# Aplikasi Graf Berbobot dalam Optimasi Jaringan Komputer

I Dewa Putu Deny Krisna Amrita – NIM : 13514096
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*idpdka@yahoo.com*

***Abstract – Kecepatan dan kenyamanan dalam mengakses internet merupakan suatu kebutuhan yang cukup esensial di zaman teknologi informasi yang semakin berkembang ini. Seringkali pengguna internet mengeluhkan koneksi internet yang kurang baik, terutama di Indonesia yang memiliki infrastruktur jaringan komputer yang masih kurang baik dibandingkan dengan infrastruktur di negara lain. Infrastruktur sangat berhubungan dengan kondisi geografis, yang dapat dipetakan dengan suatu graf Dalam makalah ini akan dibahas penggunaan graf berbobot dalam optimasi jaringan komputer dalam suatu wilayah.***

***Kata Kunci – graf, graf berarah, graf berbobot, jaringan komputer,koneksi internet, optimasi.***

## I. PENDAHULUAN

Dewasa ini, kebutuhan manusia akan teknologi secara bertahap meningkat seiring berjalannya waktu. *Information Technology* (IT) adalah salah satu cabang teknologi yang berkembang dengan sangat pesat dan memiliki efek yang sangat berpengaruh pada kehidupan kita sehari-hari. Internet, *gadget*, teknologi *cloud*, media sosial dan berbagai hal-hal yang lumrah didengarkan oleh masyarakat dunia saat ini adalah produk-produk dari perkembangan IT. Produk yang merupakan "tulang punggung" dari perkembangan IT yang dapat kita sorot adalah internet.

Internet merupakan keseluruhan dari jaringan komputer yang saling terhubung untuk melayani seluruh pengguna (*user*) di seluruh dunia. Internet menggunakan protokol *Transmission Control Protocol/Internet Protocol* (TCP/IP) sebagai protokol pertukaran paket yang digunakan untuk pertukaran informasi yang ada di dunia saat ini. Pada awalnya, internet digunakan untuk pengiriman informasi berupa pesan dan antisipasi terjadinya hal gawat darurat dalam militer Amerika Serikat (Dahulu disebut ARPANET). Saat ini, penggunaan internet dalam kehidupan manusia semakin berkembang seiring perkembangan zaman.

Penggunaan internet saat ini sangat banyak. Diantaranya adalah surat elektronik (*e-mail*), *online chatting*, media sosial, *online game*, dan lain sebagainya. Semakin berkembangnya teknologi internet, semakin meningkat pula kebutuhan akan kenyamanan dan kemudahan akses internet. Semakin banyaknya pertukaran data atau paket yang digunakan menyebabkan semakin besar pula kecepatan internet yang dibutuhkan untuk mempermudah akses dan pertukaran data tersebut serta meningkatkan kenyamanan dalam akses internet untuk penggunanya. Seringkali pengguna mengeluhkan kecepatan dan kenyamanan internet yang dinilai kurang baik, terutama di negara yang berkembang seperti Indonesia.

Pada makalah ini, penulis akan membahas tentang aplikasi dari teori graf dalam optimasi jaringan komputer. Optimasi jaringan komputer ini dapat dilakukan dengan cara memanfaatkan graf berbobot yang dapat digunakan untuk membantu pembuatan infrastruktur jaringan komputer di suatu daerah secara efektif dan efisien tanpa mengabaikan faktor kelancaran dan kenyamanan dalam menggunakan koneksi jaringan komputer tersebut.

## II. DASAR TEORI

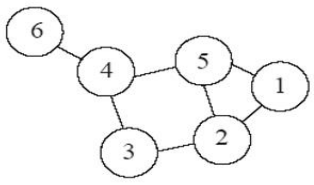
### A. Teori Graf

Graf merupakan sebuah struktur diskrit yang dapat digunakan untuk merepresentasikan hubungan yang terjadi pada suatu objek diskrit yang satu dengan yang lain.

Secara matematis, graf dapat didefinisikan sebagai suatu pasangan himpunan dari himpunan tak kosong dari simpul (V/*Vertex*) dan sisi (E/*Edges*) yang menghubungkan simpul satu dengan yang lainnya. Suatu graf dapat ditulis dengan notasi sebagai berikut :

$$\mathbf{G = (V,E)}$$

Graf dapat direpresentasikan seperti berikut :




V = {1,2,3,4,5,6}
E = {{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}

**Gambar 2.1 : Contoh Representasi dari Graf (Sumber : pustakailmuperpustakaan.files.wordpress.com)**

Graf dapat dikelompokkan menjadi beberapa kategori bergantung dari sudut pandang pengelompokannya. Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, graf dapat digolongkan menjadi :

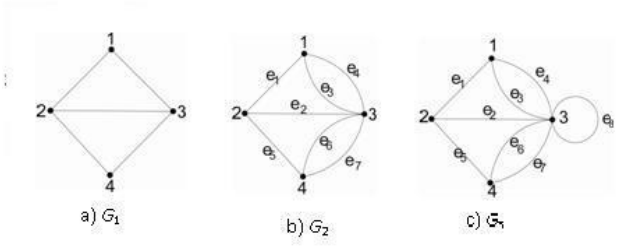
1. Graf Sederhana (*Simple Graph)*

Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi ganda.

2. Graf Tak-Sederhana (*Unsimple Graph*)

Graf tak-sederhana adalah graf yang memiliki sisi ganda atau gelang. Graf tak-sederhana terbagi menjadi dua, yaitu graf ganda (*multigraph)* dan graf semu (*pseudograph).* Graf ganda adalah graf yang memiliki sisi ganda, sedangkan graf semu mengandung gelang.

**Gambar 2.2 Graf Berdasarkan Ada dan Tidaknya Gelang dan Sisi Ganda (Sumber :**




Gambar a) Graf sederhana, b) Graf ganda, dan c) Graf semu

**http://yunikhoirunnisa.blogspot.co.id)**

Berdasarkan jumlah simpul pada graf, graf dapat dibagi menjadi :

1. Graf Berhingga (*limited graph*)

Graf berhingga adalah graf yang memiliki jumlah simpul yang berhingga.

2. Graf Tak-Berhingga (un*limited graph*)

Graf tak-berhingga adalah graf yang memiliki jumlah simpul yang tidak terhingga.

Sedangkan berdasarkan ada-tidaknya orientasi, graf dapat dibagi menjadi :

1. Graf Tak-Berarah (*undirected graph*)

Graf tak-berarah merupakan graf yang tidak memiliki orientasi arah.

2. Graf Berarah (*directed graph* atau *digraph*)

Graf berarah merupakan graf yang memiliki orientasi arah. Tidak seperti graf tak berarah, pasangan simpul yang dihubungkan tidak sama.

Dalam mempelajari teori graf, ada beberapa terminologi atau istilah yang harus diperhatikan. Terminologi yang harus diperhatikan adalah :

1. Bertetangga (*Adjacent)*

Dua buah simpul pada graf tak-berarah G dikatakan bertetangga apabila keduanya terhubung langsung dengan sebuah sisi.

2. Bersisian (*Incident)*

Untuk sembarang sisi $e = (v_j , v_k)$ sisi *e* dikatakan bersisian dengan simpul $v_j$ dan $v_k$.

3. Simpul Terpencil (*Isolated Vertex)*

Simpul terpencil adalah simpul yang tidak memiliki sisi yang bersisian dengan simpul tersebut atau tidak bersisian dengan simpul-simpul lainnya.

4. Graf Kosong (*Null Graph* atau *Empty Graph)*

Graf kosong adalah graf yang sisinya merupakan himpunan kosong.

5. Derajat (*Degree)*

Derajat suatu simpul pada graf tak-berarah adalah jumlah sisi yang bersisian dengan simpul tersebut.

6. Lintasan (*Path)*

Lintasan yang panjangnya n dan tercipta dari simpul awal $v_0$ ke simpul tujuan $v_n$ di dalam graf G adalah sisi-sisi dari graf G.

7. Siklus (*Cycle)* atau Sirkuit (*Circuit*)

Siklus atau sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama.

8. Terhubung *(Connected)*

Sebuah graf tak-berarah G disebut graf terhubung apabila untuk setiap pasang simpul $v_i$ dan $v_j$ di dalam himpunan V terdapat lintasan dari $v_i$ ke $v_j$.

9. Upagraf (*Subgraph)*

Upagraf adalah sebuah bagian dari graf G. Dengan kata lain, upagraf merupakan *subset* dari sebuah graf G.

10. Upagraf Merentang (*Spanning Subgraph)*

Upagraf merentang adalah upagraf dari sebuah graf G dimana mencakup semua simpul dari graf G.

11. *Cut-Set*

*Cut-set* dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung.

12. Graf Berbobot (*Weighted Graph)*

Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).

## B. Algoritma Dijkstra

Algoritma Dijkstra merupakan sebuah algoritma yang digunakan untuk memecahkan persoalan yang berhubungan dengan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah dan berbobot positif. Algoritma ini ditemukan oleh Edsger Dijkstra.

Input dari algoritma ini adalah sebuah graf berarah dan berbobot G dan sebuah simpul awal s dan simpul akhir t dalam G dan V adalah himpunan semua simpul dalam graf G. Bobot dari semua sisi dihitung dengan fungsi :

$$w: E \rightarrow [0, \infty)$$

Bobot yang dihitung dimulai dari bobot yang paling kecil, dimana akan dihubungkan dan dijumlahkan semua bobot yang dilewati dari simpul awal s sampai mencapai simpul akhir t.

Algoritma Dijkstra dapat dinyatakan dalam pseudo-code sebagai berikut :

```
function Dijkstra(Graph, source):
    for each vertex v in Graph:
        dist[v] := infinity ;

        previous[v] := undefined ;
    end for

    dist[source] := 0 ;
    Q := the set of all nodes in Graph ;

    while Q is not empty:
        u := vertex in Q with smallest distance in dist[] ;
        remove u from Q ;
        if dist[u] = infinity:
            break ;
        end if

        for each neighbor v of u:

            alt := dist[u] + dist_between(u, v) ;
            if alt < dist[v]:
                dist[v] := alt ;
                previous[v] := u ;
                decrease-key v in Q;
            end if
        end for
    end while
return dist;
```

**Gambar 2.3 Pseudo-code dari Algoritma Dijkstra (Sumber : wirasetiawan29.wordpress.com)**

## C. Kecepatan Transfer Internet

Kecepatan transfer internet pada dasarnya merupakan tolok ukur untuk menentukan seberapa banyak sebuah jaringan komputer mampu mengantar data dari *server* menuju komputer *client* atau sebaliknya. Kecepatan transfer internet dinyatakan dalam satuan *bits*, sedangkan ukuran sebuah *file* komputer dinyatakan dalam satuan *bytes* (1 byte = 8 bit). Sebagai contoh, sebuah *Gigabit Ethernet [1000Base-T]* memiliki kecepatan transfer sampai 1000 mbps, atau 1 gbps.

Transfer data dapat dibagi menjadi 2 jenis :

1. *Download*

*Download* merupakan suatu fitur untuk "menarik" data dari sebuah *server* menuju *client*. Kecepatan *download* dapat dinyatakan dalam satuan Mbps (megabits per second).

2. *Upload*

*Upload* merupakan suatu fitur untuk mengirim data dari *client* menuju *server*. Kecepatan *upload* juga dinyatakan dalam satuan Mbps (megabits per second).

Selain kedua istilah tersebut, ada sebuah istilah lain yang bernama *latency*. *Latency* merupakan waktu reaksi dalam suatu koneksi jaringan. *Latency* berguna untuk menentukan waktu respon suatu *request* atau permintaan dari client kepada *server* untuk melakukan sesuatu, baik itu *download*, *upload*, dan lain sebagainya. *Latency* dinyatakan dalam satuan milisecond (ms). Semakin sedikit waktu *latency* yang dibutuhkan untuk merespon sebuah *request*, semakin cepat respon koneksi tersebut. Pada umumnya semakin kecil *latency* yang dihasilkan, koneksi yang digunakan terasa semakin nyaman dan lancar. *Latency* sangat sering digunakan dalam aplikasi yang berhubungan dengan *timing* yang akurat, seperti dalam *online game*. Selain *latency*, terdapat juga istilah yang biasa disebut dengan *ping*. *Ping* merupakan besaran yang sama dengan *latency*. Perbedaannya terdapat pada jalur yang ditempuh. *Latency* membutuhkan jalur satu kali saja dari *client* menuju server, sedangkan *ping* membutuhkan jalur satu kali dari *server* menuju *client*, dan satu kali kembali dari *server* menuju *client*. Secara, teoritis, *latency* dapat dihitung dengan menggunakan rumus :

$$latency = distance / speed$$

Dimana speed merupakan kecepatan cahaya pada ruang vakum, yaitu sebesar 299792 km/second. Untuk menghitung *ping*, dapat dihitung dengan menghitung waktu *latency* dikali dengan 2.

$$ping = 2 \times latency$$

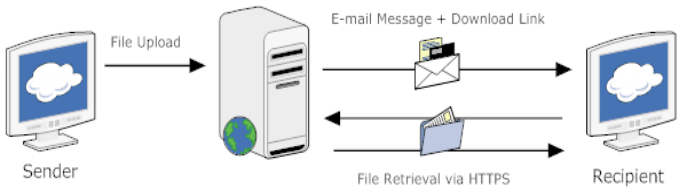
Untuk menghitung *latency* dan *ping* pada komputer anda, dapat dilakukan dengan melakukan hal-hal seperti berikut :

Pada sistem operasi Windows :

1. Buka command prompt (dengan klik start → command prompt).
2. Ketik ' ping <input> -t' (input dapat berupa IP address atau website tertentu).

Pada sistem operasi Linux :

1. Buka terminal (dengan klik start → terminal).
2. Ketik 'sudo ping -f -c <jumlah paket> -s <size paket> <input>' (input dapat berupa IP address atau website tertentu).

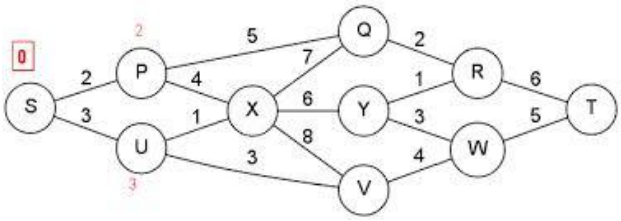



**Gambar 2.4 Skema Transfer Data (Sumber : http://pro2col.com)**

## III. PEMBAHASAN

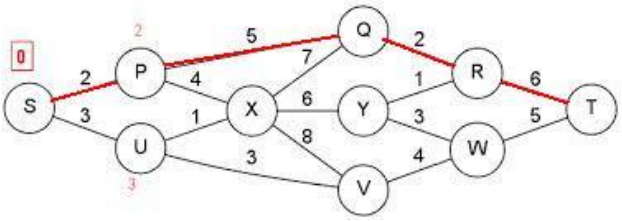
### A. Penentuan Jarak Terdekat Menggunakan Algoritma Dijkstra

Untuk menentukan kecepatan sebuah paket data dikirim dari *server* menuju *client* atau sebaliknya, dibutuhkan jarak terpendek berdasarkan kondisi geografis dari suatu daerah. Semakin kecil jarak yang dibutuhkan untuk mengantar paket data tersebut, semakin cepat waktu responnya, dan semakin sedikit pula kemungkinan paket data tersebut untuk terjadi *packet data loss*, atau kehilangan bagian dari data yang dikirim. Untuk menentukan jarak terpendek dari suatu simpul awal menuju simpul akhir dapat digunakan algoritma Dijkstra. Sebagai contoh, perhatikan gambar graf berikut :




**Gambar 3.1 Contoh Graf Berbobot (Sumber : wikipedia.org)**

Graf tersebut merupakan graf berbobot dengan bobot jarak suatu simpul dengan tetangganya. Bobot pada graf tersebut dinyatakan dalam satuan seribu kilometer (1000 km). Misalkan simpul awal S adalah sebuah *server* dan simpul akhir T adalah komputer dari *client.* Kita dapat menghitung jarak terpendek dari S ke T dengan algoritma Dijkstra. Perhatikan gambar graf berikut :




**Gambar 3.2 Jarak Terpendek yang Didapat Menggunakan Algoritma Dijkstra (Sumber : wikipedia.org)**

Dari gambar 3.2 dapat dicari jarak terpendek dari simpul awal S menuju simpul awal T. Jarak terpendek yang dapat dicapat adalah :

**D(S,T) = 2000 + 5000 + 2000 + 6000 = 15000 km.**

Lintasan yang dilewati adalah S – P – Q – R – T. Jarak tersebut juga dapat dicapai jika lintasan yang dilewati adalah S - U – V – W – T.

### B. Penentuan Latency dan Ping Secara Teoritis

Seperti yang sudah disebutkan dalam dasar teori bahwa *latency* dan *ping* merupakan suatu besaran untuk mengukur seberapa cepat respon *request* dari *client* kepada *server*. Perhitungan *latency* dan *ping* secara teoritis dapat digunakan untuk menentukan kecepatan maksimal dari *latency* atau *ping* tersebut. Untuk menentukan *latency* dan *ping* dari suatu simpul awal S sebagai *server* dan simpul akhir T sebagai *client*, dapat dihitung dengan menggunakan rumus :

$$latency = distance / speed$$

Dengan menggunakan jarak terpendek yang dihitung dengan menggunakan algoritma Dijkstra pada poin 3. A., dan menggunakan asumsi bahwa paket data yang dikirim merupakan cahaya yang bergerak dengan ruang vakum dengan kecepatan 299792 km/second, maka kita dapat menghitung *latency* dan *ping* dengan perhitungan sebagai berikut :

$$latency = distance / speed$$

$$latency = 15000\ km / 299792\ km/s$$

$$latency = 0.0500346907188984\ s$$

Dari perhitungan di atas didapat bahwa *latency* dari simpul awal S menuju simpul akhir T adalah 0.0500346907188984 sekon, atau 50034,6907188984 ms. Sedangkan *ping* yang didapat pada kasus ini dapat dihitung dengan perhitungan sebagai berikut :

$$ping = 2\ x\ latency$$

$$ping = 2\ x\ 0.0500346907188984\ s$$

$$ping = 0.1000693814377968\ s$$

Dari perhitungan di atas, didapat ping yang tercatat untuk melakukan perjalanan dari simpul awal S menuju simpul akhir T dan sebaliknya adalah 0.1000693814377968 sekon, atau 100069,3814377968 ms. Sebagai perbandingan, jika kita mengambil lintasan S – U – X – Y – R – T dengan jarak tempuh 17000 km kita akan mendapatkan *latency* dan *ping* sebesar :

$$latency = 17000\ km / 299792\ km/s$$

$$latency = 0.0567059828147516\ s$$

$$ping = 2\ x\ \ 0.0567059828147516\ s$$

$$ping = 0.1134119656295032\ s$$

Dari perhitungan *latency* dan *ping* di atas, terlihat bahwa

lintasan S – U – X – Y – R – T memiliki *latency* dan *ping* yang jauh lebih besar dibandingkan dengan lintasan S – P – Q -R – T. Dengan demikian, semakin jauh jarak antara *server* dengan *client*, maka semakin besar *latency* dan *ping* yang dihasilkan. Semakin besar *latency* dan *ping* yang dihasilkan, maka semakin lambat pula respon yang diberikan *server*, sehingga *delay* pelayanan yang diberikan lebih besar atau biasa disebut dengan *lag*. Hal ini akan mempengaruhi kenyamanan dan kelancaran dalam mengakses data di internet. Sebagai contoh, jika *latency* atau *ping* pada komputer *client* besar, maka kenyamanan dalam mengakses suatu aplikasi, seperti *online game*, akan terasa lambat dan kurang nyaman. Dengan demikian, dibutuhkan jarak yang sependek-pendeknya untuk menjamin kenyamanan dan kelancaran dalam mengakses konten di internet.

## IV. KESIMPULAN

Lancar tidaknya koneksi suatu jaringan komputer dapat ditentukan dari jumlah *ping* dan *latency* yang didapatkan dari komputer *client*. *Ping* dan *latency* dapat dihitung dari jarak yang ditempuh dari *server* menuju *client* atau sebaliknya. Jarak tersebut dapat dihitung menggunakan algoritma Djikstra untuk mendapatkan jarak terdekat. Semakin jauh jarak yang ditempuh, semakin besar *ping* dan *latency* yang dihasilkan. Semakin besar *ping* dan *latency* yang dihasilkan, semakin lambat respon yang diberikan. Lambatnya respon yang diberikan dapat menghasilkan *lag* yang dapat mengurangi kelancaran dan kenyamanan dalam berinternet. Dengan demikian, dibutuhkan perhitungan untuk mencari jarak terpendek dengan harapan *ping* dan *latency* yang dihasilkan lebih kecil dan mempercepat proses akses internet. Selain itu, beberapa faktor lain seperti bahan kabel, kondisi geografis, dan beberapa faktor lain juga dapat mempengaruhi kecepatan akses internet yang dihasilkan.

## V. REFERENSI


[1] Tech Blog. "Theoretical vs real-world speed limit of Ping". Diakses dari http://royal.pingdom.com/2007/06/01/theoretical-vs-real-world-speed-limit-of-ping/ pada 9 Desember 2015 pukul 21.22.

[2] Lyberty. "KiloBytes vs. kilobits vs. Kibibytes (transfer speeds and capacities)". Diakses dari http://www.lyberty.com/encyc/articles/kb_kilobytes.htm pada 9 Desember 2015 pukul 21.23.

[3] WikiHow. "How to Ping an IP Address". Diakses dari http://www.wikihow.com/Ping-an-IP-Address pada 9 Desember 2015 pukul 21.30.

[4] Mario Miler, Damir Medak, Drazen Odobasic. *The Shortest Path Algorithm Performance Comparison In Graph And Relational Database on A Transportation Network.* University of Zagreb, 2013.

[5] E. W. Dijkstra: *A note on two problems in connexion with graphs*. In: *Numerische Mathematik*. 1 (1959), S. 269–271

[6] Thomas H. Cormen, Charles E. Leslerson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. Section 24.3: Dijkstra's algorithm, pp.595–601.

[7] Munir, Rinaldi. 2008. *Diktat Kuliah IF2091 Matematika Diskrit.* Institut Teknologi Bandung : Bandung.


## 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, 8 Desember 2015

ttd

I Dewa Putu Deny Krisna Amrita - 13514096