ALGORITMA *ROUTING* DI LINGKUNGAN JARINGAN GRID MENGGUNAKAN TEORI GRAF

Irfan Darmawan[(1)], Kuspriyanto[(2)], Yoga Priyana[(2)], Ian Yosep M.E[(2)]
Teknik Elektro, Universitas Siliwangi
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

ABSTRAKS
Grid Computing adalah infrastruktur komputasi yang menyediakan akses berskala besar terhadap sumber daya komputasi yang tersebar secara geografis namun saling terhubung menjadi satu kesatuan fasilitas. Sumber daya ini termasuk antara lain supercomputer, sistem penyimpanan, sumber-sumber data, dan instrument- instrument. Jaringan grid adalah suatu kumpulan resource (mesin, CPU, memori) yang saling berkomunikasi satu sama lain dengan menggunakan cara-cara (protokol) tertentu. Jaringan komputer dapat dimodelkan dengan menggunakan graf. Makalah ini khusus membahas pemodelan keterhubungan antar resource dan algoritma routing yang digunakan, pada suatu jaringan komputer, dengan memanfaatkan teori graf. Pada bagian awal dijabarkan secara ringkas beberapa definisi terkait dengan teori graf. Bagian selanjutnya adalah pembahasan beberapa algoritma routing pada suatu jaringan komputer. Algoritma routing yang dibahas adalah algoritma Breadth-First, algoritma Dijkstra dan algoritma Bellman-Ford. Untuk mendapatkan kelebihan dan kekurangan dari setiap algoritma routing dilakukan dengan cara menganalisis kompleksitas pada setiap algoritma tersebut.

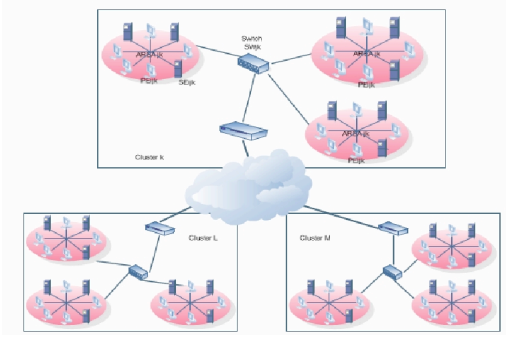
Kata kunci: graf, jaringan grid, router, algoritma routing, BreadFirst, Dijkstra, BellmanFor

1. PENDAHULUAN

Dipicu dengan kebutuhan untuk mengakses sumber daya yang bervariasi dan tersebar secara geografis serta suksesnya perancangan dan implementasi Cluster Computing dalam satu Node, maka para peneliti memperluas ide Cluster Computing. Sumber daya yang tadinya hanya difokuskan pada komputer saja, maka diperluas menjadi bermacam sumber daya. Kemudian masing masing Node akan dikaitkan menjadi gabungan beberapa Node, dan lokasi Node node tersebut bisa tersebar secara luas dalam suatu negara atau bahkan antara beberapa negara. Dari sini maka secara umum maka Grid computing banyak banyak didefinisikan oleh berbagai peneliti. Namun secara umum terdapat konsep yang sama.

Grid Computing adalah infrastruktur komputasi yang menyediakan akses berskala besar terhadap sumber daya komputasi yang tersebar secara secara geografis namun saling terhubung menjadi satu kesatuan fasilitas. Sumber daya ini termasuk antara lain supercomputer, system storage/penyimpanan, sumber sumber data, dan instrument instrument.

Suatu set G yang terdiri dari cluster (Ck) yang dihubungkan oleh gate (GTk), dimana k Є {0,…,G-1} , dimana setiap cluster terdiri dari sejumlah Network (Njk) yang dihubungkan oleh Switch (SWjk) dan setiap area terdiri dari beberapa Processing Element (PEijk) yang terhubung dalam suatu local domain seperti LAN (Local Area Network). Tampak pada gambar 1. merupakan topologi grid.



Gambar Hirarki topologi Grid

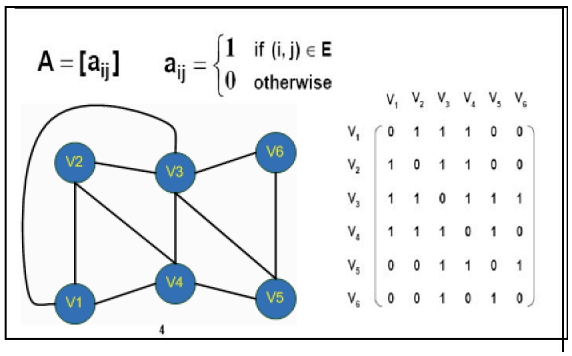
Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf sering digunakan untuk memodelkan jalur transportasi, penjadwalan, jaringan komputer , dan lain sebagainya.

Graph G = (V, E) adalah suatu graf dengan V adalah himpunan tidak kosong dari simpulsimpul (*vertices*), yaitu terdiri dari n buah simpul {v1, v1, v3, ..., vn} dan E = {e1, e2, ..., en} adalah himpunan sisi (*edges*) yang menghubungkan sepasang simpul. Jika pasangan simpul (i, j) saling terhubung, maka kedua simpul tersebut dikatakan bertetangga. Atau dengan kata lain jika e merupakan sisi yang menghubungkan simpul i dan j, e = (i, j), maka sisi e disebut bersisian dengan simpul i dan j. Kardinalitas dari himpunan simpul dilambangkan dengan |V|.

Graf G = (V, E) dapat direpresentasikan dengan | V| x |V| matriks ketetanggaan. Gambar di bawah ini merupakan contoh representasi matriks

 ISSN: 1907-5022

ketetanggaan pada suatu graf tidak berarah.

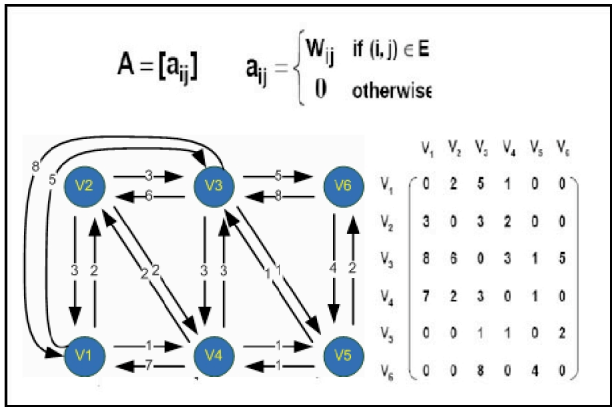


Gambar 1 Matriks Ketetanggaan Pada Suatu Graf

Beberapa definisi yang sering dijumpai dalam teori graf adalah sebagai berikut:

1. Jalur (path) antara simpul i dan j adalah sebuah urutan simpul dan sisi yang dimulai dari simpul i dan berakhir di simpul j, dengan setiap sisi bersisian dengan simpul sebelum dan sesudahnya. Sebuah jalur disebut sederhana jika setiap sisi yang dilalui pada jalur tersebut hanya sekali saja.
2. Jarak antara simpul i dan j adalah jumlah sisi minimum yang berada pada jalur dari simpul i ke simpul j.
3. Sebuah jalur sederhana disebut sebagai kalang (cycle atau loop) jika simpul awal juga merupakan simpul akhir.
4. Sebuah graf disebut terhubung jika terdapat jalur pada setiap pasang simpul i dan j.
5. Sebuah graf disebut berarah jika sisi pada graf tersebut memiliki arah. Dengan demikian
6. simpul (i, j) pada sisi e tidak secara langsung mengakibatkan bahwa simpul (j, i) juga ada pada e. Pada kondisi ini, sisi pada graf sering disebut dengan busur.
7. Graf disebut memiliki bobot (*weight*) jika pada setiap sisi (atau busur) diberi label dengan suatu bilangan.

Berbeda dengan graf tidak berarah pada gambar 1, representasi matriks untuk graf berarah ditunjukkan pada gambar di bawah ini. Pada gambar dapat dilihat bahwa elemen pada setiap matriks menggambarkan bobot (*weight*) pada setiap pasang simpul.



Gambar 2 Representasi Matriks Untuk Graf Berarah

1.1 Pohon (*Tree*)

Graf T merupakan sebuah pohon jika dan hanya j ika ada satu jalur sederhana pada setiap pasang simpul (i, j). Jika N = |V|, maka ada N - 1 sisi (busur) dan semuanya terhubung, tanpa ada kalang (*loop*). Setiap simpul dapat menjadi akar pada pohon tersebut. Sebuah pohon dapat direpresentasikan dengan menyusun simpulsimpul dalam suatu urutan tertentu yang dimulai dari akar. Beberapa definisi pada pohon adalah sebagai berik ut:

1. Setiap simpul (kecuali akar) hanya memiliki satu simpul orang tua.
2. Setiap simpul memiliki 0 atau lebih anak
3. Sebuah sisi dengan 0 anak adalah merupakan da un.

1.2 Pohon Merentang (*Spanning Tree*)

Subgraf (*subgraph*) dari sebuah graf G = (V, E) didapatkan dengan cara:

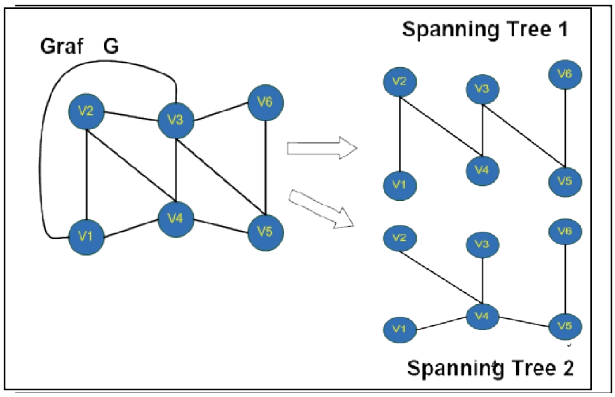
1. G' = (V', E'), dimana V' dan E' adalah himpunan bagian dari V dan E.
2. Untuk setiap sisi (busur) e pada E', simpul i dan j ada pada V'.

Sebuah subgraf T = (V', E') dari G = (V, E) adalah pohon merentang dari G jika:

1. T adalah sebuah pohon.
2. V' = V.

Gambar di bawah ini adalah dua buah pohon merentang yang didapatkan dari Graf

 ISSN: 1907-5022



Gambar 3 Pohon Merentang Dari Graf G

1.3 Teori Graf dan Algoritma *Routing*

Pada jaringan komputer, aliran setiap paket dapat dimodelkan dengan menggunakan graf yang memiliki bobot. Simpul pada graf merepresentasikan *router*, sedangkan busur pada graf merepresentasikan *subnet*. *Routing* adalah proses untuk meneruskan paket dari suatu simpul ke simpul yang lainnya dengan berdasarkan asas jalur terpendek. Tujuannya adalah agar pengiriman paket tersebut dapat dilakukan secara efektif dan efisien. Algoritma *routing* untuk mendapatkan jalur terpendek diantaranya adalah algoritma *BreadFirst*, Dijkstra, dan BellmanFord.

2 ALGORITMA ROUTING

2.1 BreadFirst

Breadth-first search adalah algoritma yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut terlebih dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpulsimpul yang tadi dikunjungi , demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pad aras d+1.

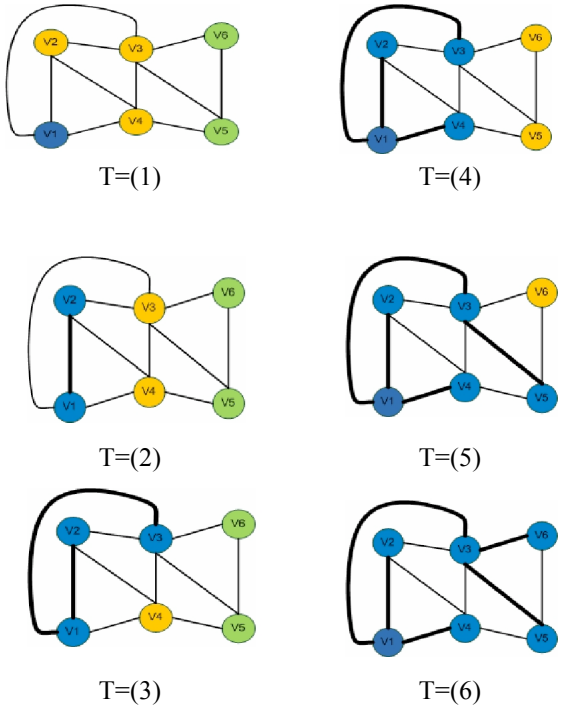
Algoritma ini memerluka-n sebuah antrian q untuk menyimpan simpul yang telah dikunjungi. Simpulsimpul ini diperlukan sebagai acuan untuk mengunjungi simpul-simpul yang bertetanggaan dengannya. Tiap simpul yang telah dikunjungu masuk ke dalam antrian hanya satu kali.

Algoritma ini juga membutuhkan table Boolean untuk menyimpan simpul yang te lah dikunjungi sehingga tidak ada simpul yang dikunjungi lebih dari satu kali.

Secara umum algoritma BreadFirst akan melakukan langkahlangkah sebagai berikut:

1. Menentukan simpul yang berperan sebagai akar (misalnya simpul x).
2. Melakukan penjelajahan terhadap simpul yang bertetangga dengan simpul x (level 1).
3. Untuk setiap simpul yang berada pada level 1, dilakukan penjelajahan terhadap semua simpul yang bertetangga dengan semua simpul pada level 1 yang belum dijelajahi pada langkah sebelumnya (level 2).
4. Untuk setiap simpul yang dijelajahi, diperiksa apakah merupakan simpul tujuan atau tidak. Jika merupakan simpul tujuan maka penjelajahan akan berhenti dengan solusi merupakan simpulsimpul yang dijelajahi dari awal sampai kepada simpul akhir tujuan.
5. Jika bukan merupakan simpul tujuan makan perulangan akan berlaku s elama belum semua simpul dijelajahi.

Gambar di bawah ini adalah contoh penerapan algoritma *BreadFirst* pada sebuah pohon merentang yang berasal dari suatu graf.



Gambar 4 Algoritma *BreadFirst*

Kompleksitas algoritma BreadFirst adalah $O(|V|^3)$.

2.2 Dijkstra

Dijkstra adalah sebuah algoritma dalam memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah (directed graph) dengan bobot-bobot sisi (edge weights) yang bernilai tak-negatif.

Ide dasar dari algoritma Dijkstra adalah memeriksa simpul dengan bobot terkecil dan memasukkannya ke dalam himpunan solusi. Pada setiap iterasi himpunan solusi ini akan berubah jika terdapat sisi dengan bobot lebih kecil dari sisi pada himpunan solusi. Secara umum langkahlangkah pada algoritma Dijkstra adalah sebagai berikut:

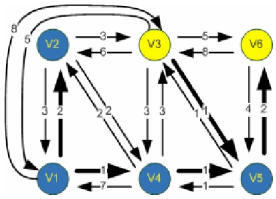
 ISSN: 1907-5022

1. Pada langkah k: cari sisi k yang dapat dijangkau dari s dengan bobot minimum, dilambangkan dengan Tk.
2. Pada langkah k+1: cari simpul v dengan jarak minimum dari s dan dapat dijangkau dengan simpulsimpul yang ada pada Tk (kecuali v sendiri).
3. Tk+1 = Tk U {v}
4. Algoritma berhenti ketika semua simpul sudah dikunjungi.

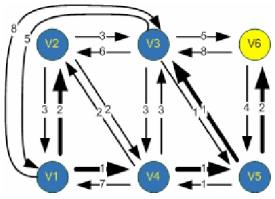
```
procedure dijkstra (w,a,z,L)
  L(a) := 0
  for semua verteks x ≠ a do
   L(x) := ∞
   T := himpunan semua verteks
   // T adalah himpunan verteks yang panjang
       terpendeknya dari a belum ditemukan
  while z ϵ T do
  begin
       pilih v ϵ T dengan minimum L(v)
       T:=T-{v}
      for setiap x ϵ T di samping v do
       L(x):=min{L(x), L(v)+w(v,x)}
  end
end dijkstra
```

Gambar 5 Algoritma Dijkstra

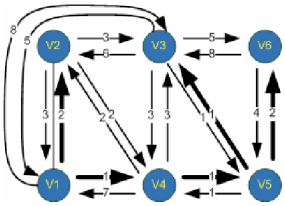
Untuk lebih jelasnya, gambar di bawah ini merupakan langkahlangkah yang dilakukan dengan menggunakan algoritma Dijkstra.



T_4={1,2,4,5}



T_5={1,2,3,4,5}



T_6={1,2,3,4,5,6}

Gambar 6 Langkah -langkah Pada Algoritma Dijkstra

Kompleksitas algoritma Dijkstra adalah $O(|V|^2)$. Dengan implementasi yang efisien dapat diperoleh kompleksitas O(|V| log|V|).

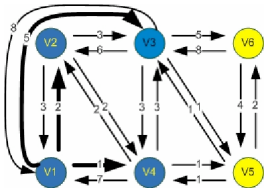
2.3 BellmanFord

Algoritma Bellman-Ford menghitung jarak terpendek (dari satu sumber) pada sebuah digraf berbobot. Maksudnya dari satu sumber ialah bahwa ia menghitung semua jarak terpendek yang berawal dari satu titik node.

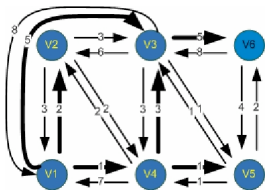
Misalnya simpul sumber adalah s, dan akan dicari jalur terpendek dari s ke semua simpul yang lain pada graf G. Langkahlangkah pada algoritma BellmanFord adalah sebagai berikut:

1. Temukan jalur terpendek antara s dan simpul lainnya, sehingga jalur ini adalah paling banyak memiliki 1 lompatan (*hop*).
2. Cari jalur terpendek antara s dan simpul lainnya dengan memiliki paling banyak 2 lompatan.
3. Lakukan iterasi sampai jalur terpendek memiliki jumlah lompatan paling banyak berjumlah diameter dari graf. Diameter graf adalah jarak maksimum antara pasangan simpul pada graf, diukur dengan lompatan (*hop*).

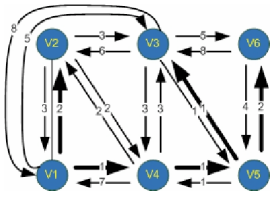
Gambar di bawah ini merupakan Langkah-langkah Pada Algoritma BellmanFord



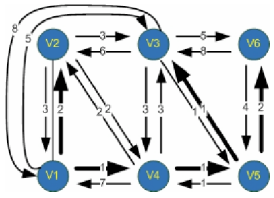
h=1



h=2



h=3



h=4

Gambar 7 Langkah -langkah Pada Bellman-Ford

Kompleksitas yang diperoleh adalah O(|V|.|E|), jadi untuk kasus terburuk adalah ketika $|E|=|V|^2$, adalah $O(|V|^3)$.

```
function BellmanFord(list semuatitik, list
semuasisi, titik dari)
for each titik v in semuatitik:
    if v is dari then v.jarak = 0
    else v.jarak := tak-hingga
         v.sebelum := null
   // Perulangan relaksasi sisi
for i from 1 to size(semuatitik):
    for each sisi uv in
        semuasisi: u := uv.dari
        v := uv.ke //uv adalah sisi dari u ke v
        if v.jarak > u.jarak + uv.bobot
          v.jarak := u.jarak + uv.bobot
          v.sebelum :=u
    // Cari sirkuit berbobot(jarak)
    negatif for each sisi uv in semuasisi:
          u := uv.dari
          v := uv.ke
          if v.jarak > u.jarak + uv.bobot
          error "Graph mengandung
          siklus berbobot total negatif"
```

Gambar 6 Algoritma Bellman-Ford

4. ALGORITMA DIJKSTRA VERSUS BELLMAN FORD

Walaupun kedua algoritma (Dijkstra dan BellmanFord) samasama berusaha menemukan jalur terpendek, namun kedua algoritma tersebut memiliki beberapa perbedaan. Salah satu perbedaan adalah dalam hal kompleksitas. Algoritma BellmanFord memiliki kompleksitas yang lebih besar dibandingkan dengan Dijkstra.

Perbedaan lainnya adalah pada algoritma Dijkstra simpul asal membutuhkan pengetahuan tentang topologi graf, yaitu informasi semua busur dan bobotnya. Oleh karena itu dibutuhkan pertukaran informasi dengan semua simpul. Sedangkan algoritma Bellman-Ford membutuhkan pengetahuan mengenai bobot dari sisi yang bertetangga dengan suatu simpul dan nilai dari jalur terpendek yang dimulai dari simpul tetangganya tersebut. Dalam hal ini hanya diperlukan komunikasi antara simpul yang bertetangga saja (implementasi terdistribusi). Walaupun algoritma BellmanFord hanya membutuhkan distribusi informasi antara simpul yang bertetangga saja, namun implementasi terdistribusi pada algoritma BellmanFord dapat menyebabkan masalah yang disebut dengan *bad news phenomenon*, dimana iterasi yang diperlukan bisa sedemikian tinggi sehingga memperlambat distribusi informasi dari satu simpul ke simpul yang lainnya.

5. KESIMPULAN

Berdasarkan pembahasan di atas, kesimpulan yang didapatkan dari makalah Penerapan Teori Graf Pada Algoritma *Routing* jaringan grid adalah sebagai berikut:

1. Keterhubungan antar *node* pada suatu jaringan komputer dapat dimodelkan dengan menggunakan graf.
2. Graf yang dihasilkan dari pemodelan tersebut dapat diproses lebih lanjut dengan menggunakan kaidah-kaidah yang berlaku pada graf.
3. Beberapa algoritma routing adalah algoritma BreadFirst, Dijkstra, dan BellmanFord.
4. Setiap algoritma routing tersebut memiliki kompleksitas yang berbedabeda. BreadFirst memiliki kompleksitas O(|V|3), Dijkstra memiliki kompleksitas O(|V|2) dan Bellman-Ford adalah O(|V|.|E|).

REFERENSI

[1] Becchetti, Lucas, *Computer Networks II: Graph theory and routing algorithms*, Universita degli Studi di Roma, 2008.
[2] Tanenbaum, Andrew S, Computer Networks, Prentice Hall PTR, 2002.
[3] Munir, Rinaldi, Matematika Diskrit, Edisi Ketiga, Penerbit Informatika: Bandung, 2005.
[4] Wilf, Herbert S. 2006. Algorithm and Complexity. University of Pennsylvania: Philadelphia.
[5] Foster and C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998.
[6] E. Deelman A.Chervenak and al. High performance remote access to climate simulation data: a challenge problem for data grid technologies. In Proceeding. of 22th parallel computing, volume 29(10), pages 13–35, 1997.
[7] Jose Duato, “Interconnection Networks: An Engineering Approach”, Morgan Kaufmann Publisher, 2003.