Permasalahan Clique dalam Graf

Adventus W. Lumbantobing, 13505112
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if15112@students.if.itb.ac.id

Subgraf merupakan himpunan subset titik simpul dari suatu graf yang terhubung. Pencarian subgraf berarti bertujuan mencari himpunan solusi dari suatu permasalahan besar yang dapat digambarkan dalam graf.

Subgraf merupakan salah satu konsep dasar dalam permasalahan matematis dan pembangunan graf. Dalam bidang informatika pencarian subgraf dapat meliputi pencarian apakah suatu subgraf merupakan permasalahan NP-complete. Terdapat berbagai metode algoritma yang diteliti untuk permasalahan ini.

Dalam area topik matematika tentang teori graf terdapat istilah *clique*, yaitu sebuah subgraf tak berarah yang terdapat dalam suatu graf. Pencarian maksimum *clique* telah diterapkan dalam sejumlah bidang keilmuan, keahlian teknik dan bahkan dalam bidang bisnis. Penerapan tersebut meliputi solusi permasalahan struktur molekul DNA. pemrosesan citra dalam pengaturan jarak jauh, penyocokan titik koordinat dalam sistem informasi, permasalahan partisi data dalam kepingan memori, pewarnaan graf dan lain-lainnya. Penerapan *clique* dewasa ini sering dipergunakan dalam bidang bioinformatika.

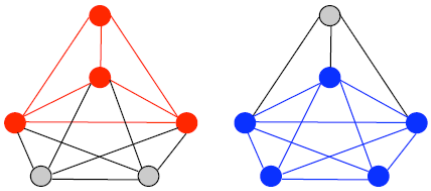
Permasalahan *clique* ini merupakan permasalahan pencarian solusi yang dianggap cukup sulit, sama seperti permasalahan *knapsack*, pewarnaan graf, sirkuit Hamilton, *n-puzzle* dan TSP. Sampai sekarang ini belum ditemukan algoritma yang benar-benar mangkus untuk penyelesaian permsalahan-permasalahan tersebut meskipun untuk beberapa kasus telah ditmukan algoritma yang memberikan solusi yang mendekati hasil yang optimal.

Kata kunci:* graf, subgraf, *clique

I. PENDAHULUAN

Sebuah subgraf G = (V,E) adalah subset dari himpunan simpul C ⊆ V sehingga setiap dua buah simpul dalam C terhubungkan dengan sebuah sisi. Subgraf maksimum adalah subgraf terbesar yang dapat terbentuk dari suatu graf. Jumlah simpul maksimum dari subgraf maksimum dinyatakan dalam ω(G). Salah satu contoh permasalahan dalam pencarian subgraf adalah pencarian subgraf minimum yang menghubungkan setiap simpul dari suatu graf.

Sebuah *clique* merupakan subgraf tak berarah dari suatu graf yang setiap simpul saling terhubung satu sama lain.



Gambar 1 Contoh clique dalam graf

Dalam gambar 1 diberikan contoh dua buah *clique* yang dapat dibentuk dari graf tersebut. Dalam suatu graf akan terdapat sebuah *clique* jika dalam graf tersebut setidaksnya terdapat sebuah upagraf lengkap berjumlah k buah simpul yang dapat berdiri sendiri. Jumlah maksimum *clique* yang munkin diperoleh dari suatu graf V dengan ukuran *clique* k dapat dirumuskan :

$$\binom{V}{k} = \frac{V!}{k!(V-k)!} \quad (1)$$

Ada 4 permasalahan umum dalam *clique*:

1. Pencarian *maximum clique*
2. Pencarian bobot maksimum
3. List *maximal clique*
4. Memecahkan masalah putusan apakah sebuah graf memiliki subgraf/*clique* dengan ukuran tertentu

2. ALGORITMA

2.1 Maximal clique

Menentukan *maximal clique* dimulai dengan *clique* dengan satu vertex dan menambahkan satu vertex baru pada graf ke *clique* sebelumnya. *Clique* akan bertumbuh dengan menambahkan vertex yang tersisa pada graf. Sebuah vertex akan dimasukkan kedalam *clique* jika setiap vertex terhububg satu sama lain. Jika tidak maka vertex

tersebut akan diabaikan. Algoritma berjalan dengan kompleksitas O(n).

2. 2 Clique dengan ukuran tertentu

Algoritma brute force dapat digunakan untuk mencari tahu apakah sebuah graf memiliki *clique* dengan jumlah vertex k. Proses ini dilakukan dengan mencari setiap subgraf dengan jumlah vertex k dan menguji coba apakah subgraf tersebut membentuk *clique*. Algoritma ini memiliki kompleksitas $O(n^k k^2)$. Terdapat $O(n^k)$ padah proses pengecekan subgraf dan setiap proses terdapat $O(k^2)$ simpul yang harus diperiksa.

2.3 List dari maximal clique

Proses me-list semua maximal *clique* dari suatu graf merupakan $O(3^{n/3})= O(1.4422^n)$. Salah satu algoritma yang dapat digunakan adalah algoritma *back-tracking* Bron-Kerbosch.

2.4 Mencari clique maksimum

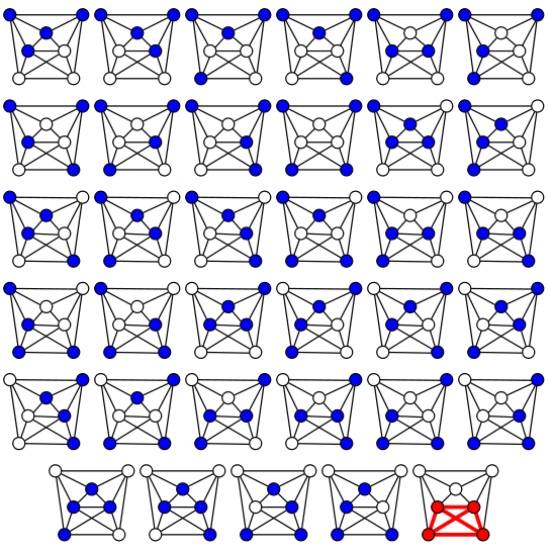
Algoritma yang tercepat untuk permsalahan ini adalah algoritma Robson yang dikembangkan berdasarkan algoritma Bron-Kerbosch dengan $O(2^{0.249n})=O(1.1888^n)$

3. METODE

Clique dapat dicari dengan menggunakan algoritma-algoritma brute-force, back-tracking dan algoritma Bron-Kerbosch dan algoritma pencarian lainnya.

3.1 Algoritma Brute Force

Dalam pencarian k-*clique* (*clique* berukuran k buah simpul) dengan algoritma brute-force dilakukan pengurutan set-set dengan nilai k. Untuk setiap simpul dalam tiap set yang bersesuaian dilakukan pemeriksaan jika setiap simpul saling bertetangga atau terhubung.



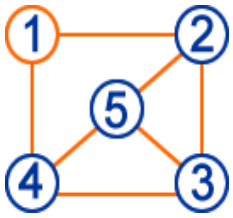
Gambar 2 Proses pencarian 4 vertex subgraf clique dalam suatu graf secara brute force

Dalam kasus pencarian *clique*, algoritma ini tidak mangkus dan sulit diimplementasikan untuk permasalahan ini.

3.2 Algoritma Backtracking

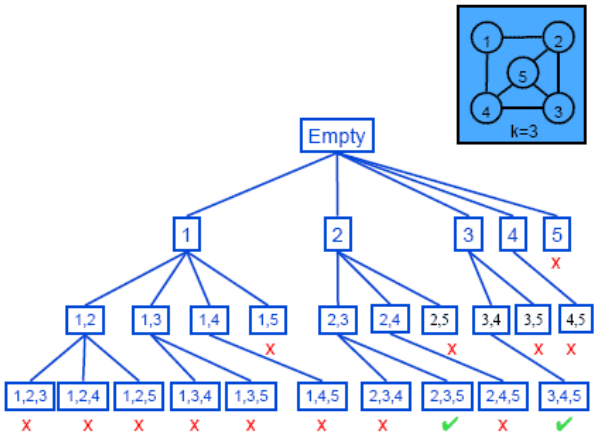
Misalkan graf G = (V,E) dan integer k, clique merupakan subset U dari V dengan $|U| \geq k$ sehinga setiap pasangan vertex dalam U saling terhubung. U merupakan *k-clique* dari graf G. Algoritma back-tracking yang digunakan yaitu dengan menggambarkan graf dalam struktur pohon lalu akar dari pohon dianggap sebagai sebuah *clique* tunggal. Dua buah *clique* akan bergabung jika setiap simpul dalam kedua buah *clique* saling terhubung.

Berikut adalah contoh yang menunjukkan pencarian *clique* dari graf G dengan k =3 :



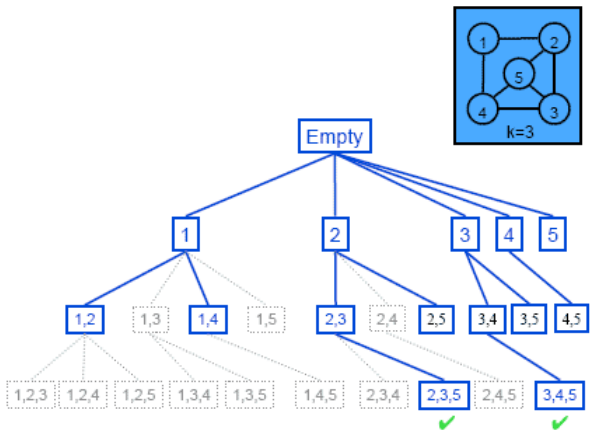
Gambar 3 Graf G dengan k=3

a. Algoritma back-tracking dengan menghitung semua kemungkinan dari setiap upagraf lalu memeriksa jika simpul tersebut terhubung dengan simpul lainnya dan ukuran *clique* bernilai k.



Gambar 4 Metode back-tracking I graf G

b. Algoritma back-tracking yang hanya memeriksa upagraf yang simpul-simpulnya saling terhubung dengan simpul yang lain dan memeriksa ukuran *clique* dengan nilai k.



Gambar 5 Metode back-tracking II graf G (metode cutoffs pruning)

Algoritma back-tracking yang kedua dengan metode *cutoffs pruning* lebih mangkus karena dengan algoritma tersebut upagraf yang tidak termasuk solusi tidak diperiksa lebih dalam lagi dengan metode DFS. Dari hasil pencarian dengan algoritma back-tracking, dari kedua cara tersebut memberikan hasil *clique* yang sama yaitu upagraf antara simpul 2,3,5 dan antara simpul 3,4,5 dan tiap simpul dari upagraf-upagraf tersebut saling terhubung satu dengan lainnya.

Pseudocode untuk pencarian k-*clique* dengan menggunakan back-tracking dengan metode cutoffs pruning dengan masukan graf sebagai vektor dan sebuah bilangan integer j:

```
CliqueBT (input A:vektor, j:integer)
{ jika j sama dengan ukuran clique, k, maka A
  adalah k-clique dari graf }
Algoritma:
  if (j == sizeClique) then
    numClique++
    return
  endif
  else
    j = j + 1
    if (j <= sizeClique) then
     { S adalah semua kemungkinan vektor dari
       j-clique }
      S  = getVektor (A)
       j
    endif
    if (S  tidak kosong) then
        j
    { Untuk setiap kemungkinan dari S ,
                                    j
      lakukan bac-tracking secara rekursif
      untuk k-clique }
      for (untuk setap vektor a  dalam S ) do
                               j       j
          CliqueBT (a , j)
                    j
      endfor
    endif
  endif
```

Pseudocode untuk mengambil daftar vektor dalam S_j sesuai dengan algoritma di atas dituliskan sebagai berikut:

```
getVektor (Vector A)
   daftarVektor { adalah S , list semua vektor
                          j
                untuk clique }
   { Jika A kosong,  S  adalah vektor pada tiap
                     j
     simpul tunggal dalam graf. }
     if (A kosong) then
       daftarVektor.add (setiap simpul dalam
       graf)
     endif
     else
       { Hitung semua daftar vektor }
       for (tiap elemen j > A.elemenAkhir)
     do
         isSemuaTerhubung := true
         { Periksa jika j berdekatan dengan
           Semua verteks dalam A }
         for (tiap tetangga i dari j) do
           if ( i and j tidak terhubung) then
             { Cutoffs pruning }
             isSemuaTerhubung := false
             break
           endif
         endfor
         if (isSemuaTerhubung == true) then
           daftarVektor.add (s )
                              j
         endif
   endfor
endif
return daftarVektor
```

Algoritma tersebut di atas bertujuan untuk mencari *clique* dalam suatu graf. Sedangkan untuk mencari *clique* maksimum (atau sering juga disebut Maximum *clique* problem) algoritma umumnya (tanpa menggunakan algoritma back-tracking) adalah sebagai berikut:

```
maximumClique(input:ccVektor,daftarVektor, I/O:
              k)
  while daftarVektor ≠ ø do
    pilih x dalam daftarVektor lalu buang
    simpan daftarVektor
    tambah x ke ccVektor
    buang simpul y dari daftarvektor
    if filter (ccVektor,daftarVektor,k)
    then
      if daftarVektor = ø then
        k←ccVektor
      endif
      else

    maksClique(ccVektor,daftarVektor,k)
      endif
```

```
      endif
      remove x from Current
      restore Candidate

   endwhile



filter (input:ccVektor,daftarVektor, I/O: k)
   do
     lanjut ← false
     for setiap y dalam daftarVektor do

       if |Γ(y)∩daftarVektor|+|ccVektor|<|k|

       then

         buang y dari daftarVektor

         if |daftarVektor|+|ccVektor|<|k|
   then

           lanjut←true

         endif

       endif

     endfor

   while lanjut

   return true
```

2.3 Algoritma Bron-Kerbosch

Algoritma Bron-Kerbosch merupakan algoritma *back-tracking* secara rekursif yang mencari *maximal clique* dalam graf G. Mencari semua subgraf dalam graf membutuhkan biaya yang besar. Jumlah *clique* dapat bertumbuh secara eksponensial sesuai dengan pertambahan simpul pada graf.

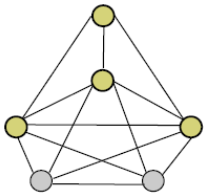
```
BronKerbosch1(R,P,X):
  if P and X are both empty:
    report R as a maximal clique
  for each vertex v in P:
    BronKerbosch1(R⋃{v},P⋂N(v),X⋂ N(v))
    P := P \ {v}
    X := X ⋃ {v}
```

Algoritma Bron-Kerbosch dapat menemukan semua *clique* dalam waktu linear dan merupakan salah satu algoritma yang tercepat dalam pencarian *clique* saat ini.

1. Simpul yang termasuk dalam suatu *clique* dalam graf dilambangkan dengan
2. Kandidat, yaitu simpul yang terhubung dengan subgraf
3. NOT, simpul yang sudah valid yang merupakan bagian dari subgraf dan tidak diakses lagi
4. Kandidat yang dipilih
5. Simpul yang tidak dimasukkan dalam pencarian

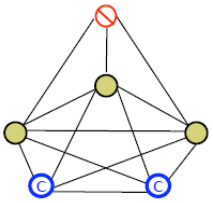
Berikut adalah skema algoritma Bron-Kerbosch dalam contoh graf pada Gambar 1:

1. Inisialisasi.
   Suatu *clique* ditemukan jika dan hanya jika tidak ada kandidat lagi dan tidak ada simpul dalam himpunan NOT jika tidak, maka *clique* tidak maksimal



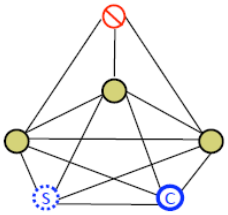
Gambar 6 Clique pertama dalam graf

2. Prosedur rekursif.
   a. Masukan salah satu simpul kedalam set NOT sehingga terdapat tiga simpul dalam subgraf dan terdapat dua kandidat tersisa



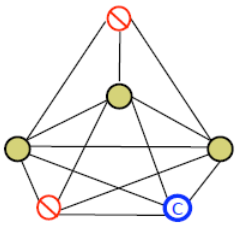
Gambar 7 Pembangkitan simpul kandidat

   b. Pilih sebuah kandidat, dan tambahkan ke himpunan subgraf



Gambar 8 Pemilihan kandidat

   c. Pilih kandidat baru dan himpunan NOT untuk rekursif berikutnya dengan memasukkan simpul yang tidak terhubung dengan kandidat



Gambar 9 Rekursif pemilihan kandidat baru

3. Terminasi.
   Jika tidak ada kandidat lagi atau terdapat simpul dalam himpunan NOT yang terhubung dengan semua simpul subgraf.

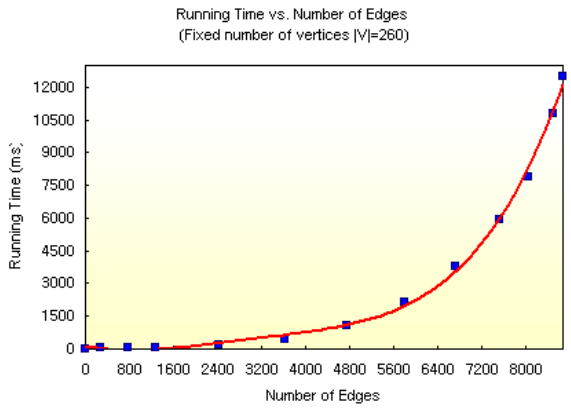
```
BronKerbosch2(R,P,X):
 if P and X are both empty:
    report R as a maximal clique
 choose a pivot vertex u in P ⋃ X
 for each vertex v in P \ N(u):
    BronKerbosch2(R⋃{v},P⋂N(v),X⋂ N(v))
    P := P \ {v}
    X := X ⋃ {v}
```

Dengan menggunakan metode pivot, jumlah proses rekursif dapat diminimumkan sehingga algoritma Bron-Kerbosch merupakan $O(3^{n/3})$.

3. HASIL IMPLEMENTASI

Algoritma bruteforce dinilai kurang mangkus dalam menyelesaikan persoalan ini karena sangat membutuhkan biaya dan waktu yang besar.

Algoritma backtracking memberikan hasil yang cukup baik. Kemangkusan k-*clique* dengan algoritma back-



Gambar 10 Hasil uji coba pencarian clique dengan back-tracking

tracking bergantung pada struktur dari graf itu sendiri, algoritma tersebut masih memiliki kompleksitas waktu untuk kasus terburuk $O(n^k)$ dengan nilai k sebagai variabel ukuran dari *clique*. Nilai n dapat berupa nilai dari simpul atau sisi yang masing-masing dapat dinyatakan dalam suatu fungsi yang saling berlawanan. Fungsi yang dihasilkan merupakan fungsi eksponensial.

Algoritma Bron-Kerbosch merupakan algoritma yang dinilai tercepat saat ini dan sangat efektif karena dapat menemukan smua *clique* dalam suatu graf.

4. KESIMPULAN

Pencarian *clique* dari suatu graf dapat dilakukan dengan metode bruteforce, back-tracking, Bron-Kerbosch dan berbagai metode lainnya.

Metode bruteforce dilakukan dengan pemeriksaan semua simpul yang terhubung dengan suatu subgraf. Metode backtracking menggunakan graf dalam bentuk pohon untuk mempermudah pencarian dengan metode DFS. Algoritma ini akan memeriksa hubungan antar simpul dalam suatu upagraf dengan upagraf lainnya sekaligus memeriksa ukuran upagraf (nilai *k*).

Akan tetapi, algoritma ini bukanlah algoritma yang mangkus dalam permaslahan pencarian clique. Algoritma back-tracking sangat bergantung pada struktur dari graf, meskipun demikian algoritma ini memiliki kasus terburuk $O(n^k)$. Oleh karena itu algoritma ini merupakan algoritma eksponensial dan merupakan NP-complete.

Algoritma Bron-Kerbosch dapat menemukan semua *clique* dalam suatu graf tak berarah.

REFERENSI

[1] Alon, Noga, dkk,*Finding a large hidden clique in a random graph*.Tel Aviv University, 1998.

[2] Michaela Regneri, dkk, *Finding All Cliques of an Undirected Graph*.2007.

[3] Munir, Rinaldi, *Diktat Kuliah IF2251 Strategi Algoritmik*, ITB, 2007.

[4] *http://en.wikipedia.org/wiki/Clique_(graph_theory)*. Tanggal 7 Desember 2010, Pukul 15.45 WIB.

[5] *http://en.wikipedia.org/wiki/Clique_problem*. Tanggal 7 Desember 2010, Pukul 16.00 WIB.

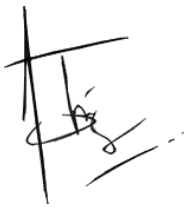
[6] *http://www.ibluemojo.com/school/clique_algorithm.html*. Tanggal 7 Desember 2010, Pukul 16.00 WIB.

[7] *http://www.dfki.de/~neumann/ie-seminar/presentations/finding_cliques.pdf* Tanggal 7 Desember 2010, Pukul 16.10 WIB.

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 2010



Adventus W. L Tobing
13505112