**Implementasi VoG (*Vocabulary* based summarization of *Graph*) Pada *Web* *Graph***

***Implementation of VoG (Vocabulary based summarization of Graph) on the Web Graph***

**Satrio Adityo Hartomo1, Kemas Rahmat Saleh W., S.T., M. Eng.2, Siti Sa’adah, S.T., M.T.3**

1Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom.

1[satrioadityo](mailto:satrioadityo)@gmail.com, 2[bagindok3m45@gmail.com](mailto:bagindok3m45@gmail.com), 3[tisataz@gmail.com](mailto:tisataz@gmail.com).

**Abstrak**

*Web* *graph* yang memiliki data semakin banyak membuat *graph* semakin sulit untuk diidentifikasi struktur dan informasinya. Untuk mengidentifikasi struktur dan informasi diperlukan suatu metode untuk meringkas *graph* yang besar menjadi *subgraph*-*subgraph* yang lebih kecil. Metode VoG dapat digunakan untuk meringkas *graph* yang besar dan mengidentifikasi struktur *subgraph*. Dalam proses pengidentifikasian *subgraph*, jumlah *node* maksimum dalam GCC berpengaruh terhadap struktur *subgraph* yang dihasilkan. Semakin banyak jumlah *node* maksimum dalam GCC semakin sedikit menghasilkan *subgraph* dengan struktur *clique* dan *star*, namum semakin banyak mengasilkan *subgraph* dengan struktur *chain*. Jumlah *node* maksimum dalam GCC juga berpengaruh terhadap waktu eksekusi VoG. Semakin banyak jumlah *node* dalam GCC semakin cepat pula waktu eksekusi VoG. Dari *subgraph*-*subgraph* yang dihasilkan dan struktur *subgraph* yang diidentifikasi, informasi yang dapat diperoleh adalah *subgraph* *clique* merepresentasikan adanya jaringan user yang saling berpendapat dalam suatu link pertanyaan, *subgraph* *star* merepresentasikan adanya satu user yang berkontribusi menjawab pertanyaan di beberapa link pertanyaan dalam *web* quora.com.

**Kata Kunci** : *web* *graph*, GCC, VoG, *subgraph*, struktur, informasi

***Abstract***

*Web graph that have increasing data make the graph more difficult for identified the structure and the information. For identify the structure and information need some method for summarize large graph into smaller subgraph. VoG method can used for summarize the large graph and identify the subgraph structure. In subgraph identification process, number of maximum node in GCC affect the generated subgraph. More number of maximum node in GCC generate less of subgraph with clique and star structure, but generate more subgraph with chain structure. The number of maximum node in GCC also affect on execution time of VoG. More number of maximum node in GCC faster execution time of VoG. From generated and identified subgraph structure, information that can obtainer is clique subgraph represent there is a user network that poll on a question link, star subgraph represent there is one user contribute an answer on some question link on quora.com.*

***Keywords****: web* *graph*, GCC, VoG, *subgraph*, *structure*, *information*

# Pendahuluan

Dengan adanya teknik penyimpanan data yang baru yaitu *graph database* [5], cara untuk menyimpan data tidak lagi terbatas pada *relational database* yang struktural. Data yang mempunyai perbedaan atribut (semi terstruktur) dapat disimpan ke *database* dengan representasi *graph* [5], seperti Facebook dan Twitter yang juga menggunakan *graph database* untuk menyimpan data-datanya [8,10]. *Website* yang lain pun memiliki data yang unik, berjumlah banyak dan semakin bertambah hingga saat ini [2]. *Website* yang dahulu hanya menampilkan data statis kini telah berevolusi menjadi *website* yang dapat menampilkan data secara dinamis dan dapat direpresentasikan sebagai *graph* (*web graph*) [5].

Pada referensi [2] disebutkan bahwa pada Juli 2000 terdapat dua milyar *web*, dan terus meningkat jumlahnya hingga empat milyar *web* pada tahun 2001. *Web graph* yang besar dengan ribuan *node* dan *edge* [9], akan sulit untuk diidentifikasi bagaimana bentuk strukturnya, apakah memiliki struktur *random* atau tidak. Dengan ukuran yang besar tersebut akan sulit juga untuk memperoleh informasi. Oleh karena itu diperlukan metode untuk meringkas *web graph* agar ukurannya menjadi lebih kecil dari *web graph* aslinya, sehingga dapat dilakukan pengidentifikasian struktur dan analisis untuk mendapatkan informasi pada hasil ringkasan *web graph*.

VoG (*Vocabulary based summarization of Graph*) adalah metode untuk meringkas *graph* yang besar secara efisien [3]. Dalam metode VoG akan dilakukan *graph decomposition* menggunakan algoritma Slashburn. Algoritma Slashburn akan menghasilkan sekumpulan *subgraph* yang mana *subgraph* yang dihasilkan akan dilakukan proses identifikasi struktur. Dalam proses menghasilkan *subgraph*, jumlah *node* maksimum pada dalam GCC (*Giant Connected Component*) akan mempengaruhi jumlah dan struktur *subgraph*. Waktu eksekusi algoritma VoG juga akan dipengaruhi oleh hal tersebut. Dalam tugas akhir ini akan dilakukan implementasi *web graph* *summarization* menggunakan metode VoG untuk meringkas *web graph* menjadi *subgraph-subgraph* yang lebih kecil ukuran atau jumlah *node*nya, menganalisis pengaruh jumlah *node* maksimum dalam GCC terhadap *subgraph* yang dihasilkan, menganalisis pengaruh jumlah *node* maksimum dalam GCC terhadap waktu eksekusi VoG dan mendapatkan informasi dari *subgraph-subgraph* yang dihasilkan.

# Dasar Teori dan Perancangan

## Teori Graph

*Graph*, secara definisi adalah sekumpulan *nodes* () dan *edges* (). Sekumpulan *nodes* bisa saja tidak terbatas atau bisa disebut ***infinite graph***, sedangkan yang sekumpulan *nodes*nya terbatas disebut ***finite graph***. Dalam tugas akhir ini hanya akan fokus pada *finite graph*. *Graph* dapat dinotasikan sebagai . adalah sekumpulan *nodes* dan adalah sekumpulan *edges* yang menghubungkan *nodes* [6]. Perbedaan jenis *graph* dapat dilihat berdasarkan ada atau tidaknya arah *edge*, *graph* jenis ini terdiri dari ***directed graph*** (*graph* yang mempunyai *edge* berarah) dan ***undirected graph* (***graph* yang mempunyai *edge* tidak berarah). Berdasarkan *edge* yang meng*hub*ungkan *nodes* terdiri dari ***simple graph*** (tidak memiliki *loop* dan *multiple edge*), ***multigraph*** (tidak memiliki *loop edge* tetapi memiliki *multiple edge*) dan ***pseudograph*** (memiliki *loop edge*). Berdasarkan ada atau tidaknya label terdiri dari ***graph* berlabel** dan ***graph* tak berlabel**. Sekumpulan *nodes* dan *edges* yang ada di *graph* dapat digunakan untuk memodelkan permasalahan yang ada pada dunia nyata, sebagai contoh *graph* jaringan komputer. Namun, dalam kondisi tertentu untuk menyelesaikan permasalahan yang ada, tidak seluruh *nodes* digunakan. Sehingga yang digunakan untuk menyelesaikan masalah adalah bagian kecil dari *graph*, atau bisa disebut dengan *subgraph*. Secara definisi, *subgraph* dari *graph* adalah *graph* dimana dan dan [6]. Cara merepresentasikan *graph* ada beberapa cara, diantaranya yaitu menggunakan *Adjacency Matrix* dan *Adjacency List* [6].

## Graph Database

*Graph Database Management System* adalah *database management system* yang memiliki metode CRUD (*Create*, *Read*, *Update*, dan *Delete*) untuk memaparkan model data *graph*. Properti model *graph* dapat dideskripsikan sebagai berikut :

* Terdiri dari *nodes* dan *relationship/edges*.
* Setiap *nodes* terdiri dari pasangan *key-value*.
* *Edge* mempunyai nama dan arah, jika tidak memiliki arah disebut *undirected graph.*
* *Edge* dapat juga mempunyai properti.

Dengan abstraksi sederhana dari *nodes* dan *edges* kedalam struktur yang saling terhubung, *graph database* memungkinkan untuk dibangunnya model yang mirip dengan permasalahan yang ada di dunia nyata [5].

## Web Graph

*Webpages* dan *hyperlinks* di *World Wide Web* dapat direpresentasikan sebagai *nodes* dan *edge* di *directed graph*. Saat ini, *graph* tersebut memiliki milyaran *nodes* dan *edge* dan bertambah banyak seiring berjalannya waktu. Ada beberapa alasan untuk mempelajari *web graph*, diantaranya adalah bagaimana algoritma yang bekerja pada *web graph*, bagaimana *web search*nya, dan bagaimana pengklasifikasiannya [12].

Tidak hanya itu, ada alasan lain untuk mengembangkan pemahaman dari *web graph* ini, diantaranya dapat :

1. Mendesain strategi *crawling* pada *web*.
2. Memahami sosiologi pembuatan konten pada *web*.
3. Menganalisis *behavior* dari algoritma *web*.
4. Memprediksi evolusi struktur *web*.
5. Memprediksi munculnya fenomena baru di *web* [1].

## Graph Compression

Jika dalam suatu *graph* dapat ditemukan *good community*, maka *graph* tersebut dapat dikompres yang mana dapat membantu untuk menyusutkan ukuran data [13]. Rincian proses *graph compression* dalam tugas akhir ini dijelaskan pada bagian 2.4.1 dan 2.4.2.

### **2.4.1 *Graph* *Shattering***

Untuk menyelesaikan permasalahan dalam *graph compression*, diperlukan proses untuk mengurai *graph* dengan cara mengeksploitasi *hubs* (*node* dengan *degree* paling tinggi). Dengan hasil eksploitasi *hubs* didapatkan *alternative community* yang berbeda dari *clique community* [13].

Dari referensi [13] disebutkan bahwa berdasarkan observasi, *real world graph* dapat dengan mudah dipecah dengan menghapus *hub node* dari *graph*. Dengan penghapusan *hub node*, *graph* terurai menjadi banyak *subgraph-subgraph*.

### **2.4.2 Slash and Burn**



Gambar 2‑1: Algoritma Slashburn

Baris 1 dan 2 dari algoritma 1 menghapus *nodes* dengan *degree* paling tinggi dan *edge* yang berasosiasi, dengan demikian *graph* terurai dalam tiga kelompok:

* -*hub*set: *nodes* dengan *degree* paling tinggi.
* GCC: *node* yang dimiliki oleh *Giant Conncected Component* dari .
* Spokes: *node* yang dimiliki oleh *non-giant connected component* dari [13].

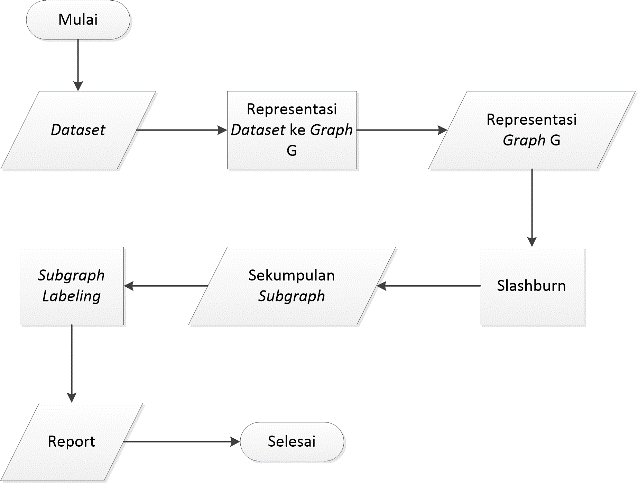
## VoG

*Vocabulary based summarization of Graph* (VoG) merupakan metode yang efektif dan efisien untuk meringkas *graph* yang besar. *Vocabulary* (Ω) adalah sekumpulan struktur *graph* yang terdiri dari struktur *full-clique, near-clique, full-bipartite, near-bipartite, perfect-star, near-star, perfect-chains* dan *near-chains*. Masih ada struktur *graph* yang lain, akan tetapi dari referensi [3] disebutkan bahwa struktur *graph* yang sering muncul dalam *real world* *graph* adalah struktur yang ada pada *Vocabulary* tersebut. Oleh karena itu, dalam tugas akhir ini dibangun sistem yang dapat mengidentifikasi struktur dalam *vocabulary* [3].

*Graph* yang ada di dunia sangatlah banyak dan besar, orang-orang sulit untuk memahami dengan mudah *graph* yang besar dan mungkin *random* strukturnya. Dengan adanya VoG, *graph* dapat diringkas dan orang-orang dapat lebih mudah memahaminya [3].

## Perancangan Sistem

Sistem yang dibangun dalam tugas akhir ini adalah sistem yang dapat meringkas *graph* yang besar menjadi sekumpulan *subgraph* yang berukuran lebih kecil. Dari masing-masing *subgraph* akan diidentifikasi struktur dan dianalisis informasinya. *Dataset* yang digunakan dalam pengujian sistem adalah *dataset* yang didapat melalui *crawling* *web* quora.com. Data yang di-*crawl* adalah data *username* dalam suatu *link question*. Dari data yang sudah diperoleh kemudian direpresentasikan sebagai *graph*, yang mana *node* merepresentasikan *username*, dan *edge* m erepresentasikan dua *username* menjawab pertanyaan di *link* yang sama. Berikut ini adalah *flow chart* dari sistem yang dibangun:



Gambar 2‑2: Flowchart sistem secara umum

Algoritma Slashburn akan berhenti dieksekusi jika . Ketika proses slashburn berjalan, *graph* diproses secara iteratif dengan batas . *Graph* akan dicari *hub node* di dalam GCC untuk dihapus. Penghapusan *hub node* ini disertai dengan *edge*-nya dan menghasilkan *subgraph*. Jika proses penghapusan *node* ini menghasilkan satelit *node* yang *isolated*, maka *hub node* akan dibuat kembali dan satelit *node* akan di*hub*ungkan dengan *hub node*. Hal ini dilakukan agar tidak terbentuk *subgraph* dengan satu *node*. Setelah proses *graph compression* selesai, akan dihasilkan *subgraph-subgraph*.

Dari *subgraph-subgraph* tersebut akan diidentifikasi strukturnya. Untuk proses identifikasi struktur, yang pertama dilakukan adalah mendapatkan jumlah *node* dalam *subgraph* dan *adjacency matrix*-nya. Setelah jumlah *node* didapat, maka dibuatlah empat model yaitu *clique, star, chain,* dan *bipartite* model. Model-model inilah yang akan digunakan untuk proses mendapatkan *error matrix*. Berikut ini adalah contoh proses untuk mendapatkan *error matrix*:

(3.1)

Contoh operasi matematika (3.1) diatas adalah operasi *adjacency matrix subgraph* XOR dengan *matrix* model *clique*, yang menghasilkan *error matrix* untuk model *clique*.

(3.2)

Contoh operasi matematika (3.2) diatas adalah operasi *adjacency matrix subgraph* XOR dengan *matrix* model *star*, yang menghasilkan *error matrix* untuk model *star*.

Dari dua contoh *error matrix* diatas, *error matrix* kemudian di-*encode* menjadi *error score* untuk masing-masing model. Dalam kasus di atas, *error score* untuk *error matrix* model *clique* = 2. Sedangkan *error score* untuk *error matrix star* adalah 0. Setelah didapatkan *error score* untuk masing-masing model, akan dipilih *error score* yang paling kecil nilainya, sehingga *subgraph* akan diidentifikasi sebagai ***Subgraph* dengan Struktur *Star***.

# Pembahasan

### **3.1 Analisis hasil pengujian algoritma *graph* *decomposition***

Pengujian *graph decomposition* menggunakan *sample dataset* yang dibuat sendiri. Terdapat dua *dataset* yang digunakan. Berikut adalah tampilan *graph* dari *dataset* pertama yang digunakan untuk menguji algoritma *graph decomposition*:



Gambar ‑: Dataset pertama untuk pengujian Slashburn

Setelah *graph* pada gambar 3-1 dilakukan proses *graph decomposition* menggunakan algoritma Slashburn, didapatlah keluaran sekumpulan *subgraph*.

Berikut adalah gambar keluaran algoritma Slashburn:



Gambar ‑: Keluaran Slashburn dari dataset pertama

Setelah didapatkan sekumpulan *subgraph* hasil algoritma Slashburn, akan diidentifikasi masing-masing struktur *subgraph*nya. Untuk proses pengidentifikasian struktur, dilakukan proses *subgraph labeling* seperti yang tertera pada Perancangan Sistem. Berikut ini adalah gambar keluaran proses *subgraph labeling*:



Gambar ‑: Keluaran proses subgraph labeling untuk datasaet pertama

Berikut adalah gambar *dataset* kedua yang digunakan:



Gambar ‑: Dataset kedua untuk pengujian Slashburn

Setelah *graph* *dataset*-2 pada gambar 3-3 dilakukan proses *graph decomposition* menggunakan algoritma Slashburn, didapatlah keluaran sekumpulan *subgraph*. Berikut adalah gambar keluaran algoritma Slashburn:



Gambar ‑: Keluaran Slashburn dari dataset kedua

Setelah didapatkan sekumpulan *subgraph* hasil algoritma Slashburn di gambar 3-4, akan diidentifikasi masing-masing strukturnya. Untuk proses pengidentifikasian struktur, dilakukan proses *subgraph labeling* seperti yang tertera pada Perancangan Sistem. Berikut ini adalah gambar keluaran proses *subgraph labeling*:



Gambar ‑: Keluaran subgraph labeling untuk dataset kedua

Dari gambar 3-2, algoritma Slashburn berhasil mengurai *graph* pada gambar 3-1 menjadi sekumpulan *subgraph* yang mana masing-masing *subgraph* berhasil diidentifikasi strukturnya. Struktur yang dapat teridentifikasi adalah *perfect clique, perfect bipartite, perfect star*, dan *perfect clique*.

Dari gambar 3-4, *dataset* kedua yang digunakan berhasil diurai oleh algoritma Slashburn menghasilkan sekumpulan *subgraph* seperti pada gambar 3-5. Terdapat beberapa *subgraph* yang kurang untuk menjadi *perfect structure*. Karena struktur *subgraph* tidak sempurna, maka akan diidentifikasi sebagai *near structure*. VoG dapat mengidentifikasi struktur *near clique, near star, near chain,* dan *near bipartite* seperti tertera pada gambar 3-6.

Dari kedua *sample dataset* yang digunakan, *graph* dapat diidentifikasi strukturnya sesuai dengan *vocabulary* dalam VoG.

### **3.2 Analisis pengaruh jumlah *node* maksimum dalam GCC terhadap *subgraph* yang dihasilkan oleh VoG**

Pada pengujian VoG ini digunakan *dataset* yang didapat dari hasil *crawl web* quora.com. Hasil VoG akan ditampilkan dalam bentuk tabel. Berikut ini adalah gambar *dataset* dari quora.com yang digunakan:



Gambar ‑: Dataset hasil crawling quora.com dengan jumlah node 2250 dan edge 5029

Dari *graph* seperti pada gambar 3-7 yang sudah dimuat ke dalam sistem, akan dilakukan *graph decomposition* untuk menghasilkan *subgraph* dengan variable jumlah *node* dalam GCC yang diubah-ubah. Berikut ini adalah gambar *graph* setelah *graph decomposition* dengan jumlah *node* maksimum dalam GCC sama dengan 3:



Gambar ‑: Graph quora yang sudah dilakukan graph decomposition, terdapat 1742 node dan 1321 edge

Dari hasil *graph docomposition*, masing-masing *subgraph* akan diidentifikasi strukturnya sesuai dengan *vocabulary*. Berikut ini adalah hasil identifikasi struktur *graph* gambar 3-8 untuk pola pertama dengan jumlah node maksimum dalam GCC sama dengan 5, 7, 9, dan 11 serta pola kedua dengan jumlah node maksimum dalam GCC sama dengan 5, 10, 15, dan 20:

Gambar 3‑9: Grafik hasil pengujian pola pertama

Gambar ‑: Grafik hasil pengujian pola kedua

Dari grafik pada gambar 3-9 dan gambar 3-10 terlihat bahwa struktur *perfect clique* semakin berkurang jumlahnya seiring bertambahnya jumlah *node* maksimum dalam GCC. Struktur *near clique* dan *near star* juga semakin menurun pada jumlah *node* maksimum dalam GCC diatas 10. Penurunan ini dikarenakan bentuk GCC yang banyak tersusun dari *subgraph* *clique* dan *star* lalu ketika algoritma Slashburn beroperasi, Slashburn menghapus *hub node* dalam GCC, memotong *edge*nya dan membentuk *subgraph-subgraph near chain*. Terlihat dari grafik 3-9 dan 3-10 *subgraph near chain* bertambah jumlahnya pada jumlah *node* maksimum dalam GCC diatas 10.

### **3.3 Analisis pengaruh jumlah *node* maksimum dalam GCC terhadap waktu eksekusi VoG**

Pada pengujian VoG ini digunakan *dataset* yang didapat dari hasil *crawl web* quora.com. Dari satu *dataset* tersebut akan dilakukan pengujian dengan mengubah-ubah jumlah *node* maksimum dalam GCC lalu mencatat waktu eksekusi VoG. Berikut ini adalah tabel hasil pengujian yang diperoleh mengenai waktu rata-rata eksekusi VoG:

Tabel ‑: Hasil pengujian waktu rata-rata VoG

| **Jumlah *node* maksimum dalam GCC** | **Eksekusi Slashburn (ms)** | **Eksekusi *Subgraph* *Labeling* (ms)** | **Eksekusi VoG (ms)** |
| --- | --- | --- | --- |
| **5** | **11529.83333** | **3403.733333** | **14934.23333** |
| **10** | **5085.966667** | **3755.133333** | **8841.6** |
| **15** | **3257.6** | **3323.433333** | **6581.466667** |
| **20** | **2375.033333** | **3472.633333** | **5848** |

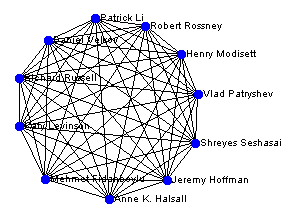
Gambar 3‑11: Grafik hasil pengujian waktu eksekusi

Dari grafik pada gambar 3-11 terlihat bahwa semakin banyak jumlah *node* maksimum dalam GCC semakin cepat juga waktu eksekusi VoG. Hal ini juga dipengaruhi oleh waktu eksekusi algoritma Slashburn. Semakin banyak jumlah *node* maksimum dalam GCC semakin cepat juga waktu eksekusi Slashburn. Hal ini dikarenakan semakin banyak jumlah *node* dalam GCC *subgraph* yang dihasilkan semakin sedikit, oleh karena itu proses untuk mengurai *graph* menjadi *subgraph* yang sedikit jumlahnya membutuhkan waktu yang lebih singkat.

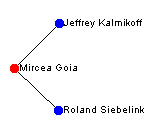
Untuk waktu eksekusi proses *subgraph labeling* dari *subgraph-subgraph* yang dihasilkan oleh Slashburn relatif stabil pada waktu 3 detik. Hal ini dikarenakan oleh jumlah banyaknya *subgraph* dan jumlah *node* yang ada pada *subgraph*. Pada jumlah *node* maksimum dalam GCC sama dengan 5, *subgraph* yang dihasilkan berjumlah lebih banyak daripada pada jumlah *node* maksimum dalam GCC sama dengan 20. Untuk memproses banyak *subgraph* dengan *node* yang sedikit waktu eksekusinya akan relatif sama dengan memproses sedikit *subgraph* dengan *node* yang banyak. Hal ini terbukti pada hasil pengujian dalam grafik di atas yang mana waktu eksekusi *subgraph* *labeling* relatif stabil pada waktu 3 detik.

### **3.4 Analisis informasi struktur *subgraph***

Dari hasil pengujian *graph* quora yang telah dilakukan sebelumnya, dapat diperoleh struktur *clique*, *star*, dan *chain*. Berikut ini adalah *sampling subgraph clique* dan *star* yang diambil dari hasil pengujian:



Gambar 3‑12: Subgraph clique hasil pengujian



Gambar 3‑13: Subgraph star hasil pengujian

Karena *graph* yang digunakan mempunyai representasi *node* sebagai *user* dan *edge*nya adalah relasi bahwa dua *user* menjawab pertanyaan yang sama, maka dengan metode *manual inspection* pada masing-masing *subgraph* yang dihasilkan, struktur *clique* dapat dianalisis informasinya sebagai *subgraph* yang banyak *user*nya melakukan jajak pendapat pada suatu pertanyaan tertentu. Sedangkan untuk struktur *star* dapat dianalisis informasinya bahwa terdapat satu *user* yang banyak berkontribusi menjawab banyak pertanyaan dalam *web* quora.

# Kesimpulan dan Saran

## 4.1 Kesimpulan

Berdasarkan hasil pengujian dapat diambil kesimpulan sebagai berikut:

1. Metode VoG dapat digunakan untuk *summarize* *graph* dan mengidentifikasi struktur *subgraph* yang dihasilkan. *Subgraph*-*subgraph* yang dihasilkan memiliki struktur *perfect* *clique*, *perfect* *star*, *perfect* *chain*, *perfect* *bipartite*, *near* *clique*, *near* *star*, *near* *chain* dan *near* *bipartite*. Struktur yang dihasilkan sesuai dengan *vocabulary* dalam VoG.
2. Jumlah *node* maksimum dalam GCC berbanding terbalik dengan jumlah struktur *perfect* *clique*, *near* *clique*, dan *near* *star* yang dihasilkan, namun berbanding lurus dengan jumlah struktur *near* *chain* yang dihasilkan. Semakin banyak *node* dalam GCC menghasilkan struktur *clique* dan *star* yang semakin sedikit namun semakin banyak menghasilkan struktur *chain*.
3. Jumlah *node* maksimum dalam GCC berbanding terbalik dengan waktu eksekusi VoG. Semakin banyak *node* dalam GCC semakin cepat waktu eksekusi VoG.
4. Struktur *subgraph* yang dihasilkan oleh VoG dapat dianalisis dengan metode *manual inspection* untuk mendapatkan informasi. Dalam tugas akhir ini, struktur *clique* menggambarkan bahwa banyak *user* di quora yang melakukan jajak pendapat dalam suatu pertanyaan. Struktur *star* menggambarkan bahwa ada satu *user* yang berkontribusi menjawab banyak pertanyaan di quora.com.

## 4.2 Saran

Berikut ini adalah saran yang untuk penelitian selanjutnya:

1. Sistem yang dikembangkan dalam tugas akhir ini tidak terbatas pada *web* *graph*. Representasi *graph* apapun dapat digunakan menjadi *dataset* dalam sistem ini. Jadi untuk selanjutnya dapat digunakan *dataset* Email *Graph* yang mungkin dapat terdapat struktur *bipartite* atau *chain* yang menggambarkan terdapat jaringan broadcast email dan email forwarding.
2. Metode Slashburn yang digunakan untuk mengurai *graph* menjadi *subgraph* masih dapat dioptimasi menggunakan algoritma Greedy untuk memilih *hub* yang terbaik untuk dihapus.
3. Metode *Subgraph* *Labeling* dapat ditingkatkan akurasi pemilihan *subgraph*nya dengan encoding *subgraph* menjadi nilai bits. Nilai hasil encode yang rendah dapat dipilih untuk menentukan struktur *subgraph*.
4. Untuk menganalisis informasi dalam *subgraph* dapat menggunakan metode lain yang lebih akurat dan objektif.

# Daftar Pustaka

|  |  |
| --- | --- |
| [1] | A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Thomkins dan J. Wiener, “Graph Structure in the Web,” *Computer Nerworks Elsevier,* vol. 33, pp. 309-320, 2000. |
| [2] | B. H. Murray dan A. Moore, “Sizing The Internet,” Cyveillance, 2000. |
| [3] | D. Koutra, U. Kang, J. Vreeken dan C. Faloutsos, “VoG : Summarizing and Understanding Large Graphs,” dalam *Proceedings of the 2014 SIAM International Conference on Data Mining*, 2014. |
| [4] | Dhia, “Finding the Patterns in a Mysterious New DGA,” [Online]. Available: https://labs.opendns.com/2013/10/24/mysterious-dga-lets-investigate-sgraph/. [Diakses 18 March 2015]. |
| [5] | I. Robinson, J. Webber dan E. Eifrem, Graph Databases, Sebastopol: O'Reilly Media, Inc, 2013. |
| [6] | K. H. Rosen, Discrete Mathematics and Its Applications, New York: The McGraw-Hill Companies, Inc., 2012. |
| [7] | M. A. Kolosovskiy, Data structure for representing a graph:, Altai State Technical University, Russia. |
| [8] | M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, G. Shen, G. Woss, C. Yang dan N. Zhang, “Unicorn: A System for Searching the Social Graph,” dalam *Proceedings of the VLDB Endowment*, Trento, 2013. |
| [9] | N. Shervashidze, S. Vishwanathan, T. H. Petri, K. Mehlhorn dan K. M. Borgwardt, “Efficient graphlet kernels for large graph comparison,” dalam *International Conference on Artificial Intelligence and Statistics (AISTATS)*, Florida, 2009. |
| [10] | P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang dan R. Zadeh, “WTF: The Who to Follow Service at Twitter,” dalam *International World Wide Web Conference Committee (IW3C2)*, Rio de Janeiro, 2013. |
| [11] | R. Balakhisnan dan K. Ranganathan, A Textbook of Graph Theory, Springer Science & Business Media, 2012. |
| [12] | R. Kumar, P. Raghavan, S. Rajagopalan, D. Shivakumar, A. S. Thomkins dan E. Upfal, “The Web as a Graph,” dalam *ACM*, Texas, 2000. |
| [13] | Y. Lim, U. Kang dan C. Faloutsos, “SlashBurn: Graph Compression and Mining beyond Caveman Communities,” dalam *IEEE Transaction On Knowledge and Data Engineering*, 2014. |