ISBN: 978-979-792-552-9

Penggunaan Algoritma Kruskal yang Diperluas untuk Mencari Semua *Minimum Spanning Tree* Tanpa Konstren dari Suatu Graf

Narwen, Budi Rudianto

Jurusan Matematika, Universitas Andalas, Padang, Indonesia

narwen@fmipa.unand.ac.id

ABSTRAK

Ada beberapa metode untuk menentukan semua *minimum spanning tree* pada graf terhubung dengan pembobotan. Salah satu metode adalah menggunakan algoritma kruskal yang diperluas. Algoritma Kruskal hanya dapat menentukan satu bentuk *minimum spanning tree* saja. Metode ini diperluas dengan cara menukar salah satu sisi pada *minimum spanning tree* dengan sisi lain pada graf tetapi tidak masuk pada *minimum spanning tree* yang bobotnya sama. Bila hasil penukaran sisi tersebut tidak membentuk *cycle* dengan sisi lain pada *minimum spanning tree*, maka akan terbentuk *minimum spanning tree* yang baru dengan satu sisi yang berbeda. Akan tetapi bila membentuk *cycle*, maka sisi tersebut tidak membentuk *minimum spanning tree*. Hal ini dilakukan untuk semua sisi yang bobotnya sama. Metode yang dilakukan ini disebut dengan Algortima kruskal yang diperluas.

Kata kunci: *Spanning tree*, algoritma Kruskal, pergantian sisi, *minimum spanning trees*, graf dengan pembobotan

1 Pendahuluan

Misalkan diberikan sebuah graf terhubung sederhana. Menggunakan metoda tertentu dapat ditentukan apakah *sisi-sisi* (*edges*) dari graf itu membentuk *spanning tree*. Bila setiap sisi ditetapkan bernilai real, maka graf yang terbentuk dinamakan *graf dengan pembobotan* (*weighted graphs*). Masalah klasik pada graf dengan pembobotan adalah mencari *spanning tree* dengan total pejumlahan bobot sisinya minimum. Tree demikian disebut *minimum spanning tree* (***mst***). Semua sisi dilibatkan dan tidak ada yang dipertahankan dalam menentukan semua mst, maka mst yang dihasilkan disebut dengan mst tanpa konstren.

Mst demikian dapat dikaitkan dengan membangun jaringan pipa baru, sehingga diperoleh panjang total pipa atau biaya untuk membangun jaringan pipa tersebut adalah minimum.

ISBN: 978-979-792-552-9

Tulisan ini akan membahas sebuah metoda untuk menentukan semua *mst* tanpa konstren yang mungkin dapat dibangun. Metodanya didasarkan pada *algoritma Kruskal* dan dikembangkan dengan mencari komponen graf yang saling bebas dari setiap kelas ekivalen yang diberikan. Selanjutnya akan diperiksa keterhubungan dari masing-masing komponen tersebut. Terakhir dengan metode ini akan dibangun suatu algoritma untuk mendapatkan semua mst tanpa konstren yang mungkin ada dalam graf itu.

2 Metodologi

Misalkan diberikan Graf $G=(V, E)$ terdiri dari himpunan dari objek-objek $V=\{v_1, v_2, v_3, \ldots\}$, $V \neq \phi$, yang disebut ***titik-titik*** (*vertices*) dan himpunan lainnya E yang elemen-elemennya disebut ***sisi-sisi*** (*edges*). Himpunan titik dan sisi dari G masing-masing dinotasikan dengan $V(G)$ dan $E(G)$.

Sisi dengan satu titik yang sama sebagai titik ujungnya disebut ***loop***. Sisi yang bersesuaian dengan lebih dari satu pasangan titik disebut ***sisi paralel***. ***Graf sederhana*** tidak mempunyai loop dan sisi paralel.

Jalan (*walk*) dari v_0 ke v_n di G adalah barisan hingga $v_0, e_{01}, v_1, \ldots, e_{(n-1)n}, v_n$ dari titik-titik dan sisi-sisi di G sehingga $e_{(i-1)i} = (v_{i-1}, v_i)$ adalah sisi di G. Jalan dikatakan *tertutup* bila $v_0 = v_n$ dan terbuka bila $v_0 \neq v_n$. Jalan terbuka yang semua titiknya berbeda disebut ***Lintasan*** (*path*), sedangkan ***cycle*** (*circuit*) adalah jalan tertutup yang semua titiknya berbeda.

Graf G dikatakan ***terhubung*** jika untuk setiap dua titik di G ada lintasan yang menghubungkan kedua titik tersebut. ***Tree*** adalah graf terhubung yang tidak mengandung *cycle*. Tree T dikatakan ***spanning tree*** dari graf terhubung G jika T *subgraf* dari G dan T memuat semua titik G.

Graf G adalah graf dengan pembobotan, bila ada *fungsi bobot* w bernilai real pada sisi-sisi G. Jadi $w(e_{ij})$ adalah bobot sisi antara titik ke-i dengan titik ke-j. *Bobot spanning tree* T pada graf G adalah penjumlahan dari semua bobot sisi yang membangun T. Diantara semua *spanning tree* tersebut pasti ada yang mempunyai penjumlahan bobot sisi terkecil. *Spanning tree* ini disebut dengan ***mst***. Untuk selanjutnya, graf G yang dimaksud adalah graf sederhana dengan pembobotan.

Ada banyak algoritma untuk mencari sebuah *mst* dalam graf G, salah satunya adalah *algoritma Kruskal*. Algoritma Kruskal bekerja dengan *mendaftarkan semua sisi graf G berurutan dari bobot yang terkecil ke yang terbesar. Selanjutnya pilih bobot paling kecil di G. Maka untuk setiap langkah selanjutnya berturut-turut pilih sisi yang bobotnya paling kecil lainnya, tetapi sisi tersebut tidak membentuk cycle dengan sisi-sisi yang telah dipilih sebelumnya. Proses ini diteruskan sampai tidak ada lagi sisi yang tersisa. Dengan demikian, sisi-sisi yang telah dipilih akan membentuk sebuah mst di G.*

Masalah selanjutnya adalah berapa banyak mst yang dapat dibentuk bila sebarang graf G diberikan. Dengan menggunakan algoritma Kruskal yang diperluas.

Misalkan T adalah mst, $f \in E(G)$ tetapi $f \notin T$. Definisikan ***P(f, T)*** sebagai *lintasan sederhana di dalam T yang menghu bungkan titik-titik dari f*. Perhatikan bahwa $P(f, T) \cup \{f\}$ akan membentuk sebuah cycle di T. Misalkan ada sisi $e \in P(f, T)$ dimana $w(e) = w(f)$. Kemudian ganti sisi $e \in T$ dengan sisi f untuk mendapatkan *mst* lainnya yang

ISBN: 978-979-792-552-9

berbeda dengan T, yaitu $S = (T - \{e\}) \cup \{f\}$. Proses ini disebut dengan ***penggantian sisi yang bobotnya sama*** di dalam T. Jadi, S diperoleh dari T dengan mengganti sisi T yang bobotnya sama dengan sisi lain di S tapi tidak di T.

Dengan menggunakan penggantian sisi yang bobotnya sama, maka diperoleh teorema berikut.

Teorema 1 *Jika T dan S adalah mst dalam graf G dengan pembobotan, maka ada sebuah barisan hingga dari pergantian sisi yang bobotnya sama dimana pergantian sisi tersebut dimulai dari T dan berakhir sampai S.*

Akibatnya, jika semua sisi dari graf terhubung G mempunyai bobot berbeda, maka G hanya mempunyai satu mst. Dalam hal ini, setiap mst pada G dapat diperoleh sebagai hasil output algoritma Kruskal.

3 Hasil dan Pembahasan

Sebelum mencari semua mst yang ada dalam graf G, terlebih dahulu ditentukan sebarang mst yang diperoleh dengan algoritma Kruskal. Mst ini disebut dengan ***tree referensi***, dinotasikan dengan $\mathfrak{I}$.

Sisi-sisi pada graf G, diantaranya ada yang masuk ke dalam *tree referensi* $\mathfrak{I}$ dan ada yang tidak. Dengan sifat berikut dapat ditentukan sisi-sisi mana yang dapat membentuk *mst* lainnya.

Teorema 2 *Sebuah sisi e dari G termasuk ke mst*
$\Leftrightarrow e \in \mathfrak{I}$ *atau* $\exists$ *sisi* $f \in P(e, \mathfrak{I})$, *dimana* $w(f) = w(e)$.

Bukti.

($\rightarrow$) Misal T sebarang *mst* di G, $e \in E(G)$, $e \in T$ dan $\mathfrak{I}$ tree referensi. Jika e tidak dilibatkan dalam penggantian sisi, maka $e \in \mathfrak{I}$. Jika e dilibatkan dalam penggantian sisi, maka terdapat penggantian sisi yang bobotnya sama dari T ke $\mathfrak{I}$. Ini berarti bahwa ada sisi $f \in P(e, \mathfrak{I})$ dimana $w(f) = w(e)$.

($\leftarrow$) Misal $e \in E(G)$ dan $\mathfrak{I}$ tree referensi dengan $e \in \mathfrak{I}$ atau ada sisi $f \in P(e, \mathfrak{I})$ dimana $w(f) = w(e)$. Jika $e \in \mathfrak{I}$, berarti e masuk ke *mst*. Jika ada sisi $f \in P(e, \mathfrak{I})$ dimana $w(f) = w(e)$, maka terdapat penggantian sisi yang bobotnya sama dari $\mathfrak{I}$ ke $T = (\mathfrak{I} - \{f\}) \cup \{e\}$. Akibatnya sisi $e \in T$. Jadi e termasuk ke *mst*. ■

Sisi-sisi yang tidak memenuhi syarat ini, yaitu sisi yang tidak masuk kepada tree referensi $\mathfrak{I}$ dan bobotnya lebih besar dari semua bobot sisi yang ada pada cycle yang dibentuk dengan sebagian sisi pada tree referensi, maka sisi demikian dapat dihapus dari graf G.

Setiap kali melakukan penggantian sisi yang bobotnya sama akan tetap mempertahankan banyaknya sisi dari kelas ekivalen. Hal tersebut dijamin oleh teorema berikut.

ISBN: 978-979-792-552-9

Teorema 3 *Semua mst akan memuat jumlah sisi yang sama dari sebarang kelas ekivalen yang diberikan.*

Bukti. Misal T dan S adalah dua mst sebarang pada graf G. Maka terdapat barisan penggantian sisi yang bobotnya sama dari *mst* T ke S. Setiap penggantian sisi akan mengganti sebuah sisi dengan sisi lainnya dalam kelas ekivalen yang sama. Akibatnya jumlah sisi yang ada di dalam kelas ekivalen tersebut akan tetap sama dengan jumlah sisi sebelum dilakukan penggantian. Ini berarti jumlah sisi dari sebarang kelas ekivalen akan tetap sama untuk semua *mst*. ■

Perhatikan bahwa sisi-sisi yang tidak masuk ke sebarang mst maka sisi tersebut dihapus dari graf G. Sebaliknya sisi yang masuk ke dalam sebarang *mst*, maka sisi itu tetap dipertahankan dalam graf G.

Misalkan g sisi yang tidak dihapus dari graf G dan g tidak masuk ke mst T. Kelas ekivalen g, dinotasikan dengan $X(g)$ atau $[g]$, didefinisikan *sebagai himpunan sisi-sisi yang terdiri dari sisi g dan semua sisi yang bobotnya sama dengan bobot g di P(g, T)*. Maka $X(g)$ mengandung g dan semua sisi yang dapat diganti oleh g dengan satu kali penggantian sisi yang bobotnya sama di dalam T.

Kombinasikan setiap pasangan himpunan $X(g)$ yang mempunyai sisi persekutuan bersama. Setiap kelas-kelas ekivalen yang telah dikombinasi kan akan membentuk satu kelas ekivalen yang disebut dengan ***kelas transitif*** dari himpunan kelas ekivalen $X(g)$. Sisi-sisi G yang tidak masuk ke salah satu $X(g)$ akan membentuk ***kelas ekivalen singelton***. Kelas singelton akan masuk ke setiap mst.

Setiap mst pada graf G mengandung banyaknya sisi dari sebarang kelas ekivalen $[e]$ yang diberikan tetap sama, tetapi tidak semua subset dari $[e]$ dengan jumlah sisinya demikian dapat membentuk sebuah mst.

Jadi, $s \subseteq [e]$ dan s ***subset terpilih dari*** **$[e]$** jika s adalah himpunan sisi di $[e]$ dan s juga dalam sebuah mst. Dengan kata lain, $s \subseteq [e]$ adalah *subset terpilih dari* $[e]$ jika ada sebuah mst T sehingga $T \cap [e] = s$.

Teorema 4 *Jika s_1 dan s_2 adalah dua subset terpilih dari [e] dan T sebuah mst sebarang dari G yang memuat s_1, maka $(T - s_1) \cup s_2$ juga sebuah mst.*

Bukti. Misal T' adalah *mst*, $s_1 \subseteq [e]$ dan $s_1 \subseteq T'$, maka terdapat barisan $T_0, T_1, \ldots, T_n$ dari pergantian sisi yang bobotnya sama dimana $T_0 = T'$ dan $T_n = T$. Asumsikan bahwa barisan ini dipilih sedemikian sehingga penggantian sisi yang melibatkan sisi-sisi $[e]$ diganti terakhir. Misalkan T_m spanning tree terakhir dalam barisan sebelum penggantian melibatkan sisi-sisi $[e]$ yang diganti. Maka sisi-sisi *mst* T_m = sisi T untuk semua sisi dari setiap kelas ekivalen selain $[e]$, dan T_m akan memuat $s_2 \subseteq [e]$ untuk menggantikan s_1. Jadi $T_m = (T - s_1) \cup s_2$**,** dimana T_m adalah *mst* yang diinginkan. ■

Teorema 4 menjamin subset terpilih dari suatu kelas ekivalen yang diberikan dapat ditukar, dan himpunan dari semua mst dalam graf G dapat dibentuk dengan cara mengambil satu kelas terpilih dari setiap kelas ekivalen yang diperoleh. Secara khusus

ISBN: 978-979-792-552-9

banyaknya mst dalam graf *G* adalah perkalian dari banyaknya subset terpilih dari setiap kelas ekivalen.

Dalam hal ini subset terpilih dari kelas ekivalen yang berbeda akan saling bebas, sehingga dapat ditentukan subset terpilih dari setiap kelas ekivalen, kemudian kombinasikan setiap subset terpilih tersebut dalam semua bentuk yang mungkin.

Berdasarkan hasil di atas, dapat disusun algoritma untuk menentukan semua mst tanpa konstren dalam graf G dengan mengembangkan algoritma Kruskal. Algoritma tersebut mempunyai tahapan sebagai berikut:

Langkah 1

Tentukan sebarang mst $\mathfrak{T}$ *sebagai tree referensi. Gunakan* $\mathfrak{T}$ *untuk menguji semua sisi dari graf terhubung G. Misalkan sisi* $e \in E(G)$. *Selidiki apakah* $e \in \mathfrak{T}$ *atau ada* $f \in P(e,\mathfrak{T})$ *dimana* $w(f) = w(e)$ *untuk sisi-sisi yang dapat dipilih. Kalau tidak demikian, maka sisi tersebut dihapus dari graf G, karena tidak akan membentuk minimal spanning tree.*

Langkah 2

Untuk setiap sisi g yang tidak di $\mathfrak{T}$, *tentukan X(g) dan hitung kelas-kelas transitifnya. Kelas transitif ini adalah kelas ekivalen. Jika* $e \in \mathfrak{T}$ *dan* $e \notin X(g)$, *maka diperoleh kelas singelton* $[e] = \{e\}$.

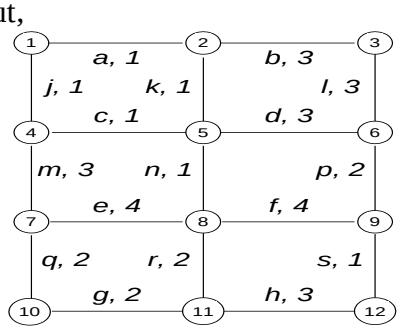
Langkah 3

Misalkan kelas ekivalen $[e]$ *dengan* $w(e) = k$ *dan H subgraf dari* $\mathfrak{T}$ *terdiri dari sisi-sisi yang bobotnya* $< k$ *dan banyaknya sisi* $[e]$ *dalam* $\mathfrak{T}$ *adalah* n_k. *Untuk setiap subset dengan* n_k *sisi pada* $[e]$, *tentukan apakah subset ini membentuk sebuah cycle dengan sisi-sisi H. Jika tidak membentuk cycle, maka subset ini akan manjadi subset terpilih dari* $[e]$. *Rute ini adalah jiwa dari algoritma Kruskal, dengan menguji cycle untuk setiap langkah.*

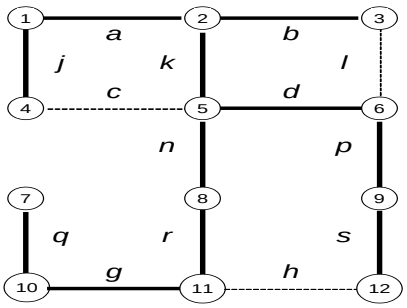
Langkah 4

Kombinasikan semua subset terpilih dari setiap kelas ekivalen $[e]$ *dengan mengalikan banyaknya subset terpilih dari setiap kelas ekivalen* $[e]$ *untuk mendapatkan semua mst yang ada dalam graf G.*

Sebagai contoh aplikasi, misalkan diberikan graf *G* dan tree referensi $\mathfrak{T}$ sebagai berikut,



Graf G



Tree referensi $\mathfrak{T}$

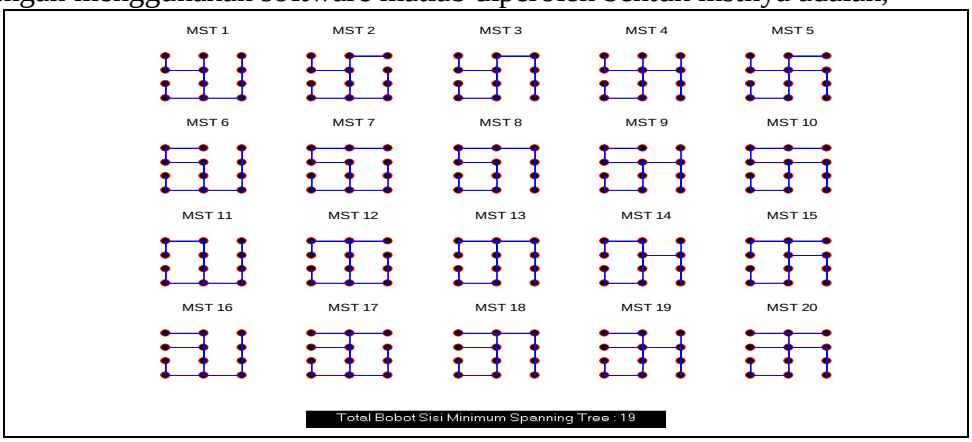
Dengan menggunakan algoritma di atas,

1. Misalkan $\mathfrak{T} = \{j, a, b, k, d, p, s, n, r, g, q\}$ adalah mst sebarang sebagai tree referensi. Uji semua sisi *G* pada $\mathfrak{T}$, maka sisi-sisi *e*, *f*, dan *m* tidak dipilih, sehingga sisi-sisi

ISBN: 978-979-792-552-9

tersebut dihapus dari G. Sedangkan sisi-sisi c, h dan l adalah sisi-sisi yang dapat dipilih (ditunjukkan oleh garis putus-putus).

2. Karena c, h dan l tidak di $\mathfrak{I}$, maka $X(c)=\{c, a, j, k\}$, $X(h)=\{h, d\}$ dan $X(l)=\{l, b, d\}$. Karena $X(h) \cap X(l) \neq \{\}$ maka $X(h)$ dan $X(l)$ membentuk kelas transitif. Dengan demikian kelas ekivalen yang bobotnya 1 adalah $[n]=\{n\}$, $[s]=\{s\}$ dan $[a]=\{a, c, j, k\}$. Kelas ekivalen yang bobotnya 2 adalah $[p]=\{p\}$, $[g]=\{g\}$, $[q]=\{q\}$ dan $[r]=\{r\}$. Kelas ekivalen yang bobotnya 3 adalah $[b]=\{b, d, h, l\}$.
3. Perhatikan untuk kelas singelton. Karena $n_k=1$ dan banyak anggota sisi di kelas itu juga 1, maka sisi tersebut dipilih ***1*** kali.
   Kelas $[a]$ mempunyai $n_k=3$ dan $w(a)=1$, sehingga H = {}. Subset-subset dari $[a]$ yaitu $\{a, c, j\}$, $\{a, c, k\}$, $\{a, j, k\}$ dan $\{c, j, k\}$ tidak membentuk cycle dengan H. Jadi sisi dari $[a]$ dipilih ***4*** kali.
   Kelas $[b]$ mempunyai $n_k=2$ sehingga H = $\{j, a, k, p, s, n, r, g, q\}$. Sehingga subset-subset dari $[b]$ yaitu $\{b, d\}$, $\{b, h\}$, $\{b, l\}$, $\{d, l\}$, $\{h, l\}$ masing-masing tidak membentuk cycle dengan H, sedangkan $\{d, h\}$ membentuk cycle dengan H. Jadi sisi dari $[b]$ dipilih ***5*** kali
4. Banyak mst tanpa konstren dalam graf G adalah $(1)(1)(4)(1)(1)(1)(1)(5)=20$. Dengan menggunakan software matlab diperoleh bentuk mstnya adalah,



Kesimpulan

Algoritma Kruskal yang diperluas dapat digunakan untuk menentukan berapa banyak mst tanpa konstren yang mungkin dapat dibentuk dalam suatu graf terhubung dengan pembobotan. Sedangkan dengan metode-metode yang sudah ada selama ini hanya dapat menentukan satu buah mst sebarang dalam graf tersebut. Semua mst ini diperoleh dengan melakukan pergantian sisi yang bobotnya sama dan tak satupun dari sisi-sisi tersebut yang dipertahankan di dalam setiap mst yang diperoleh. Menentukan semua mst tanpa konstren dapat dikaitkan dengan membangun jaringan baru dan mencari semua alternatif yang mungkin untuk membangun jaringan tersebut tetapi total biaya atau jaraknya tetap sama.

ISBN: 978-979-792-552-9

Daftar Pustaka

[1] Biggs N., *Algebraic Graph Theory, 2nd ed.*, CambridgeUniversity Press, New York, 1993.
[2] Deo N., *Graph Theory with Applications to Engineering and Computer Science*, Prentice Hall, New Delhi, 1986.
[3] Harary F., *Graph Theory*, Addison-Wesly Publishing, London, 1969.
[4] Hartsfield N., Ringel G., *Pearl in Graph Theory A Comprehensive Introduction, Revised and Augmented*, Academic Press, London, 1997.
[5] Narwen, *Suatu Metode untuk Menentukan Semua Minimum Spanning Tree dalam Suatu Graf*, Tesis Pasca Sarjana Matematika, ITB, 2002.
[6] Tucker A., *Applied Combinatorics, 3rd ed.*, John Willey & Sons, Inc., Canada, 1995.
[7] Wilson R. J., Watkins J. J., *Graphs An Introductory Approach*, John Willey & Sons, Inc., Canada, 1990.
[8] Wright P., Counting and Constructing Minimal Spanning Trees, *Bulletin of The Institute of Combinatorics and its Applications 21*, 1997, 65-76.
[9] Wright P., On Minimal Spanning Trees and Determinants, *Mathematics Magazine Vol. 73*, No.1, 2000, 21-28.