ISBN: 978-602-6465-01-6

Prosiding Seminar Sains Antariksa

Homepage:http//www.lapan.go.id

INTEGRASI ALGORITMA *DIVIDE AND CONQUER* DENGAN *FORCE DIRECTED GRAPH* UNTUK MEMVISUALISASIKAN MONITORING JARINGAN PENGAMATAN CUACA ANTARIKSA

(Integration Divide and Conquer Algorithm With Force Directed Graph For Network Monitoring Visualizing Space Weather Observations)

Ahmad Zulfiana Utama
Pusat Sains Antariksa,
Lembaga Penerbangan dan Antariksa.
e-mail: ahmad.zulfiana@lapan.go.id

Riwayat Artikel:

Diterima:
Direvisi:
Disetujui:
Diterbitkan:

Kata kunci:

Jaringan Komputer, Monitoring Jaringan, Algoritma FDG, DNC, Visualisasi Graf

ABSTRAK

Sebuah masalah dapat diselesaikan dengan beragam tipe algoritma. Contohnya dalam masalah penggambaran graf, terdapat beberapa algoritma untuk menyelesaikannya. Setiap algoritma memiliki kompleksitas yang berbeda. Kompleksitas algoritma berpengaruh terhadap waktu pemrosesan, dan kebutuhan ruang memori pada komputer. Algoritma dapat dikatakan optimal apabila pada saat pemrosesan memiliki waktu yang minimal dan membutuhkan sedikit memori. Penggambaran graf pada komputer memerlukan informasi tata letak simpul dan garis. Membutuhkan suatu algoritma untuk menentukan tata letak simpul dan garis tersebut. Salah satu metode untuk menghitung tata letak (x,y) simpul dan garis menggunakan algoritma *Force Directed Graph* (FDG). Algoritma FDG membutuhkan minimal 400 iterasi, dan setiap iterasi menghitung O(|E|) attractive force, dan O(|V2|) *repulsive force*. Untuk memperbaiki kompleksitas perhitungan FDG, ditambahkan algoritma *Divide and Conquer* (DNC). Hasilnya algoritma yang menggunakan DNC terlihat dari waktu pemrosesannya lebih optimal dengan rata-rata 1:4. Algoritma tersebut akan diimplementasikan ke dalam monitoring jaringan Pussainsa Bandung. Monitoring jaringan tersebut disajikan dalam bentuk visual (graf).

Keywords:

Computer Network, Network Monitoring, Force Directed Graph, Divide And Conquer, Visualization Graph.

ABSTRACT

A problem can be solved with various types of algorithms. An example in the graph depiction problem, there are several algorithms to solve them. Each algorithm has a different complexity. The algorithm complexity influence the processing time and memory space requirements on the computer. The algorithm can be said to be optimal if at the time of processing has a minimal time and requires less memory. Depictions graph on a computer requires layout information nodes and lines. Requires an algorithm to determine the layout of the node and the line. One method to calculate the layout (x, y)

ISBN: 978-602-6465-01-6

vertices and lines using algorithms Force Directed Graph (FDG). FDG algorithm requires a minimum of 400 iterations, and each iteration calculate the O (| E |) attractive force, and O (| V2 |) repulsive force. To improve processing complexity FDG, added algorithms Divide and Conquer (DNC). The result is an algorithm that uses DNC seen from a more optimal processing time by an average of 1: 4. The algorithm will be implemented for network monitoring Space Science Center (Pussainsa Bandung). Network Monitoring is presented in visual form (graph).

1. Pendahuluan

Algoritma merupakan langkah-langkah yang dituju untuk menyelesaikan suatu masalah. Langkah-langkah tersebut sifatnya berurutan, pada setiap langkahnya menghasilkan suatu nilai yang digunakan untuk menentukan langkah selanjutnya. Algoritma menerima suatu masukan nilai, nilai tersebut akan diproses pada setiap langkahnya dan pada hasil akhir akan menghasilkan suatu nilai baru.

Setiap algoritma memiliki kompleksitas yang berbeda. Kompleksitas algoritma berpengaruh terhadap waktu pemrosesan, dan kebutuhan ruang memori pada komputer. Algoritma dapat dikatakan optimal apabila pada saat pemrosesan memiliki waktu yang minimal dan membutuhkan sedikit memori. Sebuah masalah dapat diselesaikan dengan berbagai tipe algoritma. Dalam penggambaran graf terdapat beberapa tipe algoritma. Salah satu algoritma yang dikenal adalah *Force Directed Graph*.

Graf adalah suatu bentuk visualisasi yang penting dalam menjelaskan keterhubungan antara objek (Mazza, 2004). Graf merupakan himpunan dari simpul dan keterhubungannya direpresentasikan oleh suatu garis. Ketika simpul memiliki asosiasi yang banyak, atau jumlah simpul yang digambar berjumlah banyak, tentu penggambarannya menjadi lebih rumit dan sulit. Karena simpul tidak memiliki informasi mengenai posisi (x,y) dalam komputasi, maka dibutuhkan cara untuk menghitung peletakan posisi dari setiap simpul.

Algoritma *Force Directedd Graph* (FDG) menggunakan representasi *spring model*, dimana simpul-simpul dianggap sebagai benda-benda yang bermuatan (*repulsive force*) dan dihubungkan oleh garis yang dianggap sebagai pegas (*attractive force*). Benda-benda yang bermuatan tersebut saling tolak menolak. Algoritma ini dikembangkan oleh Fruchterman dan Reingold dan representasi pegas yang digunakannya dalam algoritma ini dikembangkan oleh Kamada Kawai. Untuk mendapatkan hasil yang optimal maka algoritma ini membutuhkan minimal 400 iterasi (Fruchterman, 1991) Setiap iterasi menghitung O(|E|) *attractive force* dan O(|V|2) *repulsive forces*. (Kobourov, 2013)

Algoritma *Divide and Conquer* (DNC) adalah algoritma yang membagi masalah menjadi sub-masalah, menghitung solusi dari setiap sub-masalah, dan sub-masalah digabungkan kembali beserta solusinya. Hasil akhir dari algoritma ini mencari nilai optimal dari solusi sub-masalah. Algoritma ini sering digunakan untuk memecahkan masalah pencarian, dan pengurutan data. Algoritma DNC akan diimplementasikan pada algoritma FDG, caranya yaitu membagi graf menjadi sub-graf. Setelah itu dilakukan perhitungan FDG pada setiap sub-graf kemudian sub-graf digabungkan kembali. Tujuannya untuk mengurangi kompleksitas kuadrat pada perhitungan FDG.

Integrasi algoritma DNC dan FDG akan di implementasikan pada visualisasi monitoring jaringan Pusat Sains Antariksa. Waktu yang dibutuhkan dalam proses visualisasi lebih cepat, dan menghasilkan bentuk graf yang lebih simetris.

Tujuan dari visualisasi monitoring jaringan untuk menggambarkan keterhubungan jaringan Stasiun Pengamatan Pussainsa. Simpul merepresentasikan *node* jaringan dan keterhubungannya digambarkan oleh garis. Harapan dari pembuatan monitoring jaringan ini agar informasi lebih mudah dimengerti oleh pengguna monitoring jaringan.

ISBN: 978-602-6465-01-6

2. Landasan Teori

Visualisasi Informasi

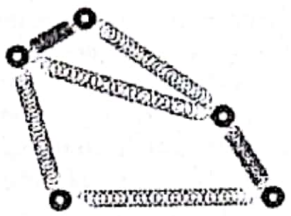
Visualisasi informasi adalah metode dalam teknologi informasi untuk menampilkan data dan menyampaikan informasi dengan tujuan agar metode tersebut dapat menampilkan data dari sudut pandang yang berbeda sehingga menghasilkan suatu informasi yang baru. Visualisasi informasi adalah sebuah proses dimana informasi dirubah ke dalam bentuk visual. Dilihat dari sisi Ilmu Komputer, visualisasi menggunakan teknik pencitraan grafis. Disamping itu. visualisasi informasi melibatkan seluruh pengguna yang terkait. Dengan demikian, pengembang harus memperhitungkan persepsi dari pengguna, kemampuan kognitif, dan karakteristik dari informasi tersebut. Visualisasi bukan sekedar gambar biasa. Visualisasi yang sukses dapat mengurangi waktu pengguna dalam mendapatkan suatu informasi (Mazza, 2004)

Transformasi data dari visualisasi informasi bertujuan untuk mengubah data abstrak ke dalam bentuk grafis. Transformasi data tersebut merupakan proses kreatif dimana perancang ataupun pengembang memberikan arti yang baru ke dalam pola-pola grafis. Dalam sudut pandang seni, visualisasi informasi bertujuan untuk mengkomunikasikan ide yang rumit sehingga para pengguna atau yang melihatnya mendapatkan informasi dalam persfektif yang lain (McKelvey, 2012)

Intisari dari visualisasi adalah informasi atau data yang disajikan menjadi suatu wawasan menarik bagi pengguna. Secara umum wawasan didefinisikan sebagai suatu penemuan yang tak terduga, memperdalam pemahaman dari cara baru untuk berpikir, dan terobosan intelektual (Zulfiana, 2012)

Algoritma *force directed graph*

Algoritma *Force Directed Graph* menggunakan representasi *spring model*, dimana simpul-simpul dianggap sebagai benda-benda bermuatan dan dihubungkan (ditarik) oleh garis yang dianggap sebagai pegas. Benda-benda yang bermuatan tersebut saling tolak menolak (Fruchterman, 1991)



Gambar 2-1. Model spring (Fruchterman, 1991)

Ada 4 langkah utama dalam Algoritma *Force Directed* Graph (Fruchterman, 1991).

Hitung daya tolak menolak antar simpul. (*Inverse Square Law / Coulomb Law*)

1. Hitung daya tarik menarik diantara semua simpul yang terhubung oleh garis (*Hooke's Law*).
2. Jumlahkan daya tolak menolak dan daya tarik menarik.
3. Penggantian posisi simpul awal dengan posisi baru.

Metode Fruchterman dan Reingold membuat persamaan untuk mencari gaya tarik-menarik (*attractive force*) dan gaya tolak-menolak (*repulsive force*) yang persamaanya dituliskan sebagai (Fruchterman, 1991) :

$$Fa(d) = \frac{d^2}{k} \quad \text{.....(2-1)}$$

$$Fr(d) = \frac{-k^2}{d} \quad \text{.....(2-2)}$$

dengan d merupakan jarak antar 2 simpul yang bertetangga dan k adalah konstanta untuk penempatan simpul yang terdistribusi secara merata. Konstanta k dinyatakan dengan :

$$k = \sqrt{\frac{area\ of\ frame}{|V|}} \quad \text{.....(2-3)}$$

Area of frame adalah perkalian dari panjang dan lebar area penggambaran (Luas), dan $|V|$ jumlah simpul yang akan digambar. Fruchterman dan Reingold menambahkan variable t dikenal dengan istilah temperatur. Dirumuskan sebagai berikut :

ISBN: 978-602-6465-01-6

$$t = \frac{width}{10} \text{.....(2-4)}$$

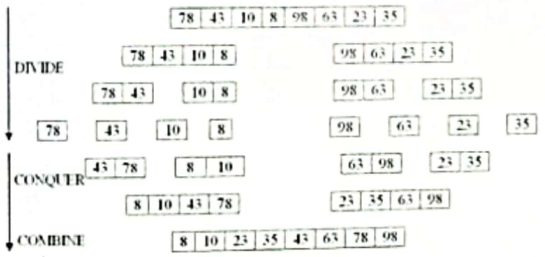
Manfaat nilai t dipakai untuk mempercepat penempatan simpul baru, karena lebar frame diperkecil sepersepuluh dari area sebelumnya (Eades, 1994)

Algoritma *Divide and Conquer*

Algoritma *Divide and Conquer* dibagi ke dalam 3 tahap penting yaitu:

1. *Divide* adalah membagi masalah menjadi beberapa sub-masalah yang memiliki kemiripan dengan masalah semula namun berukuran lebih kecil
2. *Conquer* adalah memecahkan solusi dari masing-masing sub-masalah.
3. *Combine* adalah penggabungan solusi masing-masing sub-masalah sehingga membentuk solusi masalah semula.

Objek permasalahan yang dibagi adalah tabel (larik), matrik, eksponen, dan sebagainya. Tiap-tiap sub masalah mempunyai karakteristik yang sama. Ilustrasi algoritma DND pada masalah pengurutan angka :



Gambar 2-2. Ilustrasi pemecahan masalah pada algoritma *Divide and Conquer*.

3. Data dan Metode

Indikator pertama untuk membandingkan Algoritma FDG dan DND menggunakan kriteria yang dibuat oleh Batista, parameter yang diukur adalah sebagai berikut (Battista, 1999):

1. Jumlah garis yang bersilangan / *Intersections* (K1).
2. Luas area (K2).
3. Total panjang garis yang menghubungkan antar simpul / *Total Distance* (K3).
4. Variasi panjang antar garis / *Unique Length* (K4).
5. Menampilkan sifat simetri pada hasil akhir penggambaran graf (K5).

Graf yang memiliki nilai minimum dari kriteria di atas merupakan graf yang paling optimal. Selain dari kriteria di atas, indikator kedua adalah waktu pemrosesan pada saat kedua algoritma dijalankan. Implementasi untung menghitung kriteria di atas adalah sebagai berikut

Jumlah garis yang bersilangan (*Intersection*)

Suatu garis dikatakan bersilangan apabila kedua garis tersebut saling berpotongan. Berikut adalah *source code* pada bahasa *javascript* untuk menghitung *intersection* (Zulfiana, 2012):

```
1. For(i = 0;i<=ListEdge.length-1;i++)
2.    For(j=0;j<=ListEdge.length-1;j++){
3.       If(i!=j){
4.                var u  = ListEdge[i].vertex2;
5.                var v  = ListEdge[i].vertex1;
6.                var u2 = ListEdge[j].vertex2;
7.                var v2 = ListEdge[j].vertex1;
8.                var i1x  = u.x - v.x;
9.                var i2x  = u2.x - v2.x;
10.               var i1y  = u.y - v.y;
11.               var i2y  = u2.y - v2.y;
12.               var a = (-i1y * (v.x - v2.x) + i1x *
    (v.y - v2.y)) / (-i2x
•   i1y + i1x * i2y);
13.               var b = (i2x * (v.y - v2.y) – i2y *
    (v.x – v2.x)) / (-i2x *
14.i1y + i1x * i2y);
15.      if (a > 0 && a < 1 && b > 0 && b < 1){
16.               z +=1;
17.         }
18.   }
19. }
20.}
```

Jarak garis

Menghitung jarak garis yang menghubungkan antar *vertex* menggunakan rumus *euclidean distance* :

$$d(x,y) = \text{(3-1)}$$

pada persamaan (3-1), (x_1,y_1) merupakan titik awal, dan (x_2,y_2) titik akhir. Penggunaan *euclidean distance* pada pemrograman *javascript* (Zulfiana, 2012):

```
1.  For(i=0;i<=ListEdge.length-1;i++){
2.  var u = ListEdge[i].vertex1;
3.  var v = ListEdge[i].vertex2;
4.  var dx = v.x - u.x;
5.  var dy = v.y - u.y;
6.  var dist = Math.sqrt(dx*dx + dy*dy);
7.  TotalDist += dist;
8.          };
```

Variable *TotalDist* berfungsi untung menampung nilai dari seluruh jarak garis. Sehingga variable *TotalDist* sama dengan nilai total jarak garis secara keseluruhan.

Variasi panjang antar garis

Untuk menghitung variasi panjang garis, ada beberapa langkah :

1. Hasil kriteria (K2), dimasukan ke dalam sebuah list.
2. Cek nilai dari panjang setiap garis dengan semua garis yang ada($n..n_n$).
3. Apabila nilai n sama dengan n_1-n_n maka masukan nilai n ke dalam variabel total variasi panjang garis. Hal ini bersifat rekursif.

Implementasi pada bahasa pemgrograman (Zulfiana, 2012):

```
1.  Array.prototype.unique = function () {
2.    var r = new Array();
3.    o:for(var i = 0, n = this.length; i < n; i++)
4.    {
5.      for(var x = 0, y = r.length; x < y; x++)
6.      {
7.            if( r[x] == this[i] || r[x]
    ==this[i]-2 ||
8.                  r[x] == this[i]-1 || r[x]
    == this[i]+1 ||
9.          r[x] == this[i]+2)
10.          {
11.            continue o;
12.          }
13.     }
14.     r[r.length] = this[i];
15.   }
16.   return r;
17. }
```

Perangkat keras yang digunakan dalam penelitian ini sebagai berikut:

1. Processor Intel Core 2 Duo
2. RAM 2 Ghz.

Perangkat lunak dan bahasa pemrograman yang digunakan:

1. Sistem Operasi Windows 7 64 bit.
2. Xampp 3.2.1
3. Browser.
21. Bahasa pemrograman PHP, Javascript, SVG, dan *Query Language*.

Data

Sumber data yang digunakan yaitu Alamat IP dari setiap Stasiun Pengamatan yang terhubung. Untuk menjaga keamanan jaringan, pada makalah ini tidak menampilkan IP yang sebenarnya tetapi diganti dengan kode yang mewakili nama Stasiun Pengamatan. Adapun Stasiun Pengamatan yang terhubung ke dalam jaringan adalah sebagai berikut :

Tabel 3-1

Daftar Stasiun Pengamatan yang terhubung dengan server utama (Pussainsa Bandung).

No	Site	Kode
1.	Bandung	BDG
2.	Pontianak	PTK
3.	Sumedang	SMD
4.	Biak	BIK
5.	Pameungpeuk	PMK
6.	Watukosek	WTK
7.	Kupang	KPG
8.	Yogyakarta	YGY

Selain server Stasiun Pengamatan, pada bab hasil pembahasan akan dirinci secara lebih detail komputer kontrol dari setiap Stasiun Pengamatan. Berikut adalah sebagaian komputer kontrol (alat pengamatan) dari Stasiun Pengamatan Sumedang dan Biak :

Tabel 3-2

Daftar IP komputer kontrol pada alat pengamatan.

No	Stasiun Pengamatan	Komputer Kontrol
1.	Sumedang	Beacon
2.	Sumedang	IPS71
3.	Sumedang	Celestron
4.	Sumedang	Fluxgate
5.	Sumedang	AWS
6.	Sumedang	SN4000
7.	Beacon	Beacon
8.	ALE	ALE
9.	...	...
10.	...	...

Monitoring jaringan yang akan dilakukan pada komputer kontrol dengan cara melakukan perintah *ping* secara berkala. Perintah *ping* ini berfungsi untuk memeriksa suatu koneksi perangkat jaringan atau komputer (Laverty, 2007) Apabila paket *ping* memberi suatu balasan dengan nilai yang terdiri dari alamat IP, waktu, *Time To Live* (TTL), dan *Bytes* maka hal ini berarti komputer tersebut terhubung ke dalam jaringan. Jika memberikan pesan *error*

unreachable host, atau *request time out* maka hal ini berarti ada permasalahan jaringan pada perangkat tersebut. Hasil dari perintah *ping* ditampung ke dalam basis data, dan dijadikan paramater pada visualisasi monitoring jaringan dalam bentuk graf.

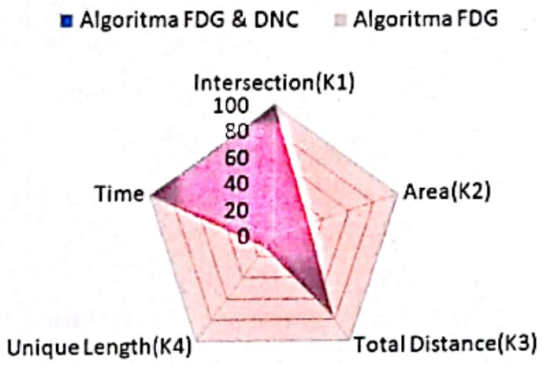
4. Pembahasan

Hasil komputasi penggambaran graf dengan jumlah 45 simpul dan 45 garis/asosiasi, ditampilkan sebagai berikut :

Tabel 4-1
Hasil perbandingan algoritma DNC dan FDG

	Algoritma DNC	Algoritma FDG
Intersections (K1)	0	0
Area (K2)	3869	13626
Total Distance (K3)	6821	8820
Unique Length (K4)	3	25
Time	52 ms	52 ms

Tabel 4-1 menunjukan bahwa tidak terdapat perpotongan garis di antara kedua algoritma dengan waktu pemrosesan membutuhkan waktu yang sama yakni 52 ms. Terdapat perbedaan yang mencolok pada kriteria K2 (area) sampai K4 (variasi panjang garis), yakni nilai algoritma DNC selalu lebih kecil dibandingkan dengan algoritma FDG sehingga dapat dikatakan algoritma DNC lebih baik dibandingkan algoritma FDG.



Gambar 4-1. Perbandingan algoritma DNC dengan FDG, DNC persentase lebih kecil pada kriteria K2-K4.

Sebelum diproses, algoritma DNC dan FDG simpul diletakan secara acak menggunakan fungsi *random* yang terdapat pada bahasa pemrograman *javascript*. Pengujian dilakukan sebanyak 5 kali, hal ini untuk memastikan keakuratannya.

Tabel 4-2
Hasil 5 kali pengujian akurasi antara DNC dan FDG.

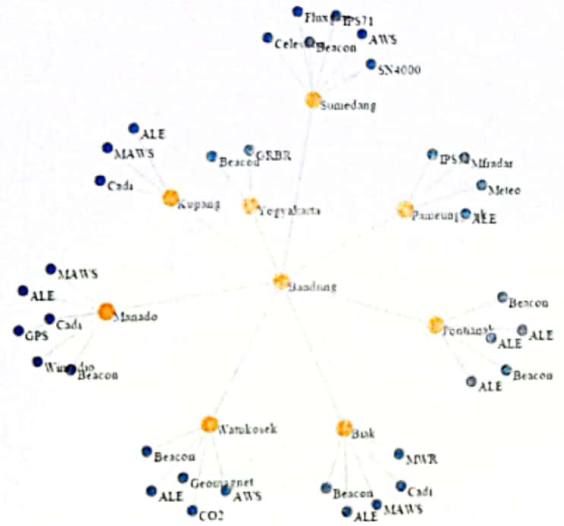
Percobaan Ke-		Kriteria			
		K1	K2	K3	K4
1	A	0	4804	6823	4
	B	0	19892	8818	25
2	A	0	5347	6820	3
	B	0	10592	8818	22
3	A	0	5329	6821	3
	B	0	20581	8821	23
4	A	0	5094	6724	3
	B	0	13779	8809	23
5	A	0	4083	6820	3
	B	0	14080	8790	24

Keterangan : A = DNC, B = FDG

Pada Tabel 4-2 terlihat bahwa hasil algoritma DNC secara berturut turut selalu lebih kecil pada semua kriteria. Waktu pemrosesan pada saat pengujian untuk kedua algoritma rata-rata 50 ms. Salah satu hasil dari pengujian adalah sebagai berikut :

ISBN: 978-602-6465-01-6

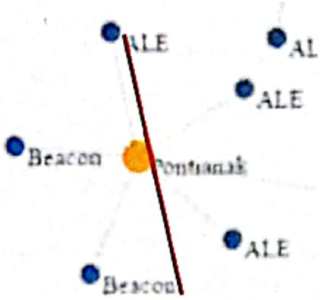
(a)



(b)

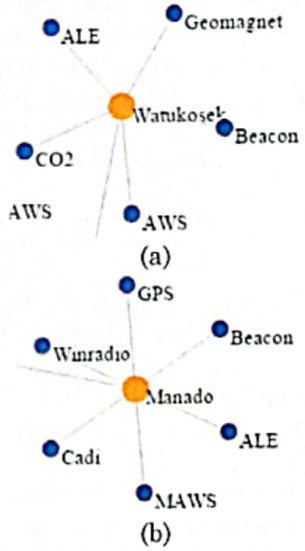
Gambar 4-2. a) Penggambaran graf menggunakan algoritma DNC, b) algoritma FDG.

Pada penggambaran graf, posisi (x,y) setiap simpul tidak perlu dihitung, cukup memasukan jumlah titik dan keterhubungan antar titik. Terlihat perbedaan hasil akhir pada penggambaran graf (Gambar 4-2). Pada algoritma DNC, simpul tersebar secara merata dan terlihat simetris. Tetapi, berbeda halnya dengan algoritma FDG, sebaran simpul berat pada satu titik. Dikatakan simetris karena apabila ditarik suatu garis lurus pada titik tengah, jumlah simpul pada ruas kanan dan kiri berjumlah sama.



Gambar 4-3. Kriteria garis simetris

Cara kerja algoritma DNC pada penggambaran graf yaitu membagi graf ke dalam sub-graf. Visualisasi graf pada monitoring jaringan membagi sub-graf berdasarkan Stasiun Pengamatan. Setiap sub-graf memecahkan solusinya masing-masing, kemudian hasil akhirnya digabungkan sehingga terbentuk solusi akhir.



Gambar 4-4. Proses pembagian graf menjadi sub-graf.

Kompleksitas perhitungan DNC yaitu O(n!) dikalikan jumlah sub-graf. Berbeda halnya dengan algoritma FDG, membutuhkan perhitungan sebanyak $400*O(|V|^2)*(|E|)$. Jumlah tahapan yang dikerjakan algoritma DNC pada kasus ini lebih sedikit dibandingkan dengan FDG. Waktu pemrosesan akan terasa berbeda apabila jumlah simpul dan asosiasi berjumlah di atas 100 simpul. Hal ini berdasarkan kompleksitas kuadrat.

Implementasikan hasil di atas digunakan untuk memvisualisasikan monitoring jaringan pengamatan sains antariksa. Berikut adalah

langkah-langkah untuk memvisualisasikan monitoring jaringan

1. *Ping* setiap IP dari seluruh Stasiun Pengamatan. Jalankan perintah tersebut menggunakan aplikasi *scheduler* setiap 1 menit sekali.
2. Masukan (*update/insert*) hasil pengecekan langkah 1 ke dalam basis data.
3. Tampilkan hasil dalam bentuk graf menggunakan algoritma DNC. Pemilihan algoritma ini berdasarkan hasil dari perbandingan.

Hasil scan IP jaringan pengamatan sains antariksa sebagai berikut :

Tabel 4-3: Daftar IP komputer alat.

id_lokasi	nama_alat	Status
1	AWS	1
1	Beacon	1
1	SN4000	1
1	Celestron	1
1	Fluxgate	1
8	Beacon	0
5	Mfradar	0
5	Meteo	0
5	IPS51	0
5	Ale	0

Tabel 4-4
Daftar IP Stasiun Pengamatan

id_lokasi	nama_lokasi	Status
1	Sumedang	1
2	Watukosek	1
3	Pontianak	1
4	Manado	1
5	Pameungpeuk	0
6	Biak	1
7	Kupang	1
8	Yogyakarta	1

Pada Tabel 4-3 dan 4-4 kolom status yang bernilai 1 menandakan bahwa jaringan tersebut terhubung, dan status dengan nilai 0 tidak terhubung ke dalam jaringan. Hasil dari kedua tabel di atas menjadi parameter untuk melakukan viasualisasi ke dalam bentuk graf. Stasiun Pengamatan yang mengalami gangguan jaringan akan diberi warna merah pada simpul dan garis.

Gambar 4-5. Hasil monitoring jaringan pusat sains antariksa.

Simpul BPAA Garut berwarna merah mengindikasikan simpul tersebut mengalami gangguan. Selain itu, komputer kontrol *Beacon* di Yogyakarta mengalami hal yang serupa. Monitoring jaringan dalam bentuk visualisasi graf menampilkan suatu informasi dalam grafis, diharapkan mempermudah pengelola jaringan untuk mengidentifikasi, dan mengetahui masalah pada jaringan.

5. Kesimpulan

Algoritma DNC dapat diaplikasikan pada penggambaran graf. Hasil dari perbandingan antara algoritma DNC dan FDG, menunjukan bahwa DNC lebih optimal. Perbandingan ini mengacu pada kriteria Batista, dan waktu pemrosesan.

Monitoring jaringan data pengamatan Pussainsa Bandung dapat divisualisasikan dalam bentuk graf. Simpul-simpul merepresentasikan Stasiun Pengamatan atau komputer alat dan garis merepresentasikan hubungan antar simpul. Algoritma *Force Directed Graph* (FDG) membantu penghitungan komputasi tata letak simpul dan garis. Hasil perhitungan algoritma FDG menghasilkan graf yang simetris dan simpul yang digambar terdistribusi secara merata. Visualisasi akan mempermudah pengelola dalam memantau jaringan data pengamatan Pussainsa.

Ucapan Terima Kasih

Ucapan terima kasih disampaikan kepada Prof. Tarzan Sembiring selaku pembimbing pada diklat peneliti tingkat pertama 2014 yang

telah memberikan dorongan dan bantuan serta masukan mengenai tatacara penulisan makalah. Ucapan terima kasih juga disampaikan kepada bapak Yusuf Dirgantara yang telah memberikan data IP Balai/stasiun kerjasama pengamatan cuaca antariksa dan penjelasan mengenai topologi jaringan.

Rujukan

Battista G., Eades P., Tamassia R., and Tollis I. (1999). *Graph Drawing: Algorithms for the visualization of graphs*. Prentice-Hall (New Jersey).

Eades P., and Tamassia R. (1994). *Algorithms for drawing graphs: an annotated bibliography*. The University of Texas (USA).

Fruchterman., Thomas, M.J. and Reingold, E. (1991). *Graph Drawing by Force-directed Placement*. Department of Computer Science (USA).

Held, P., Hempel, J., and Kruse, R. (2013). Cluster-based Visualization of Dynamic Graphs. In *Proceedings. 23. Workshop Computational Intelligence, Dortmund, 5.-6. Dezember 2013* (p. 21). KIT Scientific Publishing.

Kobourov, S.G. (2013). *Force Directed Drawing Algorithms. Handbook of Graph Drawing and Visualization*. CRC Press:383~408

Knuth, D. E. (1993). *The Stanford GraphBase: a platform for combinatorial computing* (Vol. 37). Addison-Wesley (Reading).

Laverty, D., Morrow, D. J., and Littler, T. (2007, September). *Internet based loss-of-mains detection for distributed generation*. In *Universities Power Engineering Conference, 2007. UPEC 2007. 42nd International* (pp. 464-469). IEEE.

Mazza, R. (2004). *Introduction to Information Visualisation*. University of Lugano (Italia).

McKelvey, K., Rudnick, A., Conover, M. D., & Menczer, F. (2012). *Visualizing communication on social media: Making big data accessible. arXiv preprint arXiv:1202.1367*.

Pressman R.S., Maxim BR. (1994). *"Software Engineering: A Practitioner's Approach"*. McGraw-Hill (NY)

Sofana, I. (2012). *Cisco CCNP dan Jaringan Komputer*. Penerbit Informatika (Bandung).

Zulfiana, A. (2012). *Perbandingan Algoritma Force Directed Graph dengan Random Geometric Graph Pada Penggambaran Graf*. Skripsi, FPMIPA. Universitas Pendidikan Indonesia (Bandung).