ISSN: 2338-7394

# ALGORITMA *K-NEAREST NEIGHBOR* UNTUK MENDETEKSI *CLIQUE* DALAM SUATU *GRAPH*

Wiwin Apriani[1*)]

[1]Program Studi Pendidikan Matematika Universitas Almuslim Bireuen
[*)]Email: wiwina10@gmail.com

**ABSTRAK**

*Algoritma k-nearest neighbor dapat digunakan untuk mendeteksi clique dalam suatu graf. Adapun graf yang digunakan adalah suatu graf lengkap tak berarah, berbobot, dan tidak berbobot. Untuk graf berbobot, andaikan diberikan suatu graf lengkap G = (V,E) dengan $w_{ij}$ merupakan bobot pada tiap edge (i,j). Sehingga untuk mendeteksi clique dalam graf berbobot dapat menggunakan nilai jarak euclidean pada masing-masing verteks yang saling bertetanggaan. Adapun tujuannya mendeteksi clique dengan jumlah bobot maksimum. Selain itu, untuk permasalahan graf tidak berbobot, pencarian clique dapat menggunakan nilai bilangan biner. Jika $V_iV_j \neq 0$, berarti bahwa verteks i dan verteks j saling terhubung dan dapat dimasukkan kedalam clique. Akan tetapi, jika $V_iV_j = 0$, berarti bahwa verteks i dan verteks j tidak saling terhubung, sehingga verteks yang tidak terhubung tersebut tidak termasuk kedalam clique dan dapat diabaikan. Tujuan mendeteksi clique pada graf tak berbobot adalah memperoleh maksimum clique.*

*Kata kunci: Algoritma K-nearest neighbor, Clique, Graf berbobot, Graf tak berbobot*

## 1. LATAR BELAKANG

Banyak hal didunia yang dapat digambarkan dengan menggunakan suatu diagram yang terdiri atas sekumpulan titik, dan garis untuk menghubungkan titik tersebut. Sebagai contoh, andaikan titik-titik direpresentasikan sebagai orang, dan garis direpresentasikan sebagai tingkat pertemanan. Selain itu, juga dapat dilihat pada pusat-pusat komunikasi yang disimbolkan dengan titik, dan hubungan komunikasi disimbolkan sebagai garis yang menghubungkan tiap-tiap titik tersebut. Permasalahan tersebut dalam matematika dikenal sebagai konsep graf, karena dari titik-titik pada permasalahan tersebut dapat dilihat apakah dihubungkan dengan suatu garis ataukah tidak.

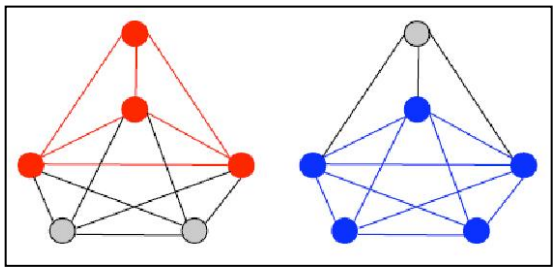
Permasalahan *clique* merupakan sebuah permasalahan graf *NP-complete*. *Clique* juga merupakan suatu subgraf lengkap yang diperoleh dari suatu graf yang terdiri atas tiga atau lebih titik (verteks). Setiap titik tersebut saling berdekatan satu sama lain dengan titik lainnya. Pencarian maksimum *clique* telah diterapkan dalam sejumlah bidang keilmuan, keahlian teknik dan bahkan dalam bidang bisnis. Beberapa penerapan *clique* bahkan mencapai bidang biologi meliputi solusi permasalahan struktur molekul DNA. *Clique* juga digunakan dalam pemrosesan citra dalam pengaturan jarak jauh, pencocokan titik koordinat dalam system informasi, permasalahan partisi data dalam kepingan memory dan lain-lain. Permasalahan ini juga merupakan suatu permasalahan pencarian solusi yang dianggap cukup sulit, sama seperti permasalahan knapsack, pewarnaan graf, sirkuit Hamilton, *n-puzzle* dan *Travelling Sales Person*.

*Clique* dapat digunakan untuk mengetahui daerah yang padat (sekumpulan unit) dalam suatu data multi-dimensional yang jarang. Sehingga untuk mempercepat laju proses dan peningkatan efektifitas terhadap pencarian *clique* yang berjumlah sangat besar, dapat diatasi dengan menggunakan suatu teknik penyederhanaan pada algoritma yang digunakan. Dengan demikian untuk menyelesaikan masalah tersebut, diperlukan suatu algoritma dan metode yang dapat diproses dan dimanfaatkan secara utuh untuk menyelesaikannya dengan mempertimbangkan waktu dan kualiatas yang dihasikan. Adapun permasalahan yang sering didapat dalam pencarian *clique* pada suatu objek yang berukuran sangat besar yaitu memerlukan waktu yang cukup lama untuk memprosesnya dan kemampuan memprosesnya terbatas. Adapun tujuan nya memperoleh maksimum *clique* pada suatu graf lengkap berbobot dan tidak berbobot dengan menggunakan algoritma *k-nearest neighbor*.

## 2. *CLIQUE*

Suatu subgraf $G = (V, E)$ adalah subset dari himpunan verteks $C \subseteq V$ sehingga setiap dua buah titik dalam $C$ terhubungkan dengan sebuah sisi. Subgraf maksimum adalah subgraf terbesar yang dapat terbentuk dari suatu graf. Jumlah titik

maksimum dari subgraf maksimum dinyatakan dalam $\omega(G)$. Salah satu contoh permasalahan dalam pencarian subgraf adalah pencarian subgraf minimum yang menghubungkan setiap titik dari suatu graf. Sebuah *clique* merupakan subgraf lengkap tak berarah dari suatu graf yang setiap titik saling terhubung satu sama lain. Dalam suatu graf akan terdapat sebuah *clique* jika dalam graf tersebut setidaknya terdapat sebuah subgraf lengkap berjumlah *k* buah simpul yang dapat berdiri sendiri (Regneri, 2007).



Gambar 1 Contoh *Clique* dalam Graf

Pada gambar di atas, diberikan dua buah contoh *clique* yang dapat dibentuk dari graf tersebut. Dalam suatu graf akan terdapat sebuah *clique* jika dalam graf tersebut terdapat setidaknya sebuah subgraf lengkap berjumlah k buah verteks yang dapat berdiri sendiri. Adapun jumlah maksimum *clique* yang mungkin diperoleh dari suatu graf *V* dengan ukuran *clique k* dapat dirumuskan sebagai berikut:

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

*Clique* merupakan suatu algoritma yang menggabungkan pengelompokan berdasarkan *grid* dan *density*. *Clique* adalah suatu metode yang tepat digunakan untuk data berdimensi tinggi dalam database yang besar. Metode ini membangun pengelompokan *multidimensional* dalam dua tahap yaitu: Pertama, data berdimensi *n* dipartisi kedalam unit-unit *rectangular* yang tidak *overlapping*. Identifikasi tentang kepadatan (*density*) dilakukan untuk setiap dimensi. Unit *density* tersebut kemudian diuji untuk menentukan *cluster*. Kedua, *clique* membuat deskripsi minimal untuk setiap *cluster*. Untuk setiap *cluster* ditentukan daerah maksimum yang meliputi *cluster–cluster* dari unit *density* yang tersambung (Cavique et al, 2002).

## 3. ALGORITMA *K-NEAREST NEIGHBOR*

Algortima *k-nearest neighbor* merupakan sebuah metode yang dapat digunakan untuk melakukan klasifikasi terhadap objek baru berdasarkan *k* tetangga terdekatnya. Algoritma ini termasuk algoritma *supervised learning*, dimana hasil dari contoh *query* yang baru, diklasifikasikan berdasarkan mayoritas dari kategori pada *k-nearest neighbor* tersebut. Kelompok yang paling banyak muncul yang akan menjadi kelompok hasil klasifikasi. Adapun kelebihan dari algoritma ini adalah lebih efektif digunakan untuk data yang besar, dan dapat menghasilkan data yang lebih akurat. Algoritma ini juga memiliki kekurangan yaitu harus menentukan nilai *k* yang paling optimal yang menyatakan jumlah tetangga terdekat. Selain itu, algoritma ini juga membutuhkan biaya komputasi cukup tinggi karena perhitungan jarak harus dilakukan pada setiap contoh *query* bersama-sama dengan seluruh contoh dari data *training* tersebut (Gorunescu, 2011).

Adapun tujuan dari algoritma ini adalah mengklasifikasikan suatu objek baru berdasarkan atribut dan *training* sampel. Proses klasifikasi tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Andaikan diberikan titik *query*, selanjutnya akan ditemukan sejumlah *k* objek atau data *training* yang paling dekat dengan titik *query*. Klasifikasi dilakukan dengan menggunakan *voting* terbanyak diantara klasifikasi dari *k* objek. Algoritma ini menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari contoh *query* yang baru.

Algoritma *k-nearest neighbor* merupakan salah satu algoritma yang sangat sederhana, yang bekerja berdasarkan jarak terpendek dari contoh *query* data *training* untuk menentukan *k-nearest neighbor*. Data *training* diproyeksikan keruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data tersebut. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data *training*. Sebuah titik pada ruang ini ditandai dengan kelas *c*, dimana kelas *c* merupakan klasifikasi yang paling banyak ditemui pada *k* buah tetangga terdekat dari titik tersebut. Dekat atau jauhnya tetangga biasanya dihitung berdasarkan jarak *euclide* yang direpresentasikan sebagai berikut:

$$D(a,b) = \sqrt{\sum_{k=1}^{d} (a_k - b_k)^2}$$

dimana matriks *D(a,b)* adalah jarak skalar dari kedua vektor *a* dan *b* pada matriks dengan ukuran *d* dimensi. Pada *fase training*, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data *training*. Pada *fase* klasifikasi, fitur-fitur yang sama dihitung untuk *testing* data (yang klasifikasinya tidak diketahui). Kemudian jarak dari vektor baru terhadap seluruh vektor *training sample* dihitung dan sejumlah *k* buah verteks yang

paling dekat diambil. Verteks yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari verteks-verteks tersebut. Secara umum algoritma *k-nearest neighbor* dapat dibuat sebagai berikut:

1. Menentukan nilai parameter *k* (jumlah tetangga yang palig dekat)
2. Menghitung jarak *euclidean* pada masing-masing objek terhadap data sampel yang diberikan
3. Mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak *euclid* terkecil
4. Mengumpulkan kategori *Y* (klasifikasi *nearest neighbor*)
5. Dengan menggunakan kategori mayoritas, maka dapat diprediksikan nilai contoh *query* yang telah dihitung

### 3.1 Algoritma *K-nearest neighbor* untuk Mendeteksi Maksimum *Clique* pada Graf Tak Berbobot

Algoritma merupakan suatu prosedur komputasi yang terdifinisi dengan baik dengan mengambil beberapa nilai atau seperangkat nilai sebagai masukan *input* dan menghasilkan beberapa nilai atau separangkat nilai sebagai keluaran *output*. Sebuah algoritma juga merupakan suatu langkah komputasi yang mengubah *input* menjadi *output*. Dengan demikian, algoritma dikatakan sebagai suatu urutan langkah-langkah logis dan sistematis untuk menyelesaikan suatu permasalahan. Adapun permasalahan *clique* dalam suatu graf dapat diselesaikan dengan menggunakan salah satu algoritma pencarian yaitu algoritma *k-nearest neighbor*.

Diberikan suatu graf *G* dengan *n* buah verteks, dimana $|V| = n$, dan setiap pasang verteks dihubungkan dengan sebuah *edge*. suatu graf $G^C$ disebut sebagai komplemen dari graf *G* dengan memiliki jumlah verteks yang sama pada *G*. Akan tetapi *u,v* merupakan *edge* dalam $G^C$ jika dan hanya jika *u,v* bukan anggota *edge* dalam *G*. Jika sepasang verteks *u,v* terhubung oleh suatu *edge* dalam *G*, maka *u* disebut sebagai tetangga *v* dan dapat dinyatakan sebagai $uv \in$.E Dengan demikian *neighborhood* dapat dikatakan sebagai suatu hubungan simetris dari *u* dan *v*, dimana $uv \epsilon E$ jika dan hanya jika $vu \epsilon E$. Derajat dari verteks *V* dinotasikan dengan *d(v)*, yang merupakan jumlah tetangga *v*. Derajat minimum pada semua verteks dalam *G* dinotasikan dengan $\theta$.

Suatu matriks *adjacency* dari *G* merupakan suatu matriks *n x n* dengan entri bari *u* dan kolom *v*, akan bernilai 1 jika $uv \epsilon E$ dan bernilai 0 jika sebaliknya. *Clique Q* dari *G* merupakan suatu himpunan verteks, dimana setiap verteks saling terhubung satu sama lain. Jika diberikan suatu *clique Q* dalam *G*, dan terdapat suatu verteks *v* diluar *Q*, sehingga dapat dikatakan bahwa *v adjoinable* jika terdapat suatu himpunan $Q \cup v$ yang merupakan *clique* pada *G*. $\rho(Q)$ dinotasikan sebagai jumlah verteks *adjoinable* pada *clique Q* dalam *G*. Pada masalah maksimum *clique*, tidak ditemukan suatu verteks *adjoinable* dalam graf yang mengandung *clique* maksimum tersebut.

Untuk mendeteksi maksimum *clique* dalam suatu graf tidak berbobot dengan menggunakan algoritma *k-neareast neighbor* dapat dilakukan dengan mengikuti dua prosedur berikut:

**Prosedur 1**: Diberikan suatu graf sederhana *G* dengan *n* buah verteks, dan diberikan suatu *clique Q* pada *G*, jika *Q* tidak memiliki verteks *adjoinable*, maka *clique* yang diperoleh adalah *clique Q* itu sendiri. Sebaliknya, untuk setiap verteks *adjoinable* pada *Q*, diperoleh $\rho(Q \cup v)$ dari verteks *adjoinablea* pada *clique* $Q \cup$ v. Andaikan $v_{maks}$ dinotasikan sebagai verteks *adjoinable*, sedemikian sehingga $\rho(Q \cup v_{maks})$ bernilai maksimum dan diperoleh *clique* $Q \cup v_{maks}$. Pencarian tersebut diulangi sampai tidak terdapat lagi verteks yang *adjoinable*.

**Prosedur 2**: Diberikan suatu graf sederhana *G* dengan *n* buah verteks, dan diberikan suatu maksimum *clique Q* pada *G*. Jika tidak terdapat verteks *v* diluar *Q*, sedemikian sehingga terdapat tepat satu verteks *w* dalam *Q* yang bukan tetangga *v*, maka diperoleh *output clique Q* itu sendiri. Sebaliknya, jika terdapat suatu verteks *v* diluar *Q*, sedemikian sehingga terdapat tepat satu verteks *w* dalam *Q* yang bukan tetangga dari *v*. Sehingga *Q (v,w)* didefinisikan sebagai tetangga *v* pada *Q* dan *w* dapat diabaikan dari *Q*. Selanjutnya ikuti prosedur 1 dan diperoleh hasil maksimum *clique* pada graf yang diinginkan.

Suatu graf *G* dengan *n* buah verteks disimbolkan dengan *1, 2, ..., n*, akan dicari sebuah *clique* yang berukuran paling sedikit *k*. Pada setiap langkah, jika diperoleh *clique* berukuran paling sedikit *k*, maka proses pencarian berhenti. Adapun langkah-langkah pencarian yang dilakukan adalah:

I. Untuk $i = 1, 2, ..., n$
   Inisialkan *clique* $Q_i = \{i\}$
   Lakukan prosedur 1 pada $Q_i$
   Untuk $r = 1, 2, ..., k$, lakukan prosedur 2, dan ulangi sampai *r* kali
   Diperoleh hasil berupa maksimum *clique* $Q_i$

II. Untuk tiap pasang maksimum *clique* $Q_i$, $Q_j$ yang diperoleh pada bagian I,
   Inisialkan *clique* $Q_{\{i,j\}} = Q_i Q_j$

Lakukan prosedur 1 pada $Q_{\{i,j\}}$
Untuk $r = 1, 2, ..., k$, lakukan prosedur 2, dan ulangi sampai $r$ kali
Diperoleh hasil berupa maksimum *clique* $Q_{\{i,j\}}$

### 3.2 Algoritma *K-nearest neighbor* untuk Mendeteksi Maksimum *Clique* pada Graf Berbobot

Terdapat tiga hal penting dalam membangun sebuah algoritma untuk masalah maksimum *clique* pada graf berbobot, yaitu penjumlahan batas atas, mencari solusi layak pada batas bawah, dan pemilihan variabel percabangan. Andaikan $G$ merupakan suatu graf lengkap, $u_i$ merupakan bobot pada node $i$, dan $w_{\{ij\}}$ merupakan bobot pada edge $(i,j)$, sehingga subproblem yang terdiri dari suatu himpunan $S$ dari node harus termasuk kedalam kelompok dan himpunan $U$ dari node dapat termasuk kedalam kelompok atau tidak termasuk.

Secara umum, untuk menyelesaikan permasalahan tersebut dilakukan pada setiap langkah dari tiap node terbaik yang berada didalam kelompok, tidak berada didalam kelompok, atau samar-samar. Node samar-samar dipilih sebagai variabel percabangan dan dua masalah baru yang dibuat, yaitu node berada didalam kelompok dan node tidak berada didalam kelompok. Jika batas atas subproblem memiliki bobot kelompok tidak lebih baik, maka subproblem dapat digunakan karena solusi yang diperoleh lebih baik. Jika batas bawah merupakan solusi yang layak, dan itu lebih baik dari pada kelompok terbaik yang diperoleh, maka kelompok terbaik yang ditemukan diganti dengan solusi batas bawah.

### 3.3 Mendeteksi *Clique* dalam suatu Graf tak berbobot

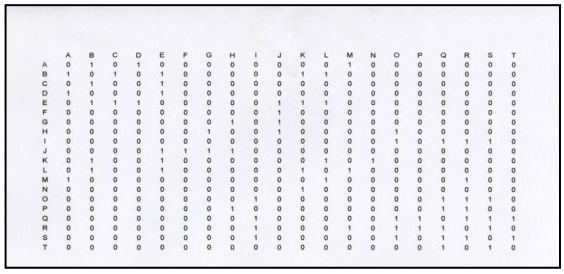
*Clique* dalam suatu graf dapat dicari dengan menggunakan algoritma-algoritma pencarian seperti algoritma *brute-force*, *back-tracking*, *bron kerbosch*, *k-nearest neighbor*, dan lain sebagainya. Dalam kasus ini akan menggunakan algoritma *k-nearest neighbor* untuk mendapatkan *clique* yang berukuran $k$ dalam suatu graf lengkap $G = (V,E)$ dengan $n$ buah verteks yang dimiliki. Oleh karena itu, pencarian *clique* pada graf tak berbobot dilakukan dengan mengecek setiap verteks pada graf tersebut. Jika setiap verteks saling terhubung satu sama lain, maka verteks-verteks yang saling terhubung tersebut akan membentuk sebuah *clique* yang berukuran $k$. Akan tetapi, jika verteks-verteks tersebut tidak terhubung dan tidak membentuk sebuah *clique* berukuran $k$, maka verteks tersebut akan diabaikan.

Pencarian *clique* yang berukuran $k$ verteks dengan menggunakan algoritma *k-nearest neighbor* dapat dilakukan dengan mengecek hubungan pada masing-masing verteks yang bertetanggaan. Selanjutnya dilakukan pemeriksaan terhadap verteks-verteks yang saling bertetanggaan. Jika $V_i$ dan $V_j \neq 0$, untuk $i = 1, 2, ..., k$ dan $j = 1, 2, ..., k$ maka verteks $i$ dan verteks $j$ saling terhubung. Namun sebaliknya jika $V_i$ dan $V_j = 0$, maka verteks $i$ dan verteks $j$ tidak terhubung dan dapat diabaikan. Adapun langkah-langkah untuk mencari *clique* dalam suatu graf tak berbobot dengan menggunakan algoritma *k-nearest neighbor* adalah sebagai berikut:

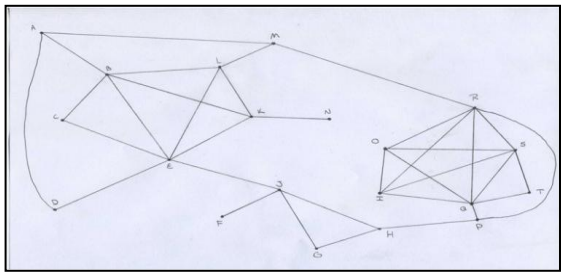
1. Buat suatu graf lengkap tak berarah $G = (V,E)$ dengan $n$ buah verteks (misal $V$=20),
2. Tentukan ukuran *clique* ($k$) yang akan dicari (misal $k$=3),
3. Tentukan verteks awal untuk memulai pencarian (misal $V_i$),
4. Lakukan Pengecekan terhadap verteks-verteks yang bertetanggaan dengan verteks awal ($V_i$ dan $V_j$),
5. Jika $V_i$ dan $V_j \neq 0$, maka pencarian dilanjutkan ke $V_j$ selanjutnya,
6. Jika $V_i$ dan $V_j = 0$, maka pencarian tidak dilanjutkan dan $V_j$ dapat diabaikan, kemudian kembali kelangkah 4,
7. Jika semua verteks telah dilakukan pengecekan, maka proses selesai dan diperoleh suatu *clique* yang berukuran k.

Sebagi contoh misalkan terdapat 20 desa disuatu kecamatan. Desa-desa tersebut dapat dikatakan terhubung jika antar desa tersebut terdapat suatu jalan. Setelah dilakukan pencatatan, diperoleh data desa-desa yang langsung terhubung dan desa-desa yang tidak langsung terhubung sebagai berikut:
Nama Desa: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T.
Desa yang langsung terhubung: AB, AM, AD, BL, BK, BE, BC, CE, DE, EJ, EL, EK, FJ, GH, GJ, HJ, HP, IO, IR, IS, IQ, KN, KL, LM, MR, OQ, OS, OR, PQ, PR, QR, QS, QT, RS, ST.
Untuk mendeteksi apakah dalam suatu graf terdapat *clique* atau tidak, maka permasalahan diatas akan disajikan dalam bentuk matriks adjacency yang berukuran $n$ x $n$. Bernilai 1 jika diperoleh bahwa desa langsung terhubung, dan bernilai 0 jika tidak langsung terhubung. Adapun matriks *Adjacency* yang berukuran 20 x 20 seperti terlihat pada Gambar 2.

Berdasarkan matriks *adjacency* pada Gambar 2, dapat digambarkan ke dalam suatu bentuk graf dengan kriteria bahwa bernilai 1 berarti terhubung, dan 0 berarti tidak terhubung. Adapun graf yang dibentuk dari matriks tersebut adalah seperti yang terlihat pada Gambar 3.



Gambar 2 Matriks *Adjacency* berukuran 20 x 20



Gambar 3 Graf desa-desa yang terhubung dari matriks *adjacency*

Berdasarkan Gambar 3, dapat dicari verteks-verteks yang saling terhubung dan membentuk *clique*. Adapun verteks-verteks yang membentuk *clique* berukuran *k*=3 adalah: verteks BCE, GHJ, QST, PRQ, BEK, BKL, IOQ, OQS, QRS, ORS, IRS, IQS, OQR, dan IOR. Selain terdapat *clique* berukuran 3, juga terdapat *clique* berukuran 4 dan 5. Secara berturut-turut dapat dilihat *clique* dengan ukuran *k*=4 adalah verteks BLKE, dan *clique* berukuran *k*=5 adalah verteks IOQRS.

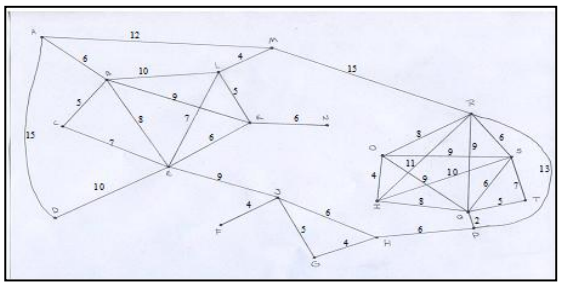
### 3.4 Mendeteksi *Clique* dalam suatu Graf berbobot

Pada masalah graf berbobot, proses pencarian dilakukan hampir sama seperti pada pencarian graf tidak berbobot. Andaikan $w_{ij}$ merupakan bobot pada edge $(i,j) \in$ E dalam suatu graf, maka pencarian *clique* untuk graf berbobot dapat menggunakan perhitungan jarak euclid pada tetangga terdekat. Adapun tujuan pencarian yaitu memperoleh *clique* dengan jumlah bobot yang maksimum pada graf tersebut.

Proses pencarian *clique* pada graf berbobot dilakukan dengan menggunakan algoritma *k-nearest neighbor*. Adapun langkah-langkah yang dilakukan adalah sebagai berikut:

1. Buat suatu graf lengkap tak berarah $G=(V,E)$ dengan *n* buah verteks, dan menentukan nilai $w_{ij}$ yang merupakan bobot pada edge $(i,j) \in$ E.
2. Menentukan nilai parameter *k* (jumlah tetangga yang palig dekat)
3. Menghitung jarak euclidean pada masing-masing objek terhadap data sampel yang diberikan dengan menggunakan persamaan 3.3 sesuai bobot yang dimiliki pada masing-masing edge dalam graf tersebut
4. Mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak euclid terkecil
5. Mengumpulkan kategori *Y* (klasifikasi *nearest neighbor*)
6. Menghitung bobot pada masing-masing kategori yang diperoleh
7. Menentukan ukuran maksimum hasil kategori yang juga merupakan *clique* dengan bobot maksimum

Sebagai contoh misalnya menggunakan graf yang terdapat pada kasus graf tidak berbobot, kemudian diberi jarak pada tiap edge, sehingga diperoleh graf dengan nilai bobot $w_{ij}$ sebagaimana terdapat pada Gambar 4.



Gambar 4 Graf berbobot

Berdasarkan Gambar 4 tersebut, dapat dicari verteks-verteks yang saling terhubung dan membentuk *clique*, dan kemudian dihitung jumlah bobot yang dimiliki. Adapun verteks-verteks yang membentuk *clique* berukuran *k*=3 dengan bobot yang dimiliki adalah: verteks BCE = 20 , GHJ = 15, QST = 18, PRQ = 24, BEK = 23, BKL = 24, IOQ = 21, OQS = 24, QRS = 21, ORS = 23, IRS = 27, IQS = 24, OQR = 26, dan IOR = 23. Selain terdapat *clique* berukuran 3, juga terdapat *clique* berukuran 4 dan 5. Secara berturut-turut dapat dilihat *clique* dengan ukuran *k*=4 adalah verteks BLKE = 46, dan *clique* berukuran *k*=5 adalah verteks IOQRS = 89.

## 4. KESIMPULAN

Dari hasil yang diperoleh dapat disimpulkan bahwa pada proses pencarian, hasil yang diperoleh bahwa maksimum *clique* terdapat pada k=5, yang berarti bahwa terdapat *clique* dengan jumlah verteks paling banyak 5 buah. Selanjutnya pada kasus graf berbobot, jumlah bobot maksimum diperoleh pada *clique* maksimum. Hal ini dikarenakan pada *clique* maksimum mengandung jumlah verteks dan edge yang paling banyak,

ISSN: 2338-7394

sehingga secara otomatis akan diperoleh bobot yang maksimum. Dengan demikian bobot maksimum akan terdapat pada maksimum *clique*.

## DAFTAR PUSTAKA


Brualdi, R.A., dan Ryser H.J. (1991). *Combinatorial Matrix Theory*. Cambridge University Press, Cambridge.

Chopra, S. dan Rao, M. R. (1991). *On the multiway cut polyhedron*. Networks 21, 51-89

Chopra, S. dan Rao, M. R. (1993). *The partition problem*. Mathematical programming 59, 87-116.

Cavique, L., Rego. dan Themido, I. (2002). *A Scatter Search Algorithm for Maximum Clique Problem*. in:Essays and Surveys in Metahauristics. Kluwer Academic Publishers. 227-244

Gorunescu, F. (2011) *Data Mining*: Concepts, Models and Techniques. Springer.

Grotschel, M. dan Wakabayashi, Y. (1989). *A cutting plane algorithm for a clustering problem*. Mathematical Programming Series B 45, 59-96.

Mehrotra, A.,dan Trick, A. M. (1998). *Clique*s and clustering: A combinatorial approach. Elsevier. Operation Research letters 1-12

Palla, G., Derenyi, I., Farkas, I., dan Vicsek, T. (2005). *Uncovering to overlappingcommunity structure of complex network in nature and society*. Nature 435 (70043): 814-818

Regneri, M. (2007). *Finding all cliques of an undirected graph*. Current trends in IE WS 06/07


## Penulis:


***Wiwin Apriani***
Menyelesaikan Pendidikan Magister Matematika di Universitas Sumatera Utara. Saat ini bertugas sebagai dosen di Prodi Pendidikan Matematika Universitas Almuslim.