Aplikasi Teori Graf dalam Algoritma Pengalihan Arus Lalu Lintas

Muhammad Yafi 13512014
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512014@std.stei.itb.ac.id

***Abstrak*—Makalah ini membahas tentang penggunaan salah satu materi Matematika Diskrit, yaitu Graf, untuk merancang algoritma pengalihan arus lalu lintas. Dalam kehidupan nyata, sering terjadi masalah kemacetan lalu lintas. Penyebabnya antara lain adalah jumlah kendaraan yang lewat dalam suatu jalan melebihi kapasitas jalan tersebut. Salah satu alternatif mengatasi kemacetan lalu lintas tersebut adalah pengalihan arus. Untuk itu diperlukan suatu algoritma yang efektif sehingga arus lalu lintas menjadi lancar di setiap ruas jalan.**

***Index Terms*—Algoritma, Asiklik, Graf, Jalan, Kapasitas, Lalu Lintas**

I. PENDAHULUAN

Kesejahteraan masyarakat Indonesia dan gaya hidup menyebabkan jumlah kendaraan pribadi yang meningkat di jalan raya. Kapasitas ruas jalan yang tersedia tidak mampu menampung banyaknya kendaraan yang lewat. Hal ini berdampak pada timbulnya masalah lalu lintas yaitu kemacetan. Kemacetan terjadi pada saat waktu sibuk, misalnya pada pagi dan sore hari saat orang-orang hendak pergi maupun pulang kerja. Selain itu, kemacetan juga terjadi pada saat libur panjang. Orang-orang berpergian untuk berwisata atau ke rumah saudara dalam waktu yang sama. Kemacetan tersebut memiliki jangka waktu yang pendek, artinya hanya terjadi pada rentang waktu tertentu saja.

Ketika kemacetan jangka pendek terjadi, harus ada solusi yang dapat melancarkan arus lalu lintas kembali dengan cepat. Salah satu alternatif solusi masalah tersebut adalah dengan cara pengalihan arus lalu lintas. Pengalihan tersebut bertujuan untuk mengurangi jumlah kendaraan yang lewat dalam suatu ruas jalan. Pengguna kendaraan dipaksa untuk mengambil rute lain untuk mencapai tujuannya.

Pengalihan arus lalu lintas harus dilakukan secara cermat, sehingga pengalihan arus di suatu tempat tidak berakibat kemacetan lalu lintas di tempat lain. Oleh karena itu, pihak yang berwenang dalam mengatur lalu lintas memerlukan suatu algoritma yang efektif untuk mengatur pengalihan arus.

Pada makalah ini, penulis merancang suatu algoritma yang mengambil dua buah tempat, yaitu tempat asal dan tempat tujuan. Dari tempat asal ke tempat tujuan, terdapat satu atau lebih alternatif rute dengan masing-masing rute menggunakan ruas jalan dengan arah tertentu. Setiap ruas jalan memiliki kapasitas maksimum untuk menampung jumlah kendaraan yang lewat pada interval waktu tertentu. Diketahui jumlah kendaraan yang bergerak dari simpul asal ke simpul tujuan dan penyebaran kendaraan tersebut pada masing-masing ruas jalan. Jika jumlah kendaraan dalam suatu ruas jalan melebihi kapasitas ruas jalan tersebut, dikatakan bahwa jalan tersebut macet. Jika kapasitas ruas jalan melebihi jumlah kendaraan yang lewat dalam ruas jalan tersebut, dikatakan bahwa jalan tersebut lancar.

Tujuan dari algoritma ini mendistribusikan kembali kendaraan yang bergerak dari simpul asal menuju simpul tujuan sehingga setiap ruas jalan dapat menampung jumlah kendaraan sesuai kapasitasnya.

II. DASAR TEORI

2.1 Definisi Graf

Graf G didefinisikan sebagai pasangan himpunan (V,E) dengan

V = himpunan tak-kosong dari simpul
= { v_1 , v_2 , v_3, ... , v_n }

E = himpunan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul
= { e_1 , e_2 , e_3, ... , e_n }

Atau dapat ditulis singkat notasi G = (V,E).

Jika sebuah *edge e* menghubungkan simpul v_i dan v_j maka *e* dapat ditulis sebagai $e = (v_i , v_j)$.

2.2 Jenis-Jenis Graf

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

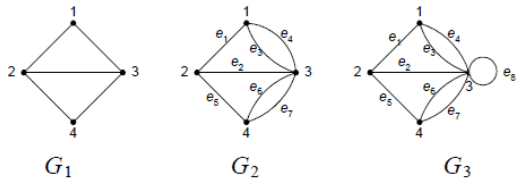
1. Graf sederhana
   Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.

2. Graf tak-sederhana

Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana. Ada dua macam graf tak-sederhana, yaitu graf ganda dan graf semu.

Graf ganda adalah graf yang mengandung sisi ganda. Sebuah graf memiliki sisi ganda jika ada 2 buah simpul yang dihubungkan lebih dari satu sisi.

Graf semu adalah graf yang memiliki sisi gelang (*loop)*. Sisi gelang adalah sisi yang menghubungkan sebuah simpul dengan simpul itu sendir. Untuk lebih jelasnya, perhatikan Gambar 1.

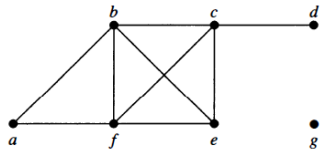


Gambar 1-(a) graf sederhana, (b) graf ganda, dan (c) graf semu.[5]

Sisi graf dapat memiliki orientasi arah. Berdasarkan arah dari sisi, graf dibedakan menjadi 2 jenis :

1. Graf tak-berarah

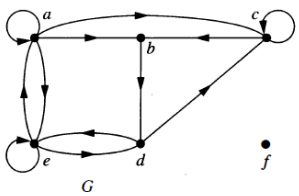
Graf yang sisinya tidak memiliki orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul pada sisi tidak diperhatikan. Sebuah sisi e = (u, v) sama dengan e = (v, u)



Gambar 2-Graf tak-berarah[6]

2. Graf berarah

Graf yang setiap sisinya memiliki orientasi arah disebuh graf berarah. Pada graf berarah, sebuah sisi dikenal juga sebagai busur (*arc*). Pada graf berarah, (u, v) dan (v, u) menyatakan dua buah sisi yang berbeda. Pada sebuah sisi (u, v), simpul u menyatakan simpul asal (*initial vertex)* dan simpul v menyatakan simpul terminal (*terminal vertex)*.



Gambar 3 - Graf berarah[6]

Sisi pada graf dapat memiliki bobot atau tidak. Berdasarkan bobot pada sisinya, graf dapat digolongkan menjadi dua :

1. Graf berbobot (*weighted graph)*

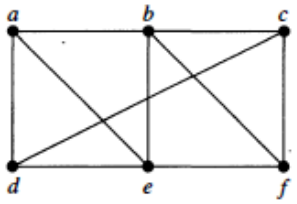
Graf berbobot adalah graf yang setiap sisinya memiliki bobot. Bobot pada sisi graf dapat merepresentasikan kapasitas, biaya, atau keuntungan.

2. Graf tak-berbobot (*unweighted graph)*

Graf tak-berbobot adalah graf yang setiap sisinya tidak memiliki bobot.

2.3 Lintasan dan Sirkuit

Sebuah lintasan (*path)* didefinisikan sebagai urutan dari n buah sisi $e_1, \ldots, e_n$ sedemikian hingga e_1 merupakan (v_0, v_1), e_2 merupakan (v_1, v_2), …, e_n merupakan (v_{n-1}, v_n). Lintasan juga dapat didefinisikan sebagai urutan simpul yang dikunjungi $v_1, v_2, v_3, v_4, \ldots, v_n$, Sebuah lintasan disebut sirkuit (*cycle/circuit)* jika $v_0 = v_n$ atau dengan kata lain simpul asal dan simpul tujuan sama. Sebagai contoh, perhatikan Gambar 4. Rute *a, b, c, f, e, d* merupakan lintasan dari *a* ke *d*. Rute *a, b, f, c, d, a* merupakan sirkut karena memiliki simpul asal dan tujuan yang sama, yaitu *a*.



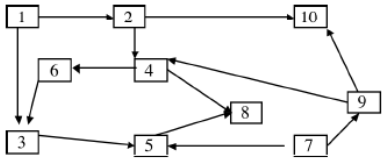
Gambar 4-Graf *cyclic*[6]

Sebuah graf yang tidak memiliki lintasan siklik disebut graf asiklik (*acyclic graph)*. Graf pada Gambar 4 merupakan graf siklik karena memiliki lintasan siklik.

2.4 Algoritma Topological Sort

Terdapat sebuah algoritma yang diperlukan untuk studi kasus maslah ini yaitu algoritma *topological sort* (biasa disingkat toposort). Definisi persoalan topological sort adalah *diberikan urutan parsial dari elemen suatu himpunan, tentukan barisan elemen sehingga elemen tersebut memiliki keterurutan lanjar.*

Syarat algoritma topological sort adalah graf yang diselesaikan merupakan graf berarah asiklik (*directed acyclic graph)*. Sebagai contoh, pada gambar di bawah, urutan elemen yang dikehendaki adalah 7, 9, 1, 2, 4, 8, 6, 3, 5, 10. Bisa juga dengan 1, 2, 7, 9, 4, 6, 3, 5, 8, 10



Gambar 5- Directed Acyclic Graph[4]

Algoritma topological sort adalah sebagai berikut

1. Mula-mula S merupakan himpunan seluruh simpul yang ada di graf.
2. Cari sebuah simpul X di mana simpul tersebut tidak memiliki predecessor. Minimal ada satu

simpul seperti ini. Jika tidak, maka akan terjadi *cycle* dan grafnya bukan asiklik.

3. Hapus X dari S dan keterhubungan dari X ke item lainnya. Insert X sebagai elemen paling belakang L.
4. Jika S tidak kosong, kembali ke langkah 2.
5. Cetak L.

Penerapan algoritma topological sort pada gambar adalah sebagai berikut

No	Simpul	Deleted edge	L
1	1	{(1,2) , (1,3)}	{1}
2	2	{(2,4),(2,10)}	{1,2}
3	7	{(7,5),(7,9)}	{1,2,7}
4	9	{(9,4),(9,10)}	{1,2,7,9}
5	4	{(4,6),(4,8)}	{1,2,7,9,4}
6	6	{(6,3)}	{1,2,7,9,4,6}
7	3	{(3,5)}	{1,2,7,9,4,6,3}
8	5	{(5,8)}	{1,2,7,9,4,6,3,5}
9	8	{}	{1,2,7,9,4,6,3,5,8}
10	10	{}	{1,2,7,9,4,6,3,5,8,10}

III. Algoritma Pengalihan Arus Lalu Lintas

3.1. Deskripsi Permasalahan

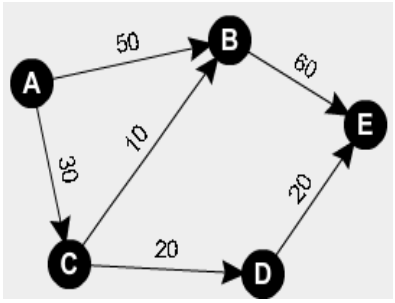
Untuk mempermudah pendeskripsian masalah, penulis mengambil kasus kemacetan arus lalu lintas pada musim lebaran. [3]Pada musim lebaran, diperkirakan akan terjadi lonjakan arus lalu lintas di beberapa ruas tol yang menuju ke arah Jawa Barat dan Jawa Tengah dari arah Jakarta. Untuk memperlancar para pemudik menuju kampung halamannya, pihak kepolisian bersama pihak Jasa Marga melakukan rekayasa lalu lintas di berberapa titik yang diprediksikan akan mengalami peningkatan lalu lintas yang sangat signifikan. Bila kepadatan kendaraan terjadi, pengguna jalan tol dari Jakarta yang hendak menuju arah Pantura diarahkan untuk belok ke arah Tol Cipularang dan keluar di Gerbang Tol Sadang. Untuk selanjutnya melalui Subang, Cikamurang, Kadipaten dan Cirebon. Dan apabila jalur Pantura macet total, Kepolisian akan mengatur kendaraan yang akan menuju Jawa Tengah atau Jawa Timur untuk melalui Lintas Selatan (masuk Cipularang, keluar di Gerbang Tol Cileunyi, Sumedang, Kadipaten dan Cirebon, bisa juga melalui Nagrek, Malangbong, Tasikmalaya, dan seterusnya).

Dari deskripsi di atas, diambil hal-hal berikut :

1. Sumber kemacetan berasal dari Jakarta, yaitu kendaraan yang menuju Jawa Barat dan Jawa Tengah.
2. Dalam kasus ini kita hanya akan melihat Jakarta sebagai kota awal dan Jawa Tengah sebagai tujuan.
3. Terdapat berberapa alternatif lintasan Jakarta-Jawa Tengah yaitu melalui Subang – Cirebon, Cipularang – Cileunyi – Cirebon, dan Cipularang – Tasikmalaya.

Gambar 6 - Gambar Peta Jakarta-Jawa Tengah

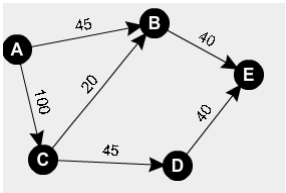
Peta tersebut dapat direpresentasikan sebagai graf berbobot berarah asiklik dengan bobot dari sebuah graf merupakan banyak kendaraan yang lewat dalam interval waktu tertentu sebagai berikut (misalkan saja dihitung banyak kendaraan yang lewat dalam 1 menit).



Tambahan
Keterangan
A : Jakarta
B : Cirebon
C : Bandung
D : Tasikmalaya
E : Jawa Tengah
A-B : Jalur Subang
B-E : Jalur Pantura
C-B : Tol Cileunyi
A-C : Cipularang

Gambar 7 - Representasi Graf Berarah dari Peta Jakarta-Jawa Tengah dengan bobot berupa banyak kendaraan yang lewat.

Dalam 1 menit, terdapat 80 kendaran dari arah Jakarta yang ingin menuju Jawa Tengah. Jika diperhatikan, kebanyakan kendaraan melewati Jalur Pantura (60 kendaraan) dibanding melewati Tasikmalaya (20 kendaraan). Misalkan terdapat graf yang merepresentasikan kapasitas masing-masing jalan dalam satu menit.



Gambar 8 - Representasi Graf Berarah dari Peta Jakarta-Jawa Tengah dengan bobot berupa kapasitas sisi

Dari 2 buah gambar tersebut, terjadi kemacetan di Jalur Pantura (kapasitas 40 dibanding jumlah yang lewat 60) dan Jalur Subang (kapasitas 45 dibanding jumlah yang lewat yaitu 50). Namun, terdapat ruas jalan yang lancar jika pemudik ingin menuju Jawa Tengah melewati Tasikmalaya.

Algoritma yang akan dijelaskan dalam makalah ini akan mengatur ulang jumlah kendaraan yang lewat di setiap ruas jalan sehingga tiap ruas jalan dapat menampung kendaraan yang lewat sesuai kapasitasnya.

3.2. Bentuk Umum Permasalahan

Sebelum melakukan analisis, terlebih dahulu penulis menyebutkan batasan-batasan yang harus dipenuhi

1. Menentukan dua buah simpul, yaitu simpul asal/*source* dan simpul tujuan/*finish*. Simpul asal tidak boleh memiliki sisi yang arahnya masuk ke simpul. Simpul tujuan tidak boleh memiliki sisi yang arahnya keluar dari simpul.
2. Banyak kendaraan yang keluar dari simpul harus sama dengan banyak kendaraan yang masuk dari suatu simpul, kecuali pada simpul asal dan tujuan. Dengan kata lain, diasumsikan penambahan kendaraan pada simpul selain pada simpul asal adalah nol atau sangat sedikit/tidak signifikan terhadap jumlah kendaraan dari simpul asal.
3. Graf yang diuji merupakan graf berbobot berarah sederhana. Bobot pada graf ada dua macam yaitu bobot untuk merepresentasikan banyak kendaraan yang lewat dan bobot untuk merepresentasikan kapasitas sisi. Batasan ini diberikan karena pada kejadian nyata, setiap jalan memiliki arah dan kapasitas masing-masing. Sisi ganda dapat diganti menjadi satu buah sisi dengan kapasitas sama dengan jumlah kapasitas sisi yang digabung.
4. Graf yang diuji tidak memiliki gelang. Hal ini dikarenakan saat kendaraan ingin mencapai simpul tujuan, sisi gelang tidak akan dilewati karena hal ini menyebabkan posisi kendaraan tidak maju (dari suatu simpul kembali lagi ke simpul yang sama)
5. Graf yang diuji merpakan graf asiklik. Hal ini dikarenakan saat mengambil lintasan, setiap simpul hanya akan dilewati sekali dan tidak ada lintasan yang merupakan permutasi dari lintasan alternatif lain.
6. Kapasitas setiap jalan ditentukan sendiri oleh pembaca. Penentuan kapasitas jalan dapat ditentukan dari lebar jalan, struktur jalan, dan kondisi medan. Kapasitas yang ditentukan pada setiap jalan harus memungkinkan pendistribusian ulang sedemikian hingga setiap jalan memiliki banyak kendaraan tidak melebihi kapasitasnya. Untuk itu, penentuan kapasitas jalan tidak boleh *underestimate* atau terlalu kecil.

Jika permasalahan yang terjadi tidak sesuai dengan batasan di atas, maka permasalahan tersebut harus diubah terlebih dahulu sedemikian hingga batasan di atas terpenuhi. Masalah bagaimana cara mengubah permasalahan agar memenuhi batasan diserahkan kepada pembaca.

Bentuk umum permasalahan pengalihan arus lalu lintas dalam makalah ini adalah sebagai berikut

1. Diberikan graf berbobot berarah asiklik yang merepresentasikan kapasitas tiap jalan pada interval waktu tertentu dan jumlah kendaraan yang lewat pada interval waktu yang sama
2. Terdapat simpul asal dan simpul tujuan yang ingin ditinjau.
3. Algoritma akan mengatur ulang pendistribusian lalu lintas pada graf sedemikian hingga kapasitas setiap jalan tidak kurang dari jumlah kendaraan yang lewat.

3.3. Analisis Permasalahan dan Penerapan Algoritma

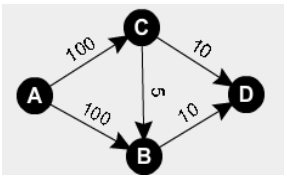
Untuk merancang algoritma pengalihan arus lalu lintas, ada analisis pendahuluan yang harus dilakukan terlebih dahulu.

Misalkan pada sebuah simpul s terdapat himpunan *out-edge* (sisi dari simpul s ke simpul lain) $\{out_1, out_2, \ldots out_n\}$ dan himpunan *in-edge* (sisi dari simpul lain ke simpul s $\{in_1, in_2, \ldots in_m\}$. Jika kapasitas dari $out_1, out_2, \ldots out_n$ masing-masing adalah $c_1, c_2, \ldots, c_n$ maka didefinisikan kapasitas dari simpul s adalah total kapasitas simpul keluar yaitu $ct = c_1 + c_2 + \ldots + c_n$.

Jika total kendaran yang masuk dari $in_1, in_2, \ldots in_m$ melebihi ct maka kemacetan tidak dapat dihindari sehingga kendaraan yang masuk dari $in_1, in_2, \ldots in_m$ harus dialihkan sehingga total kendaraan yang masuk dari $in_1, in_2, \ldots in_m$ tidak melebihi ct.

Jika total kendaraan yang masuk dari $in_1, in_2, \ldots in_m$ tidak melebihi ct, dan terdapat sebuah sisi out_i di mana kapasitas out_i kurang dari jumlah kendaraan yang lewat, maka kita dapat "mengambil" sebagian kendaraan dan menaruhnya ke sisi *out* yang jumlah kendaraan lewatnya lebih sedikit dibanding kapasitasnya.

Selain itu, terdapat kapasitas sebenarnya dari suatu sisi jalan, didefinisikan sebagai kapasitas efektif. Kapasitas tiap jalan harus diubah dari kapasitas maksimal (kapasitas awal) menjadi kapasitas efektif (kapasitas maksimal sebenarnya yang akan dipakai). Untuk lebih jelas membedakan antara kapasitas maksimal dengan kapasitas efektif, perhatikan gambar berikut

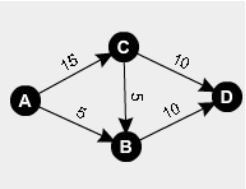


Gambar 9 - Graf kapasitas maksimal (awal)

Jika kita perhatikan, jumlah kendaraan yang lewat pada sisi AC tidak akan lebih dari 15. Hal ini dikarenakan untuk mencapai D melewati C, hanya ada 2 cara yaitu dengan C-B-D (max. 5) atau C-D (max 10). Untuk sisi AB tidak akan lebih dari 10 karena untuk ke D hanya ada cara B – D dengan kapasitas 10.

Misalkan kita ubah kapasitas graf tersebut sehingga 15 kendaraan melewati AC, 10 melewati CD, 5 melewati CB dan 10 melewati BD. Pada akhirnya, kapasitas efektif pada sisi AB hanya 5. Hal ini dikarenanan pada simpul B, jumlah kendaraan keluar adalah 10 dan 5 kendaraan masuk melewati CB sehingga hanya tersisa 10 – 5 kapasitas pada sisi AB.

Kapasitas efektif graf tersebut menjadi gambar berikut.



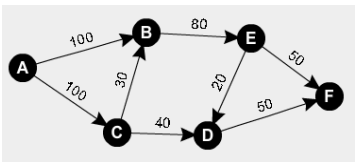
Gambar 10 - Graf kapasitas efektif

15 adalah kapasitas efektif dari sisi AC, dan 5 adalah kapasitas efektif dari AB. Untuk menentukan kapasitas efektif masing-masing sisi, kita menyimpan informasi tambahan di simpul berupa kapasitas simpul, yaitu banyak kendaraan yang boleh masuk ke dalam simpul tersebut.

Untuk menentukan kapasitas efektif suatu graf digunakan algoritma sebagai berikut :

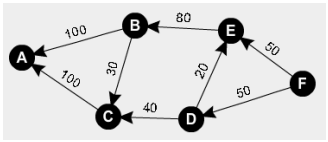
```
SetCapEff(G : Graf)
  cap : array of integer {kapasitas vertex}
  visited : array of simpul

  for each e ∈ G.E
          e (u,v) ← e (v,u)
  endfor
  for each v ∈ G.V
    cap[v] ← 0
  endfor
  cap[finish] ← ∞

  {topological sort}
  while (NbElmt(visited) < numberOfVertex(G.V)) do
    u ← vertex pada G.V sehingga u ∉ visited dan setiap
predecessor dari u ada di visited
    {maksimum kapasitas dari u adalah total kapasitas sisi
masuk}
    for each e(v,u) ∈ G.E
      cap[u] ← cap[u] + cost(e)
    endfor

    temp ← cap[u]
    {distribusikan kembali secara sekuensial}
    for each e(u,p) ∈ G.E
      cost(e) ← min(cost(e),temp)
      temp ← max(0,temp - cost(e))
    endfor
    insert u into visited
  endwhile
  for each e ∈ G.E
        e(u,v) ← e(v,u)
```

Ilustrasi penggunaan algoritma topological sort sebagai berikut



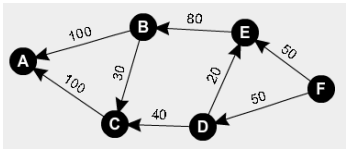
Kapasitas Simpul
A : 0 D : 0
B : 0 E : 0
C : 0 F : ∞
Visited = {}

Pertama, ubah arah sisinya terlebih dahulu



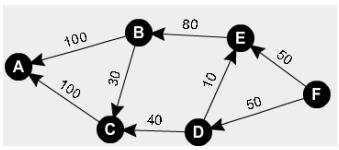
Kapasitas Simpul
A : 0 D : 0
B : 0 E : 0
C : 0 F : ∞
Visited = {}

Ambil simpul F.



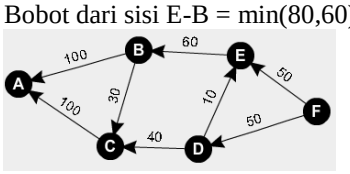
Kapasitas Simpul
A : 0 D : 0
B : 0 E : 0
C : 0 F : ∞
Visited = {F}

Ambil simpul D.
Kapasitas D = 50.
Bobot dari sisi D-C = min(50,40) = 40.
Bobot dari sisi D-E = min(20,10) = 10



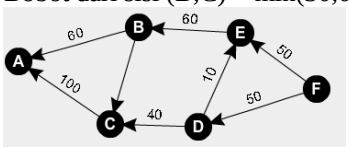
Kapasitas Simpul
A : 0 D : 50
B : 0 E : 0
C : 0 F : ∞
Visited = {F,D}

Ambil simpul E.
Kapasitas E = 50 + 10 = 60
Bobot dari sisi E-B = min(80,60) = 60.



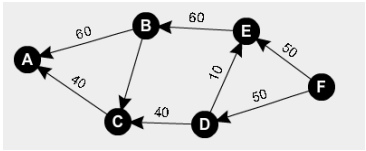
Kapasitas Simpul
A : 0 D : 50
B : 0 E : 60
C : 0 F : ∞
Visited = {F,D,E}

Ambil simpul B
Kapasitas B = 60
Bobot dari sisi (B,A) = min(100,60) = 60
Bobot dari sisi (B,C) = min(30,0) = 0



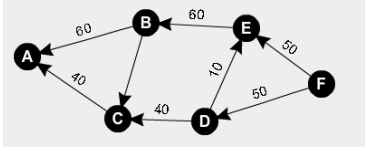
Kapasitas Simpul
A : 0 D : 50
B : 60 E : 60
C : 0 F : ∞
Visited = {F,D,E,B}

Ambil simpul C
Kapasitas C = 40
Bobot dari sisi (C,A) = min(100,40) = 40



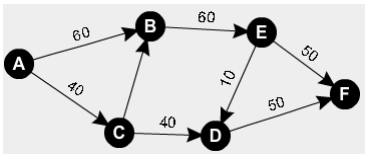
Kapasitas Simpul
A : 0 D : 50
B : 60 E : 60
C : 40 F : ∞
Visited = {F,D,E,B,C}

Ambil simpul A
Kapasitas A = 60 + 40 = 100



Kapasitas Simpul
A : 100 D : 50
B : 60 E : 60
C : 40 F : ∞
Visited = {F,D,E,B,C,A}

Balik seluruh sisi menjadi keadaan awal



Dengan demikian kita sudah mengubah graf tersebut dengan kapasitas efektif. Perhatikan juga sisi C-B memiliki kapasitas efektif nol, artinya sebenarnya kita tidak membutuhkan sisi C-B.

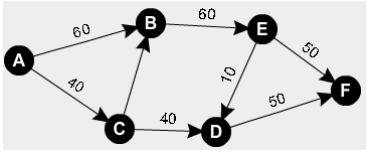
Selanjutnya, kita akan mencari bagaimana cara mendistribusikan kendaraan agar tidak terjadi kemacetan. Kemacetan terjadi ketika jumlah kendaraan yang lewat melebihi kapasitas efektif jalan.

Misalkan pass(e) menyatakan banyak kendaraan yang lewat pada sisi e dan cost(e) menyatakan kapasitas efektif jalan. Algoritma yang digunakan adalah sebagai berikut :

1. Cari sisi *e* yang paling dekat dengan simpul *source* sedemikian hingga *pass(e) > cost(e).* Jika tidak ada maka algoritma selesai.
2. Hitung sisa kendaraan yang harus didistribusikan *sisa = pass(e) – cost(e).*
3. Cari simpul initial *u* dari sisi tersebut.
4. Jika sisa = 0, tunggu berberapa saat. Letakkan petugas keamanan atau penjaga di simpul ini agar pendistribusian yang sudah dilakukan tidak berubah dan tidak kembali macet. Kembali ke langkah 1.
5. Langkah ini bertujuan untuk mendistribusikan kelebihan kendaraan pada *e* untuk dialihkan ke sisi lain. Cari *x* di mana *u* merupakan simpul inisial dari *x*, *v* merupakan simpul terminal dan *pass(x) < cost(x). Delta* adalah seberapa banyak kita mengalihkan kendaraan dari sisi *e* ke sisi *x.* Hitung *delta = min(sisa,cost(x) – pass(x))* dan kurangi *sisa* dengan *delta.* Jika tidak ada sisi *x* yang memenuhi, artinya ada suatu sisi *e'* yang juga mengalami kemacetan dan berimbas kepada *e.* Sisi *e'* harus didistribusikan ulang sehingga kendaraan yang melewati *e* dapat didistribusikan kembali. Kembali ke langkah 1. Jika ada sisi *x* yang memenuhi, lanjut ke langkah 6
6. Setelah ini, kembali ke langkah 4.

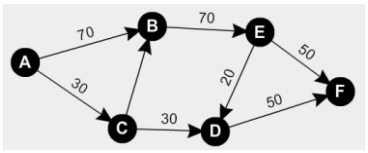
Langkah 3 mewajibkan tunggu berberapa saat. Hal ini dikarenakan pada keadaan di lapangan, setelah ada pendistribusian ulang pada suatu sisi, akan terjadi kemacetan di sisi lain. Namun, sisi yang lain ini pasti berada setelah sisi *e* dan lebih jauh dari simpul asal sehingga pengulangan algoritma di atas tidak akan mengakibatkan *infinite loop*.

Sebagai contoh, misalkan terdapat graf kapasitas sebagai berikut :



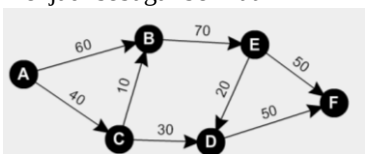
Kapasitas Sisi
AB : 60 CD : 40
AC : 40 ED : 10
CB : 0 EF : 50
BE : 60 DF : 50

Sedangkan keadaan di lapangan sebagai berikut



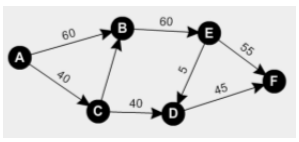
Kapasitas Sisi
AB : 60 CD : 40
AC : 40 ED : 10
CB : 0 EF : 50
BE : 60 DF : 50

Langkah-langkah yang dilakukan adalah sebagai berikut
Sisi AB mengalami kemacetan (70 > 60) , maka distribusikan AB sehingga AB = 60 dan AC = 40. Setelah ditunggu sekian menit, keadaan dilapangan berubah menjadi sebagai berikut



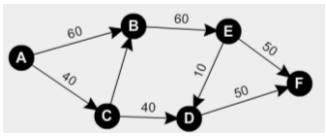
Kapasitas Sisi
AB : 60 CD : 40
AC : 40 ED : 10
CB : 0 EF : 50
BE : 60 DF : 50

Berdasarkan gambar di atas, 10 kendaraan yang dipindah dari AB ke AC ternyata melewati CB sehingga *e* = BE menjadi macet. Sisi CB juga mengalami kemacetan (10 > 0). Perhatikan bahwa ketika ambil sisi *e* = BE, pendistribusian ulang tidak dapat dilakukan sehingga kita harus memilih sisi lain, yaitu *e' = CB.* Pindahkan kendaraan dari CB ke CD. Setelah misalkan keadaan di lapangan berubah menjadi seperti ini.



Kapasitas Sisi
AB : 60 CD : 40
AC : 40 ED : 10
CB : 0 EF : 50
BE : 60 DF : 50

Ambil sisi EF dan distribusikan kemacetan ke sisi ED sehingga keadaan lapangan menjadi sebagai berikut



Kapasitas Sisi
AB : 60 CD : 40
AC : 40 ED : 10
CB : 0 EF : 50
BE : 60 DF : 50

Setelah langkah ini tidak ada lagi jalan yang macet sehingga bisa dikatakan pengalihan arus lalu lintas berhasil dilakukan.

3.4. Pengembangan Algoritma

Ada berberapa trik dan cara untuk mengembangkan algoritma ini.

Pertama, saat menentukan kapasitas relatif, pada bagian **distribusikan kembali secara sekuensial** pemilihan edge diambil secara *greedy* sekuensial tanpa mempertimbangkan sisi-sisi lain. Hal ini mengakibatkan pada contoh di atas ketika mengambil simpul B, sisi CB menjadi tidak digunakan sama sekali karena seluruh kendaraan dialihkan pada sisi AB. Agar pendistribusian

bisa rapi, ada berberapa alternatif

- Memberikan prioritas pada berberapa sisi, misal sisi CB diprioritaskan untuk dipenuhi terlebih dahulu. Pada keadaan di lapangan, prioritas ini bisa dipertimbangkan dari jarak yang ditempuh pada sisi, medan, atau *demand* dari pengguna kendaraan sendiri.
- Membagi dengan proporsi tertentu, misal pembagian dilakukan sehingga minimal setiap jalan dilewati oleh 5 kendaraan.

Kedua, saat mendistribusikan kendaraan pada jalan yang macet, dilakukan sedemikian hingga total selisih dari *pass(e)* awal dan *pass(e)* akhir seminimal mungkin.

Ketiga, penerapan algoritma pengalihan jalan dengan *multiple source* dan *multiple finish.* Sumber kendaraan dapat berasal dari lebih dari satu tempat dan menuju lebih dari satu tempat. Pengembangan algoritma untuk *multiple source* dan *multiple finish* tidak dibahas dalam makalah ini.

IV. KESIMPULAN

Jika terdapat, peta yang dapat direpresentasikan dalam sebuah *directed acyclic graph* dengan setiap sisi memuat kapasitas dan jumlah kendaraan yang lewat, maka kita dapat mendistribusikan ulang jumlah kendaraan yang lewat dengan mengalihkan arus lalu lintas sehingga setiap jalan menampung jumlah kendaraan tidak melebihi kapasitasnya.

REFERENSI

[1] Google Maps, http://maps.google.co.id, tanggal akses : 14 Desember 2013, pukul 15.00 WIB.

[2] Graph Creator, http://illuminations.nctm.org/Activity.aspx?id=3550, tanggal akses : 14 Desember 2013, pukul 9:00 WIB

[3] Jasa Marga, http://www.jasamarga.com/id_/release/item/645-jasa-marga-lakukan-rekayasa-lalu-lintas-untuk-memperlancar-arus-mudik-lebaran-2013.html tanggal akses: 13 Desember 2013, pukul 20.00 WIB

[4] Liem, Inggriani, *Diktat Struktur Data*. Bandung: Program Studi Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2008.

[5] Munir, Rinaldi. 2009. *Matematika Diskrit*. Bandung : Informatika.

[6] Rosen, Kenneth H., *Discrete Mathematics and Its Applications, 4th*, McGraw-Hill International 1999.

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 2013

Muhammad Yafi
13512014