Pemodelan Intersection dengan Compatibility Graph dan Menentukan Waktu Optimal secara Brute Force

Christopher Jeffrey - 13520055[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520055@std.stei.itb.ac.id

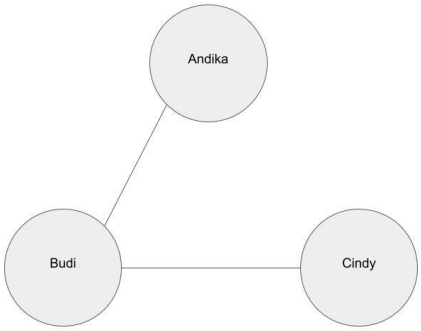
***Abstract*—At some point during our development as a human species, the need for traffic control suddenly got important. The need travel between places got very important, and our time become a valuable resource. To optimize our time, we should find a way to maximize our resource as optimal as possible. One way to do it in the traffic world is by deciding the optimal right-of-way duration for each clique in an intersection. We can also use a brute force algorithm to simulate the intersection and find the best right-of-way duration.**

***Keywords*—Right-of-way, Brute Force, Intersection, Time.**

I. PENDAHULUAN

A. Graph

Dalam matematika, ada banyak cara untuk menyimpan suatu informasi. Sebuah *placeholder* untuk sebuah informasi sederhana biasanya variabel. Seiring berkembangnya ilmu pengetahuan, kita mencoba hal-hal baru dalam rangka mendapatkan informasi yang masih belum kita ketahui dan seiring berjalannya eksperimen tersebut, beberapa temuan bertahan. Informasi yang kita miliki seringkali tidak hanya satu. Misalnya kita ingin satu simbol mewakili data kecepatan, dan simbol lainnya mewakili data waktu. Terkadang juga simbol memiliki properti yang sama, namun dimiliki oleh lebih dari satu instansi yang berbeda. Misalnya simbol satu mewakili kecepatan A dan simbol dua mewakili kecepatan si B. Misal kita memiliki data nama orang, Andika, Budi, dan Cindy. Kita dapat menyimpannya dalam suatu simbol. Lalu kita ingin menyimpan informasi pertemanan diantara ketiga orang tersebut. Kita bisa membuat data baru yang mewakili pertemanan di antara keduanya, misalnya data (Andika, Budi), jika keduanya saling mengenal. Informasi tersebut ternyata dapat direpresentasikan dengan gambar berikut.



Gambar 1. Visualisasi data saling mengenal
sumber: arsip penulis

Di gambar tersebut kita bisa tahu bahwa ada tiga orang, Andika, Budi dan Cindy, dan dari garis di antara data tersebut kita tahu bahwa Andika dan Budi saling mengenal, begitu juga dengan Cindy dan Budi.

Gambar tersebut disebut dengan graph. Graph adalah sebuah cara untuk memodelkan data. Keunggulan graph adalah adanya visualisasi hubungan antara instansi data(yaitu dengan garis diantara).

Terminologi yang sangat melekat dengan graph adalah, *vertex* (*vertices* untuk jamak, simpul dalam bahasa Indonesia) adalah satuan instansi data dalam graph. Pada *gambar 1*, Andika adalah vertex, begitu juga dengan Budi, dan Cindy. Terminologi kedua adalah *edge* (sisi dalam bahasa Indonesia), yaitu perwujudan hubungan antara vertex. Pada *gambar 1*, edge mewakili hubungan saling mengenal, dan digambarkan dengan garis. Andika dan Budi dihubungkan dengan sebuah edge, artinya Andika dan Budi memiliki hubungan saling mengenal. Graph dapat dituliskan dengan G = (V, E), dengan V adalah himpunan vertex dan E adalah himpunan edge.

Graph dapat diklasifikasikan menjadi berbagai macam. Pengklasifikasian bisa berdasarkan tampak(atau gambar) graph tersebut, berdasarkan edge yang lebih khusus(misalnya edge yang hanya bersifat satu arah, *directed graph*), dan batasan-batasan lainnya yang dibuat oleh kita. Batasan ini dibuat karena graph dengan sifat tersebut sering digunakan, dan pembatasan tersebut memungkinkan untuk mendalaminya lebih dalam lagi. Graph yang akan digunakan untuk memodelkan intersection pada makalah ini disebut dengan *compatibility graph.*

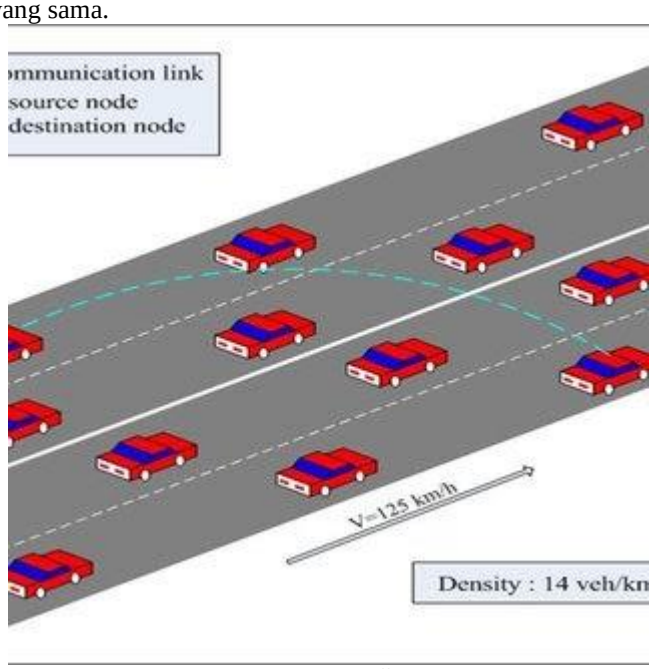
B. Traffic Flow

Traffic flow adalah cabang ilmu yang mempelajari tentang interaksi antara objek ketika menggunakan sebuah *resource* bersama. Tujuan mempelajari traffic flow adalah agar kita dapat menemukan sistem yang optimal, yaitu ketika semua objek dapat melakukan tujuannya dengan sebaik mungkin (berdasarkan parameter tertentu, misalnya waktu). Lalu lintas menerapkan ilmu ini. Objek-objeknya adalah manusia-manusia yang bepergian, entah menggunakan kendaraan mobil, sepeda motor, berjalan kaki, ataupun kendaraan lainnya. *Resource* yang digunakan bersama adalah jalanan umum. Sedangkan untuk tujuan, salah satunya adalah memastikan agar setiap kendaraan dapat berjalan mencapai tujuannya dalam waktu yang sesingkat-singkatnya. Ini bukan perkara yang mudah, dari pandangan sekilas dapat kita sadari bahwa semakin cepat, misalnya A, mencapai tujuannya, bisa saja hal tersebut memperlambat,

misalnya B, mencapai tujuannya(karena *resource* bersama digunakan oleh A, sehingga B tidak dapat menggunakannya di saat tersebut).

C. Traffic Stream

Salah satu diksi yang digunakan ketika meninjau permasalahan traffic flow adalah *traffic stream*.
Traffic stream adalah sekumpulan objek yang memiliki arah gerak yang sama, yaitu memiliki lokasi awal dan lokasi akhir yang sama.



Gambar 2. Contoh traffic stream
sumber: https://www.researchgate.net/figure/Unidirectional-uniform-traffic-stream-on-4-lanes-with-a-density-of-14veh-km-lane-and-a_fig11_224116337

Pada gambar diatas, terdapat sekumpulan objek(mobil merah) yang memiliki arah gerak yang sama. Artinya, gambar diatas dapat kita sebut sebagai suatu traffic stream. Dalam sebuah jaringan jalan yang terstruktur, sebuah traffic stream pasti memiliki bagian jalannya sendiri.

D. Intersection

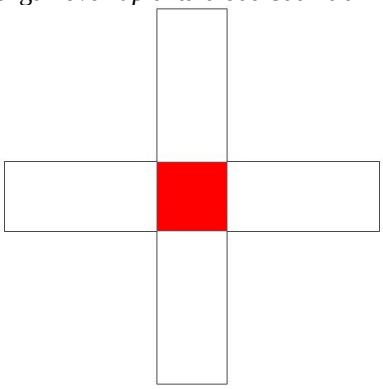
Intersection adalah sebuah tempat ketika dua buah traffic stream atau lebih, bertemu(tepatnya berpotongan).

Gambar 3. Contoh intersection
sumber: https://unsplash.com/photos/FvboK58pymA

Pada gambar tiga dapat dilihat terdapat beberapa traffic stream yang bertemu.

E. Conflict Area

Conflict area adalah sebuah area terjadinya penggunaan *resource* bersama. Dalam sebuah intersection, conflict area adalah area dengan *overlap* antara dua buah traffic stream.



Gambar 4. Gambar conflict area pada intersection
sumber: arsip penulis

Pada gambar 4, intersection telah disederhanakan dengan menghilangkan informasi yang tidak diperlukan, seperti visualisasi dari mobil dan objek-objek diluar jalan (seperti pohon). Menyisakan *border* dari jalan, dan conflict area. Conflict area pada gambar tersebut ditandai dengan warna merah.

F. Traffic Light

Gambar 5. Contoh traffic light
sumber: https://unsplash.com/photos/pB0y7Nf7xpU

Traffic light adalah sebuah alat yang digunakan untuk memberikan informasi, traffic stream mana yang memiliki hak untuk menggunakan *resource* bersama. Biasanya traffic light diletakkan di intersection, tepat sebelum conflict area. Informasi yang diberikan traffic light bermediakan tiga buah lampu, berwarna merah, kuning(atau amber, orange,jingga), dan hijau yang mengikuti sebuah perjanjian universal. Warna merah bermakna larangan untuk menggunakan conflict area. Warna hijau bermakna sebuah traffic stream memiliki hak untuk menggunakan conflict area. Warna kuning digunakan sebagai warna transisi, tepatnya transisi dari merah ke hijau, atau hijau ke merah. Lampu ini digunakan agar para pengguna jalan bisa lebih hati-hati. Traffic light memiliki siklus yang berulang, merah-kuning-hijau-kuning, lalu kembali merah.

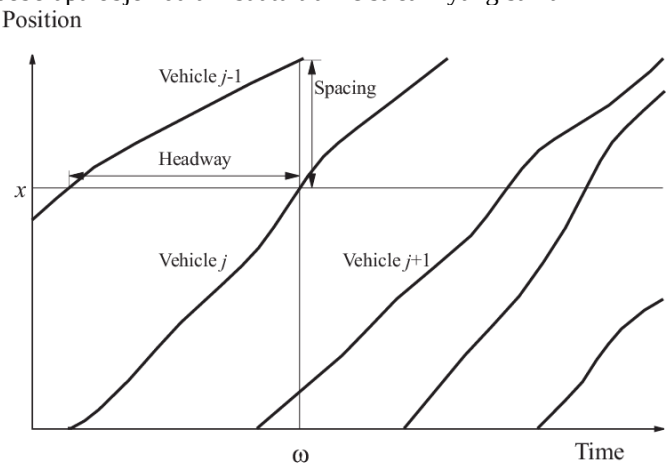
G. Right-of-way

Ketika sebuah traffic stream mendapatkan informasi warna hijau dari traffic light, artinya dia memiliki hak untuk menggunakan conflict area. Traffic stream tersebut dapat disebut memiliki *right-of-way* (ROW). Jadi, traffic light dapat

dikatakan sebagai sebuah alat untuk menentukan ROW.

H. Time-space Diagram

Time-space diagram adalah sebuah diagram dua dimensi dengan masing-masing axis mewakili waktu(time), dan lainnya jarak(space). Dengan diagram ini, kita dapat dengan cepat mendapat informasi mengenai kecepatan suatu objek. Jika beberapa objek dalam suatu traffic stream yang sama



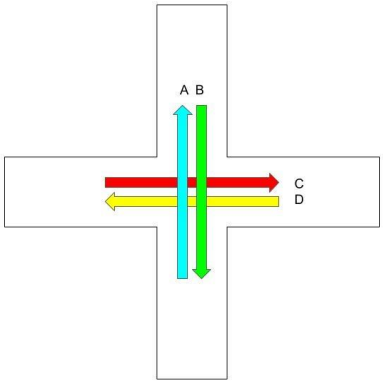
Gambar 6. Contoh space-time diagram
sumber: https://www.researchgate.net/figure/Time-space-diagram-of-vehicle-trajectories_fig2_285485763

I. Flow

Flow adalah perbandingan antara jumlah kendaraan yang melewati suatu titik, dengan waktu. Nilainya dapat dihitung dari jumlah kendaraan yang melewati suatu titik dibagi dengan waktu. Flow dapat berupa jumlah kendaraan tiap jam, atau jumlah kendaraan tiap detik. Pada persoalan ini, akan digunakan jumlah kendaraan tiap detik. Untuk memudahkan penggunaan flow dalam masalah traffic control, kita buat dua buah diksi baru. Flow memasuki sebuah traffic stream kita sebut dengan *in flow*, flow keluar dari suatu traffic stream kita sebut dengan *out flow*.

II. MEMODELKAN INTERSECTION DENGAN GRAPH

Tujuan dari traffic control adalah memastikan semua objek(kendaraan) dapat bergerak dari posisi asal ke tujuannya dengan secepat-cepatnya(tentunya dalam batas aman dalam kondisi nyata). Idealnya, semua kendaraan dapat bergerak dari asal ke tujuan dengan garis lurus, namun hal ini tidak mungkin, terlalu banyak kemungkinan pergerakan dari masing-masing kendaraan, dan membuat sebuah jalan untuk masing-masing kemungkinan tersebut jelas tidak efisien, atau bahkan tidak mungkin. Opsi lainnya adalah menggunakan jalan bersama, yang relatif berliku dan memutar. Hal ini lebih masuk akal daripada opsi pertama, namun ini bukan cara yang paling cepat untuk mencapai objektif, yaitu *meminimalkan waktu perjalanan masing-masing kendaraan*. Karena itu, dibuatlah intersection. Selain lebih efektif dari pembangunan, opsi ini bisa saja lebih efektif daripada opsi kedua(terjadi ketika opsi kedua terlalu berliku dan menyebabkan waktu perjalanan menjadi lebih lama). Dalam sebuah intersection, kita menggunakan traffic light untuk menentukan traffic stream mana yang memiliki right-of-way. Salah satu cara penentuan ini adalah, masing-masing traffic stream mendapatkan ROW dalam durasi tertentu. Hal ini dilakukan untuk masing-masing traffic stream, lalu diulang kembali. Hal ini bisa bekerja, namun tidak efektif.

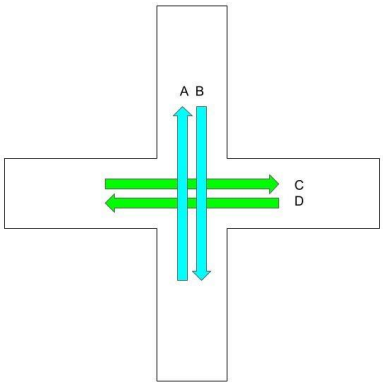


Gambar 7. Contoh intersection

sumber: arsip penulis

Misalnya terdapat 4 buah traffic stream, dan masing-masing traffic stream mendapatkan durasi ROW selama 45 detik. Artinya durasi satu buah *cycle* adalah durasi tunggu dikali dengan jumlah traffic stream, yaitu 180 detik. Artinya, setiap cycle, suatu traffic stream mendapatkan durasi berjalan sebesar 45 detik dan durasi tunggu sebesar 135 detik. Pada gambar x, tiap warna mewakili giliran yang berbeda-beda. Tiap traffic stream memiliki warna yang berbeda, sehingga masing-masing traffic stream memiliki giliran yang berbeda.

Terdapat cara yang lebih efektif.



Gambar 8. Ilustrasi intersection dengan penentuan right-of-way yang lebih efektif

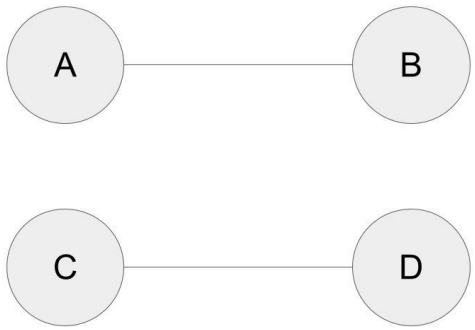
sumber: arsip penulis

Pada gambar tersebut, traffic stream A dan B dapat menggunakan conflict area bersamaan. Begitu juga, traffic stream C dan D dapat menggunakan conflict area bersamaan. Dengan durasi ROW yang sama, cycle berkurang menjadi 90 detik. Setiap cycle, suatu traffic stream mendapatkan durasi berjalan sebesar 45 detik dan durasi tunggu 45 detik. Hal ini lebih baik dan lebih efektif untuk mencapai tujuan daripada opsi sebelumnya. Semakin cepat suatu cycle, maka semakin sedikit waktu yang dihabiskan oleh setiap kendaraan, dan semakin sedikit kendaraan yang *menumpuk* di suatu traffic stream. Menentukan waktu cycle terendah dengan waktu ROW

maksimal itu penting, karena hal ini yang akan meminimalkan waktu tunggu tiap kendaraan, yaitu tujuan dari traffic control.

Pada contoh tersebut, yang kita lakukan adalah *memberikan right-of-way di waktu yang sama, untuk lebih dari satu buah traffic stream.* Hal ini menjadi salah satu alat utama dalam meminimalkan waktu tunggu. Langkah yang perlu kita lakukan untuk melakukan hal tersebut adalah menentukan traffic stream yang dapat mendapatkan right-of-way bersamaan. Idenya adalah, ketika jalur dua buah traffic stream tidak melakukan perpotongan, artinya kedua traffic stream tersebut dapat berjalan bersamaan.

Dengan menggunakan compatibility graph, kita dapat menghilangkan abstraksi sebuah intersection nyata, menyisakan data-data yang kita butuhkan untuk menentukkan traffic stream dengan ROW bersamaan. Vertex mewakili individual traffic stream. Edge mewakili hubungan path yang dimiliki vertex(yaitu individual traffic stream) tidak berpotongan. Pada contoh diatas, terdapat empat buah traffic stream. Jika kita modelkan menjadi graph, akan terdapat empat buah vertex, masing-masing mewakili traffic stream A, B, C, dan D. Selanjutnya, pada *gambar 8*, dapat kita lihat bahwa jalur A dan B tidak berpotongan. Begitu juga dengan C dan D. Sehingga pada graph kita, akan ada edge antara A dengan B, dan C dengan D. jalur A dengan C berpotongan sehingga, tidak ada edge antara A dengan C. begitu juga A dengan D, B dengan C, dan B dengan D.



Gambar 9. Graph pemodelan intersection pada gambar 8

sumber: arsip penulis

Traffic stream yang dapat berjalan bersamaan dapat kita sebut dengan *clique*. Jadi A dengan B dapat kita sebut sebagai suatu clique, begitu juga C dengan D.

III. Algoritma untuk Menentukan Durasi ROW Optimal

Setelah berhasil memodelkan suatu intersection menjadi sebuah graph, kita bisa lanjut ke langkah selanjutnya, yaitu menemukan durasi ROW yang paling optimal untuk masing-masing clique. Data yang dimiliki setiap traffic stream sangat kompleks, mulai dari kecepatan masing-masing kendaraan, ukuran kendaraan, in flow yang gradual. Untuk menyederhanakan permasalahan, kita anggap data kompleks tersebut berhasil disederhanakan menjadi hal - hal berikut

1. in flow
2. out flow
3. jumlah kendaraan saat ini
4. jumlah kendaraan maksimal

Dengan bermodalkan empat buah data tersebut, kita dapat membuat suatu algoritma untuk menentukkan durasi ROW yang paling efektif. Pendekatan penyelesaian masalah ini dapat dilakukan dari berbagai arah. Pendekatan yang akan digunakan disini adalah *brute force*, yaitu mencoba semua kemungkinan konfigurasi durasi right-of-way dari masing-masing clique, dan memilih konfigurasi yang memberikan durasi tunggu kendaraan yang paling rendah. Pemilihan pendekatan brute force didasarkan dengan, pendekatan ini cukup mudah untuk dimengerti. Kekurangan pendekatan ini adalah kompleksitas yang tidak efektif (dalam notasi big-oh, kompleksitas waktu algortima yang dibuat disini adalah$O(N^5)$, dengan N sebagai baris dan kolom dari clique), namun karena data yang digunakan masih dalam batas wajar, sehingga tidak memberikan masalah berarti (seperti durasi algoritma yang relatif lama, hingga durasi jam). Berikut langkah algoritma satu per satu.

1. ubah intersection menjadi graph lengkap dengan spesifikasi dari masing-masing traffic stream. Buat pula clique yang efektif.
2. tentukan durasi maksimal ROW dari masing-masing clique. idenya adalah, lampu tidak boleh hijau(memberikan ROW kepada traffic stream tersebut) ketika kendaraan yang bisa melaju dibawah kendaraan optimal. Hal ini dilakukan agar jumlah kendaraan yang menggunakan conflict area selalu optimal. hal ini dapat kita hitung dengan jumlah kendaraan saat ini dibagi dengan out flow.
3. uji coba semua konfigurasi, mulai dari 0 hingga max time. uji coba dilakukan dengan cara mensimulasikan waktu waktu konfigurasi, untuk mendapatkan waktu tunggu rata-rata kendaraan, yang didapatkan dari waktu total perjalanan dan total kendaraan yang lewat.
4. konfigurasi optimal didapat dari waktu tunggu rata-rata kendaraan yang paling minimal.

IV. Implementasi Algoritma

Berikut implementasi program tersebut dalam bahasa C.
Pada langkah satu, intersection perlu kita ubah menjadi sebuah graph. Graph dapat kita implementasikan dengan menggunakan *adjacency list*.

```
/*LANGKAH SATU  SETUP GRAPH DAN CLIQUE */
   int cliqueCount = 3;
   int maxTrafficStreamInOneClique = 2;
   int sizeOfClique[cliqueCount];
   char *clique = malloc(sizeof(char)* cliqueCount *
maxTrafficStreamInOneClique);
   graph g = createGraph();

   /*membuat graph*/
   addVertex(&g, 'A', 1, 6, 200, 50);
   addVertex(&g, 'B', 3, 10, 200, 50);
   addVertex(&g, 'C', 4, 15, 200, 50);
   addVertex(&g, 'D', 3, 4, 200, 50);
   addVertex(&g, 'E', 1, 4, 200, 50);
   addVertex(&g, 'F', 2, 10, 200, 50);
   addEdgeBetweenTwoVertex(&g, 'A', 'B');
   addEdgeBetweenTwoVertex(&g, 'C', 'D');
   addEdgeBetweenTwoVertex(&g, 'E', 'F');

   /*
   [
      ['A','B'],
      ['C', 'D'],
```

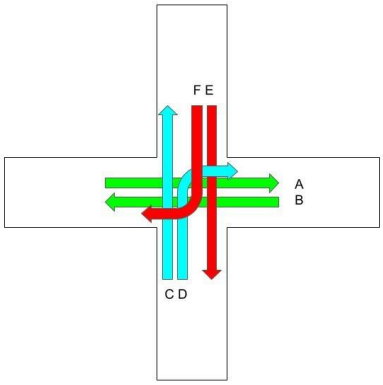
```
    ['E', 'F']
  ]
  */

  /*membuat clique, disimpan dalam bentuk array*/
  *(clique + 0) = 'A';
  *(clique + 1) = 'B';
  *(clique + 2) = 'C';
  *(clique + 3) = 'D';
  *(clique + 4) = 'E';
  *(clique + 5) = 'F';
  sizeOfClique[0] = 2;
  sizeOfClique[1] = 2;
  sizeOfClique[2] = 2;
```

Pada kode tersebut, intersection yang digunakan memiliki enam buah jalur, dengan nama terurut A hingga F. masing-masing traffic stream memiliki konfigurasi sebagai berikut,

1. Traffic stream A
   in flow 1 kendaraan tiap detik
   out flow 6 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50
2. Traffic stream B
   in flow 3 kendaraan tiap detik
   out flow 10 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50
3. Traffic stream C
   in flow 4 kendaraan tiap detik
   out flow 15 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50
4. Traffic stream D
   in flow 3 kendaraan tiap detik
   out flow 4 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50
5. Traffic stream E
   in flow 1 kendaraan tiap detik
   out flow 4 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50
6. Traffic stream F
   in flow 2 kendaraan tiap detik
   out flow 10 kendaraan tiap detik
   jumlah maksimum kendaraan yang dapat ditampung sebanyak 200 kendaraan
   jumlah kendaraan saat ini sebanyak 50

Konfigurasi ini dibuat sebagai contoh untuk seluruh penjelasan implementasi algoritma. Traffic stream tersebut dapat di visualisasikan dalam *gambar 10*.



Gambar 10. Visualisasi intersection
sumber: arsip penulis

Dari gambar dapat dilihat bahwa jalur A dengan B tidak saling berpotongan, sehingga dapat dijadikan sebagai clique. Begitu juga C dengan D, dan E dengan F. akan didapat tiga buah clique, yaitu {A,B}, {C,D}, {E,F}.

Selanjutnya pada langkah kedua, kita perlu menentukkan durasi lampu hijau maksimum dari masing-masing clique. Durasi maksimum dapat ditentukan dari nilai minimum dari jumlah kendaraan dibagi out flow masing-masing traffic stream di suatu clique. Contohnya pada clique {C,D}, durasi maksimum traffic stream C mendapatkan right-of-way adalah 50 dibagi 15. Hasilnya kita bulatkan ke bawah, dan didapatkan angka 3. Pembulatan dibawah dilakukan agar penggunaan conflict area selalu optimal. Jika dibulatkan keatas, jumlah kendaraan yang keluar dari suatu traffic stream akan lebih rendah daripada jumlah out flow, sehingga tidak optimal. Durasi maksimum traffic stream D mendapatkan right-of-way adalah 50 dibagi 4, yaitu 12 (dibulatkan ke bawah juga, sama dengan argumen sebelumnya). Karena 3 kurang dari 12, right-of-way maksimum suatu clique {C,D} adalah 3. Hal ini dilakukan untuk setiap clique.

Berikut algoritma yang digunakan untuk mendapatkan durasi right-of-way maksimum tiap clique.

```
/*LANGKAH DUA FINDING MAX ALLOWED TIME*/
    float bestTime = 999;
    int bestConfig[3];
    int maxTime[3];
    maxTime[0] = 999;
    maxTime[1] = 999;
    maxTime[2] = 999;

    for(int b = 0 ; b < 3 ; b++){
        for(int k = 0 ; k < sizeOfClique[b]; k++){
            AddressVertex walker = g;
            while(walker->id != *(clique +(b*sizeOfClique[b]) + k)){
                walker = walker->next;
            }
            int temp = floor((float)walker->currentVehicleCount /
(float)walker->outFlow);
            if(temp < maxTime[b]){
                maxTime[b] = temp;
            }
        }
    }
```

Selanjutnya langkah keempat. Akan diuji coba tiap konfigurasi, dari satu hingga durasi right-of-way maksimum tiap clique. Pada intersection gambar x, durasi maksimumnya adalah 5, 3, 5, berurutan dari clique {A,B}, {C,D}, dan {E,F}. Maka konfigurasi yang akan diujikan mulai dari 1, 1, 1; 1, 1, 2; hingga terakhir 5, 3, 5. Bisa kita hitung jumlah kemungkinannya, yaitu

5 dikali 3 dikali 5, 75 kemungkinan konfigurasi. Berikut kode untuk mengujikan semua konfigurasi.

```
/*LANGKAH TIGA UJI COBA SEMUA KONFIGURASI*/
   for(int i = 1; i <= maxTime[0]; i++){
      for(int j = 1; j <= maxTime[1]; j++){
         for(int k = 1; k <= maxTime[2]; k++){
            int timeToTest[3];
            timeToTest[0] = i;
            timeToTest[1] = j;
            timeToTest[2] = k;
            graph tempGraph = copyGraph(g);
            float temp = computeAverageWaitingTime(tempGraph, timeToTest,
clique, 3, 2, sizeOfClique);
            AddressVertex walker = g;
            AddressVertex walkerTemp = tempGraph;
            for(int i = 0 ; i < 6 ; i++){
               if(walker->currentVehicleCount <
walkerTemp->currentVehicleCount){
                  temp = 999;
                  break;
               }
               else{
                  walker = walker->next;
                  walkerTemp = walkerTemp->next;
               }
            }
            if (temp < bestTime && temp > 0){
               bestTime = temp;
               bestConfig[0] = i;
               bestConfig[1] = j;
               bestConfig[2] = k;
            }
         }
      }
   }
```

Dalam menentukan waktu terbaik, algoritma ini memiliki beberapa tambahan ketentuan. Jumlah kendaraan yang di akhir harus lebih kecil atau sama dengan jumlah kendaraan di awal. Jika hal ini tidak dipenuhi, artinya konfigurasi mengarah kepada kemacetan. Untuk memberikan abstraksi, kita menggunakan sebuah fungsi yang akan menerima graph intersection tersebut beserta konfigurasinya, lalu mengujikan dan memberikan nilai rata-rata tunggu kendaraan.

Terakhir pada langkah keempat, setelah di uji cobakan setiap konfigurasi, kita cari dan dapatkan konfigurasi dengan waktu tunggu terendah. Berikut algoritma langkah keempat.

```
/*OUTPUT*/
   if(bestTime == 999){
      printf("konfigurasi traffic pasti mengarah ke kemacetan\n");
   }
   else{
      printf("best time = ");
      printf("%.2f\n", bestTime);
      printf("best config = %d %d %d\n", bestConfig[0], bestConfig[1],
bestConfig[2]);
   }
```

Dapat dilihat pada algoritma tersebut, terdapat kasus ketika waktu terbaik bernilai 999. Angka ini dipilih untuk mewakili waktu terbaik yang tidak valid(dengan asumsi waktu tunggu algoritma tidak mungkin melebihi 999). Kasus ini terjadi ketika konfigurasi dari lalu lintas pasti mengarah kepada kemacetan, contohnya ketika semua in flow dari traffic stream sebesar 100 dan out flownya sebesar 1. Dalam kasus ini, diapatkan konfigurasi efektifnya adalah 1, 2, 1 terurut untuk masing-masing clique.

V. KESIMPULAN

Graph dapat membantu kita menghilangkan informasi yang tidak diperlukan (abstraksi), dan hanya menyisakan informasi yang benar-benar berguna untuk kita. Hal ini membantu kita untuk fokus kepada masalah sebenarnya yang ingin diselesaikan. Algoritma brute force bukanlah algoritma yang efisien, namun mudah diterima oleh kita karena langkahnya yang relatif terus terang (*straight forward*).

VI. UCAP SYUKUR

Puji syukur kepada Tuhan Yang Maha Esa, atas berkat dan rahmat-Nya membimbing penulis menyelesaikan makalah ini. Penulis berterima kasih secara pribadi kepada Dr. Ir. Rinaldi Munir, M.T yang telah membagikan ilmunya yang menjadi landasan penyusunan makalah ini.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/matdis20-21.htm. Diakses pada 6 Desember 2021.

[2] Baruah, A.K and Niky Baruah, "Signal Groups of Compatible Graph in Traffic Control Problems", Int. J. Advanced Networking and Applications, Pages:1473-1480.

[3] Björck, Erik and Fredrik Omstedt, A comparison of algorithms used in traffic control systems, Sweden:Stockholm, 2018.

[4] https://www.fhwa.dot.gov/publications/research/operations/tft/index.cfm. Diakses pada 6 Desember 2021.

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, 13 Desember 2021

Christopher Jeffrey
13520055