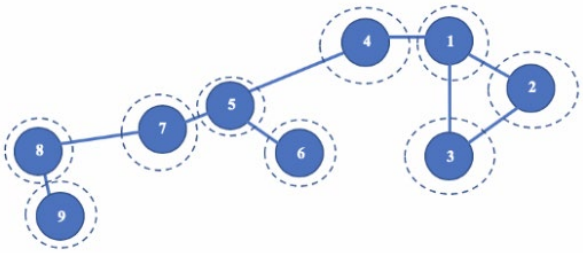
Community Detection Algorithm dapat dibagi menjadi dua kelompok besar, yaitu *Agglomerative Methods* dan *Divisive methods*. Pada *Agglomerative Methods*, sebuah pohon dibentuk secara *bottom-up*. Pada awalnya, semua simpul dianggap sebagai sebuah *cluster* berisi *singleton* atau entitas tunggal. Kemudian pada tiap iterasi, dibentuk *cluster* baru yang berisi gabungan dua *clusters* terdekat. Algoritma untuk menghitung jarak antar *cluster* berbeda-beda bergantung pada algoritma yang dipakai. *Divisive Methods* melakukan deteksi secara *top-down*, deteksi dimulai dengan keseluruhan simpul sebagai satu *cluster*, kemudian pada tiap iterasi, *cluster* ini dipecah menjadi dua *sub-cluster*, dimana *sub-cluster* ini membentuk bipartisi dari *cluster* asalnya.

Terdapat banyak algoritma yang berbasis pada kedua metode tersebut. Beberapa algoritma yang paling populer adalah *Louvain Community Detection, Leiden Community Detection,* dan *Walktrap Community Detection.*

a. *Louvain Community Detection*

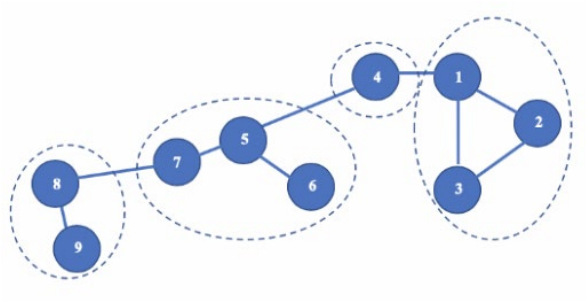
Pada algoritma *Louvain*, semua simpul dijadikan sebagai sebuah komunitas sendiri. Apabila graf memiliki *n* simpul, maka tahap pertama *Louvain* akan membuat *n* buah komunitas. Kemudian untuk setiap komunitas tersebut dihitung modularitasnya. Selanjutnya dilakukan iterasi pada tiap simpul untuk dihitung modularitas nya apabila simpul tersebut diletakkan pada komunitas lain. Simpul tersebut akan dimasukkan pada komunitas lain apabila selisih modularitas (*modularity gain)* nya positif dan terbesar. Apabila semua *modularity gain* negatif, maka simpul tidak dipindah ke komunitas manapun. Apabila semua simpul sudah tidak dapat dipindahkan ke komunitas manapun, algoritma ini akan membentuk graf baru berisi hasil pengelompokkan simpul tersebut dan kemudian mengulangi langkah-langkah sebelumnya sampai didapatkan pengelompokkan yang optimum.

Algoritma ini memiliki kelebihan dari segi kecepatan dan kemampuan untuk menemukan sub-komunitas dari sebuah komunitas. Akan tetapi, algoritma ini memiliki kelemahan pada besarnya penggunaan memori.



Tahap pertama Louvain Community Detection

Sumber: Demystifying Louvain's Algorithm and Its implementation in GPU

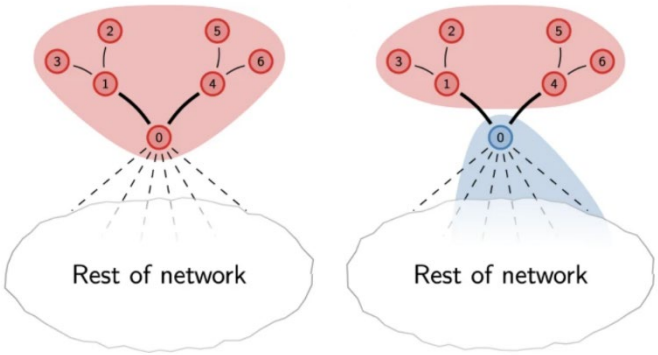


Hasil Louvain Community Detection

Sumber: Demystifying Louvain's Algorithm and Its implementation in GPU

b. *Leiden Community Detection*

Leiden Community Detection merupakan pengembangan dari *Louvain Community Detection.* Pengembangan ini dilakukan karena terdapat tiga masalah utama, yaitu *Louvain Community Detection* memiliki kecenderungan untuk menemukan komunitas yang terhubung dengan sangat lemah, memiliki kecenderungan untuk mendapatkan komunitas yang awalnya terhubung tetapi menjadi tidak terhubung pada suatu iterasi (*badly connected communities)*, dan mengelompokkan komunitas yang kecil ke komunitas yang lebih besar. Kecenderungan kedua dapat terjadi apabila terdapat dua sub-komunitas yang terhubung oleh satu simpul. Apabila simpul tersebut dipindahkan ke komunitas lain, dua sub-komunitas tersebut menjadi tidak terhubung tetapi tetapi dianggap sebagai satu komunitas.



Masalah pada Louvain Community Detection Sebelum dan Setelah Pemindahan

Sumber: From Louvain to Leiden: guaranteeing well-connected communities

Sementara itu, kecenderungan ketiga terjadi akibat batas resolusi dari *modularity*. Ketika terdapat komunitas yang terlalu kecil, komunitas tersebut tidak akan terdeteksi sebagai komunitas terpisah, tetapi akan menjadi sub-komunitas dari komunitas yang lebih besar.

Leiden Community Detection terdiri dari tiga tahap, dengan dua tahap sama dengan *Louvain Community Detection*, dan satu tahap tambahan yaitu penyempurnaan partisi. Pada tahap penyempurnaan ini, komunitas yang dibentuk setelah terjadi pemindahan dapat berpisah menjadi banyak komunitas baru. Pemisahan ini dilakukan secara random tetapi harus meningkatkan fungsi kualitas yang telah ditentukan terlebih dahulu. Pemisahan secara random ini mengakibatkan komunitas yang terbentuk lebih luas kemungkinannya.

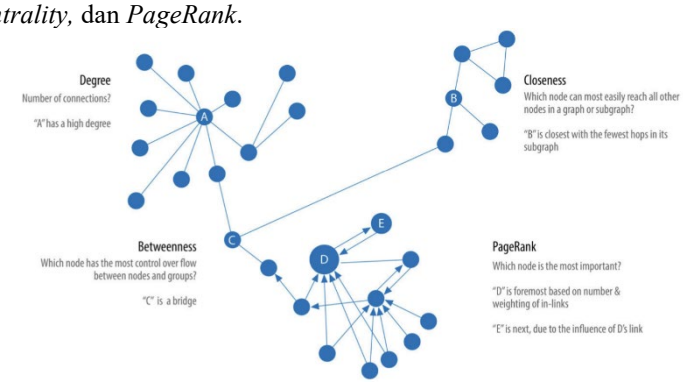
c. *Walktrap Community Detection*

Walktrap Community Detection merupakan algoritma yang menghitung jarak antar simpul menggunakan *random walks*. Pada *random walks,* program melakukan traversal secara acak dimulai dari suatu simpul. Meskipun dilakukan secara acak, traversal ini memiliki kecenderungan untuk tetap pada satu komunitas karena beban sisi antar simpul pada suatu komunitas tergolong besar. Akan tetapi, karena traversal yang dilakukan bersifat acak, perlu diperhatikan simpul mulai dari traversal tersebut agar semua komunitas di dalam graf dapat terdeteksi.

3. Deteksi pemimpin kelompok pelaku penipuan

Deteksi pemimpin pada kelompok pelaku penipuan dapat dilakukan dengan menggunakan *Centrality Algorithm*. Pada dasarnya, algoritma ini digunakan untuk memahami peran sebuah simpul pada suatu jaringan. Dengan menggunakan algoritma ini, dapat diketahui simpul yang berperan besar dalam sebuah komunitas pada graf sehingga pada kelompok pelaku penipuan dapat ditemukan pemimpinnya.

Terdapat beberapa algoritma pada *Centrality Algorithm* seperti *Degree Centrality, Closeness Centrality, Betweenness Centrality,* dan *PageRank*.



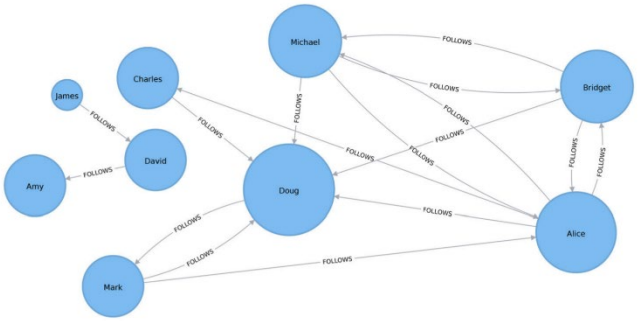
Ilustrasi Perbedaan Algoritma Centrality

Sumber: Graph Algorithms: Practical Examples in Apache Spark & Neo4j

Masing-masing algoritma tersebut memiliki fungsi masing-masing, yaitu

a. *Degree Centrality*

Degree Centrality merupakan algoritma *centrality* paling sederhana, dimana algoritma ini menghitung jumlah sisi yang masuk dan keluar sebuah simpul dan kemudian membandingkannya dengan derajat rata-rata dari suatu komunitas di dalam graf. *Centrality* ini biasanya digunakan untuk mencari pengaruh suatu simpul berdasarkan popularitasnya.

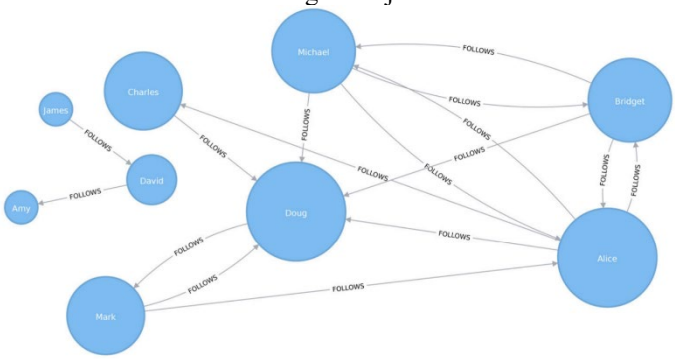


Ilustrasi Degree Centrality, semakin besar simpul semakin tinggi centrality

Sumber: Graph Algorithms: Practical Examples in Apache Spark & Neo4j

b. Closeness Centrality

Closeness Centrality adalah algoritma yang digunakan untuk mengetahui apakah suatu simpul dapat menyebarkan informasi dengan efisien di dalam suatu komunitas. Algoritma ini menghitung satu kemudian dibagi total jarak minimum ke semua simpul dalam suatu komunitas. Formula ini dapat dinormalisasi dengan mengurangi banyak simpul dengan satu terlebih dahulu sebelum dibagi total jarak minimum.

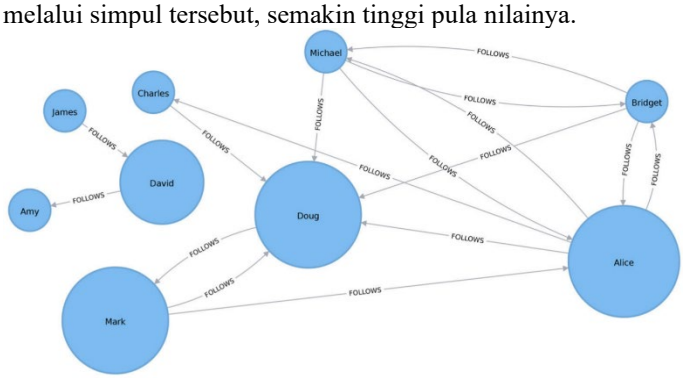


Ilustrasi Closeness Centrality, semakin besar simpul semakin tinggi centrality

Sumber: Graph Algorithms: Practical Examples in Apache Spark & Neo4j

c. Betweenness Centrality

Betweenness Centrality menggunakan pendekatan yang berbeda untuk menghitung simpul yang paling berpengaruh pada suatu komunitas karena algoritma ini mencari simpul yang berperan sebagai penghubung satu bagian komunitas ke bagian lain. Algoritma ini menghitung *shortest weighted path* antara setiap pasang simpul di komunitas. Setiap simpul kemudian mendapatkan nilai berdasarkan jumlah *shortest path* yang melalui simpul tersebut. Semakin banyak *shortest path* yang melalui simpul tersebut, semakin tinggi pula nilainya.



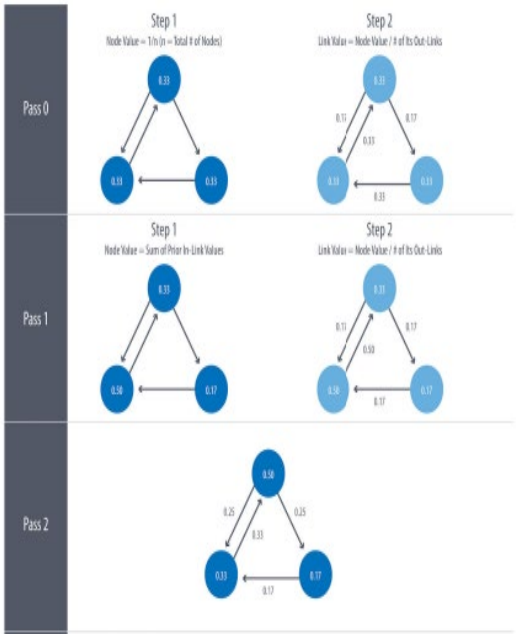
Ilustrasi Between Centrality, semakin besar simpul semakin tinggi centrality nya

Sumber: Graph Algorithms: Practical Examples in Apache Spark & Neo4j

d. PageRank

PageRank adalah algoritma *centrality* yang paling populer. Kepopuleran ini disebabkan oleh sifat algoritma yang mengukur pengaruh simpul lain juga dalam menentukan pengaruh suatu simpul. Sifat ini lah yang menyebabkan *PageRank* paling tepat digunakan sebagai algoritma untuk deteksi pemimpin pada kelompok pelaku penipuan.

Algoritma *PageRank* terdiri dari dua tahap pada setiap iterasi. Pertama, menentukan nilai semua simpul pada komunitas. Kedua, menentukan nilai semua sisi pada komunitas. Dalam menentukan nilai sebuah simpul, *PageRank* menjumlahkan beban semua sisi yang masuk ke ke simpul tersebut. Dalam menentukan nilai sebuah sisi, *PageRank* menghitung nilai simpul asal dibagi dengan jumlah sisi yang keluar dari simpul asal tersebut. Karena tiap iterasi menggunakan nilai hasil iterasi sebelumnya, *PageRank* baru bisa dianggap selesai ketika nilai simpul tiap iterasi menjadi tetap (*converges to a value)* atau ketika iterasi telah dilakukan sebanyak yang diinginkan. Berikut ilustrasi algoritma *PageRank*



Ilustrasi Algoritma PageRank

Sumber: Graph Algorithms: Practical Examples in Apache Spark & Neo4j

C. Tahap Menyimpulkan

Setelah melakukan tahapan deteksi pada bagian sebelumnya akan didapatkan *fraud rings* beserta orang yang paling berpengaruh pada *fraud rings* tersebut. Selanjutnya perlu dilakukan verifikasi terhadap seluruh anggota *fraud rings* yang terdeteksi.

Untuk mempermudah proses deteksi selanjutnya, metode-metode tersebut dapat disatukan menjadi sebuah *pipeline* untuk dijalankan secara otomatis setiap durasi waktu tertentu. Apabila ingin dilakukan pencegahan terjadinya *fraud*, dapat dijalankan *Similarity Algorithm* seperti *Jaccard Similarity* pada setiap transaksi yang dilakukan pengguna.

IV. KESIMPULAN

Teori graf dapat diterapkan untuk menyelesaikan berbagai macam permasalahan. Salah satunya adalah deteksi *fraud rings* dari suatu aplikasi. Tahapan deteksi adalah melakukan konversi data menjadi bentuk graf, kemudian deteksi individu pelaku penipuan, deteksi *fraud rings,* kemudian deteksi orang yang berpengaruh pada *fraud rings* tersebut. Dengan melakukan

tahapan tersebut dapat dideteksi *fraud rings* beserta orang yang berpengaruh di dalam *fraud rings* tersebut.

V. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu pengerjaan makalah ini sehingga dapat selesai dengan maksimal. Puji syukur kepada Tuhan Yang Maha Esa karena telah melancarkan pengerjaan makalah ini. Terima kasih juga kepada dosen mata kuliah IF 2120 Matematika Diskrit, Dra. Harlli S., M.Sc., Dr. Ir. Rinaldi Munir, M.T., dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc, atas bimbingan selama ini dalam mengajar sehingga penulis dapat membuat makalah ini. Penulis juga berterima kasih kepada keluarga serta rekan-rekan yang telah memberikan semangat kepada penulis.

REFERENCES

[1] Munir, Rinaldi. 2021. Graf (Bagian I). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf, diakses 13 Desember 2021 pukul 13.21 WIB

[2] Munir, Rinaldi. 2021. Graf (Bagian II). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian2.pdf, diakses 13 Desember 2021 pukul 13.21 WIB

[3] Blondel, Vincent D. dkk. (2008). Fast Unfolding of Communities in Large Networks. Belgium, 1-12.

[4] Traag, V.A. dkk. (2019). From Louvain to Leiden: guaranteeing well-connected communities, 1-6.

[5] Maurice Roux. A Comparative Study of Divisive and Agglomerative Hierarchical Clustering Algorithms. Journal of Classification, Springer Verlag, 2018, 35 (2), pp.345-366. ff10.1007/s00357-018- 9259-9ff. ffhal-02085844f

[6] Needhan, Mark. Hodler dan Amy E. 2019. Graph Algorithms: Practical Examples in Apache Spark & Neo4j. *California*, O'Reilly.

[7] Hodler, Amy E. 2021. Financial Fraud Detection with Graph Data Science: How Graph Algorithms & Visualization Better Predict Emerging Fraud Patterns. https://go.neo4j.com/rs/710-RRC-335/images/Neo4j-Financial-Fraud-Detection-GDS-white-paper-EN-A4.pdf, diakses 13 Desember 2021 pukul 15.30 WIB

[8] Hodler, Amy E. 2020. Financial Fraud Detection with Graph Data Science: Identifying Fraud Rings. https://neo4j.com/blog/financial-fraud-detection-graph-data-science-identifying-fraud-rings/, diakses 13 Desember 2021 pukul 15.00 WIB

[9] Hodler, Amy E. 2020. Financial Fraud Detection with Graph Data Science: Identifying First-Party Fraud. https://neo4j.com/blog/financial-fraud-detection-graph-data-science-identifying-first-party-fraud/, diakses 13 Desember 2021 pukul 15.00 WIB

[10] Jayawickrama. 2021. Community Detection Algorithms. https://towardsdatascience.com/community-detection-algorithms-9bd8951e7dae, diakses 12 Desember 2021 pukul 14.00 WIB.

[11] Mishra, Abhishek. 2019. Demystifying Louvain's Algorithm and Its implementation in GPU. https://medium.com/walmartglobaltech/demystifying-louvains-algorithm-and-its-implementation-in-gpu-9a07cdd3b010, diakses 12 Desember 2021 pukul 15.00 WIB.

[12] Paoletti, Teo. Leonard Euler's Solution to the Konigsberg Bridge Problem. https://www.maa.org/press/periodicals/convergence/leonard-eulers-solution-to-the-konigsberg-bridge-problem, diakses 12 Desember 2021 pukul 13.00 WIB.

[13] Oracle. What Is a Database?. https://www.oracle.com/database/what-is-database/, diakses 12 Desember pukul 13.30 WIB.

[14] Carlson, Stephan C.. "graph theory". Encyclopedia Britannica, 24 Nov. 2020, https://www.britannica.com/topic/graph-theory. diakses 14 Desember 2021.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Desember 2021

Dimas Shidqi Parikesit - 13520087