Penerapan Algoritma Steiner Tree dalam Konstruksi Jaringan Pipa Gas

Achmad Baihaqi, NIM: 13508030
Program Studi Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesa 10 Bandung
e-mail: baihaqi@students.itb.ac.id

ABSTRAK

***Steiner tree* merupakan suatu topik di dalam Teori Graf yang memiliki penggunaan luas dalam bidang perancangan jaringan yang efisien. Dalam makalah ini akan dijelaskan algoritma pencarian *steiner tree* pada graf untuk diimplementasikan dalam konstruksi jaringan pipa gas. Jaringan pipa gas yang akan dibahas yakni jaringan pipa gas dalam satu pulau. Algoritma yang digunakan dalam pengkostruksian jaringan ini adalah algoritma *heuristic* yang dirancang oleh Markowsky et al. Pemilihan algoritma *heuristic* didasarkan atas kecepatan waktu komputasinya jika dibandingkan dengan algoritma eksak yang bekerja secara *brute force* untuk memeriksa setiap kemungkinan minimal *spanning tree*. Bobot *Steiner tree* yang dihasilkan melalui algoritma *heuristic* belum tentu merupakan bobot yang minimum, akan tetapi bobot tersebut tidak akan melebihi suatu nilai batas atas. Apabila Steiner Tree diterapkan dalam pembuatan jaringan pipa gas di Indonesia, maka pendistribusian gas akan lebih optimal dalam hal biaya dan waktu.**

Kata kunci: graf, *tree*, *steiner tree*, *heuristic*, *spanning tree*, jaringan pipa gas.

1. PENDAHULUAN

Saat ini gas bumi merupakan salah satu kebutuhan penting bagi masyarakat Indonesia. Apalagi setelah muncul kebijakan pemerintah yaitu konversi minyak ke gas. Akan tetapi, penyaluran gas bumi yang dilakukan saat ini sebagian benar masih menggunakan metode konvensional, yaitu pengangkutan melalui transportasi darat dalam satu pulau. Pendistribusian gas melalui pipa juga telah dilakukan tetapi hanya sebagian kecil dalam Pulau Jawa. Penyaluran dengan sistem pipa memiliki beberapa kelebihan dibanding metode konvensional, diantaranya adalah waktu penyaluran yang relatif lebih singkat dan biaya pengangkutan yang ekonomis.[6]

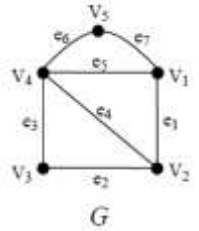
Untuk itu, diperlukan adanya suatu desain jaringan pipa gas yang menghubungkan semua kota dalam suatu pulau yang memiliki depo gas dengan panjang pipa yang seminimum mungkin. Algoritma yang digunakan untuk merancang desain jaringan ini didasarkan pada konsep *Steiner Tree*. Secara khusus, algoritma yang digunakan meruapakan aproksimasi *Steiner Tree* karena akan memberikan waktu perhitungan yang lebih cepat dibandingkan dengan algoritma *Steiner Tree* yang eksak. Walaupun hasilnya tidak seakurat algoritma eksak, metode aproksimasi *Steiner Tree* dapat menentukan *tree* hampiran untuk graf dengan banyak titik sekitar 100 buah dalam waktu kurang dari 1 detik.

Dalam algoritma *Steiner Tree*, diperlukan pula metode yang dapat menghitung jarak terdekat dari semua titik ke titik lain dalam suatu graf berbobot. Selain itu, hasil perhitungan pun sebaiknya tersimpan dalam matriks sehingga dapat mudah diolah. Sehubungan dengan itu, algoritma *Floyd-Warshall* yang akan digunakan untuk menghitung jarak terpendek antar dua titik. Selanjutnya, berkaitan dengan pemodelan masalah pencarian *tree* dengan bobot minimum tersebut, dalam satu pulau dimodelkan sebagai suatu graf dengan kota-kota sebagai titik, jalur penghubung antar kota sebagi sisi, dan panjang jalur penghubung tersebut merupakan bobot sisi graf.

2. TEORI DASAR

2.1. Graf

Graf didefinisikan sebagai sistem yang terdiri dari 2 komponen, yaitu himpunan tak kosong V(G) yang anggotanya disebut titik dan himpunan sisi E(G) yang berupa hinpunan pasangan tak terurut dari dua buah titik berbeda di V(G) [1]. Pada pembahasan ini diasumsikan bahwa setiap sisi pada E(G) tunggal. Dengan demikian, graf dalam pembahasan ini merupakan graf yang tidak memiliki sisi ganda. Jika e merupakan sebuah sisi, sedangkan $e = \{u,v\}$, maka *e* disebut sebagai pengait *u* dan *v*. Titik-titik *u* dan *v* disebut titik-titik ujung dari *e*. Sebagai ilustrasi tentang graf, perhatikan contoh berikut:



G

Gambar 1. Representasi Graf G

Dari graf G = (V(G),E(G)) di atas diperoleh:
V(G) = {v_1, v_2, v_3, v_4, v_5},
E(G) = {e_1, e_2, e_3, e_4, e_5, e_6, e_7}
= {{v_1,v_2}, {v_2,v_3}, {v_3,v_4}, {v_4,v_2}, {v_4,v_1}, {v_4,v_5}, {v_5, v_1}}.

Graf G dikatakan graf berbobot *(weighted graph)* jika setiap sisinya mempunyai sebuah nilai real *w(e)* yang disebut sebagai bobot.

2.2. Lintasan

2.2.1. Definisi Lintasan

Walk didefinisikan sebagai barisan selang seling titik dan sisi. Lintasan (*path*) adalah *walk* dengan titik-titik yang berbeda ($v_i \neq v_j$, untuk $i \neq j$). $u = v_0e_1v_1e_2...e_kv_k = v$ dikatakan sebuah lintasan dalam graf G jika setiap vi berbeda dan ei = {v_{i-1}, v_i}, untuk $1 \leq i \leq k$, dan panjang lintasan dari *u* ke *v* adalah *k* [1]. Sekarang, kita dapat mendefinisikan graf terhubung sebagai graf yang setiap dua titiknya dapat dihubungkan oleh suatu lintasan.

2.2.2. Lintasan Terpendek

Misalkan G suatu graf berbobot dengan *v*, $w \in V(G)$ dan $v \neq w$, lintasan terpendek dari *u* ke *w* merupakan lintasan dengan titik awal *v* dan titik akhir *w* dengan bobot yang minimum. Bobot dari lintasan tersebut disebut bobot minimum dari *u* ke *w*.

Selanjutnya, apabila $(m_{ij}) \in M$ merupakan bobot dari lintasan terpendek dari v_i ke v_j untuk setiap *i* dan *j* maka M disebut sebagai matriks bobot minimum.

2.3. Pohon

2.3.1. Definisi Pohon

Pohon (atau *tree*) didefinisikan sebagai graf terhubung yang tidak memuat lingkaran [1]. Dengan demikian, suatu lintasan merupakan salah satu contoh dari *tree*. Bobot dari *tree* didefinisikan sebagai jumlah seluruh bobot sisi pada *tree*.

2.3.2. Minimal Spanning Tree

Sebelum mendefinisikan *spanning tree*, perlu didefinisikan dahulu tentang subgraf. Graf H disebut sebagai subgraf dari graf G, ditulis $H \subseteq G$, jika $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$. Dalam kasus V(H) = V(G) maka subgraf tersebut disebut sebagai *spanning* subgraf. *Spanning tree* dari graf G merupakan *spanning* subgraf dari G yang berupa *tree*. Pada graf berbobot, minimal *spanning tree* merupakan *spanning tree* dengan bobot yang paling kecil [1].

2.3. Steiner Tree

Konsep *Steiner tree* pertama kali diperkenalkan oleh Jacob Steiner, salah satu penerapannya adalah dalam bidang perancangan sistem jaringan yang efisien.

2.4.1. Definisi Steiner Tree

Misalkan terdapat graf G dengan $W \subseteq V(G)$ dan fungsi bobot $f : E \rightarrow \mathbb{R}^+$ pada setiap sisinya. *Steiner tree* pada G atas W merupakan subgraf H = (W*,F) dari G yang merupakan *tree* dengan bobot minimum dan $W \subseteq W^*$. Dengan demikian, *Steiner tree* dapat dipandang sebagai minimal *spanning tree* pada G jika W = V(G). [3]

2.4.2. Titik Customer dan Titik Steiner

w disebut sebagai titik customer jika $w \in W$, sedangkan *s* disebut sebagai titik *Steiner* apabila $s \in W^* \backslash W$. [3]

3. ALGORITMA STEINER TREE

3.1. Penentuan Steiner Tree Secara Heuristik

Untuk membentuk *Steiner Tree* dari suatu graf dapat digunakan algoritma *heuristic Steiner tree* yang dirancang oleh *Markowsky et al*. Berikut ini merupakan algoritma heuristic tersebut[4] :

Algoritma Heuristic:
Masukan: Graf $G = (V, E)$ dengan fungsi bobot $f: E \rightarrow \mathbb{R}^+$ pada setiap sisinya, $W \subseteq V(G)$ sebagai titik customer.
Keluaran: *TH* sebagai *Steiner tree* dari *G* atas *W*.
Langkah-langkah:

1. Tentukan matriks bobot minimum M dan matriks lintasan terpendek L bagi *G*.
2. Konstruksi graf komplit $G1=(W, E_1)$ dengan $f(\{i, j\}) = (m_{ij})$ untuk $\{i, j\} \in E_1$.
3. Konstruksi minimal *spanning tree* T_1 pada G_1.
4. Ganti setiap sisi $\{i, j\}$pada T_1 dengan lintasan terpendek dari *i* ke *j* pada *G* berdasarkan L. Jika ada lebih dari satu lintasan terpendek, pilih salah satu secara sebarang. Hasilnya adalah subgraf G_2 dari *G*.
5. Konstruksi minimal *spanning tree* T_2 pada G_2.
6. Jika $v \notin W$, $v \in V(T_2)$, dan $d(v) = 1$ maka hapus *v* dari T_2 sehingga akan dihasilkan T_H yang merupakan *Steiner tree* hasil aproksimasi.

3.2. Batas Atas Bobot Steiner Tree Heuristik

Steiner tree yang dihasilkan dari algoritma *heuristic* hanyalah suatu aproksimasi. Akibatnya, bobot *Steiner* yang dihasilkan belum tentu merupakan bobot yang minimum. Walaupun demikian, dapat ditunjukkan bahwa bobot *Steiner tree heuristic* memiliki suatu batasan. Misalkan D_H bobot dari *Steiner tree* yang dihasilkan oleh

algoritma heuristic dan D_{MIN} bobot dari *Steiner tree* yang yang dihasilkan oleh algoritma eksak maka dapat ditentukan batas atas dari D_H / D_{MIN}. Sebelumnya, kita memerlukan lemma berikut:

Lemma 1:

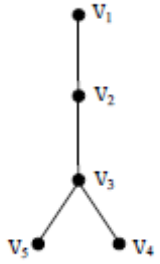
Misalkan *T* suatu *tree* dengan banyak sisi $m \geq 2$ maka akan terdapat sebuah *loop* di *T*, $u_0u_1u_2..u_{2m}$ dengan u_i, $1 \leq i \leq 2m$, merupakan titik di *T*, sehingga:

a. Setiap sisi di *T* akan muncul tepat dua kali pada *loop*.
b. Setiap titik berderajat 1 (*leaf*) di *T* akan muncul tepat sekali pada barisan $u_0u_1u_2..u_{2m}$ ($u_0 = u_{2m}$ dihitung sebagai satu kemunculan pada *loop*) dan jika u_i dan u_j dua buah *leaf* di dalam *loop*, tanpa ada *leaf* lain antar keduanya, maka $u_0u_{i+1}..u_j$ adalah lintasan[4].

Bukti:

Akan dibuktikan dengan induksi pada *m*. Untuk $m = 1$, misalkan $\{u_1, u_2\}$ himpunan titik di *T*. Diperoleh bahwa loop $u_1u_2u_1$ memenuhi (a) dan (b) pada lemma. Asumsikan untuk $m = k \geq 1$ *lemma* tersebut benar. Maka, untuk $m = k + 1$, misalkan v_p suatu *leaf* di *T* dan $\{v_p, v_q\}$ sisi yang terhubung dengan v_p. Pandang *tree T'* yang dikonstruksi dengan cara menghapus v_p dan $\{v_p, v_q\}$ pada *T*. Akibatnya berdasarkan hipotesis induksi, akan terdapat *loop* $u'_0u'_1..u'_{2k}$ di *T'* yang memenuhi (a) dan (b). Selanjutnya, dengan mengganti v_q pada *loop* tersebut dengan $\{v_q, v_p, v_q\}$ pada kemunculan pertamanya akan diperoleh bahwa *T* dengan $m = k + 1$ sisi juga akan memenuhi (a) dan (b). Dengan demikian, terbukti bahwa *lemma* tersebut benar[4].

Sebagai ilustrasi dari *lemma* tersebut perhatikan *tree* di bawah ini:



Gambar 2. Ilustrasi Lemma 1

Misalkan *loop* yang dimaksud dalam *Lemma* 1 dari *tree* di atas adalah $v_1v_2v_3v_4v_3v_5v_3v_2v_1$, maka semua sisi pada *tree* tersebut, yaitu $\{v_1,v_2\}$, $\{v_2,v_3\}$, $\{v_3,v_4\}$, dan $\{v_3,v_5\}$ muncul tepat dua kali dalam *loop*. Selain itu, semua *leaf* pada *tree*, yaitu v_1, v_4, v_5 muncul tepat sekali dalam *loop* (kemunculan v_1 dihitung sebagai satu kemunculan pada *loop*). Selain itu, v_1 dan v_4, v_4 dan v_5, serta v_5 dan v_1 merupakan dua buah *leaf* pada *loop* yang di antara keduanya tidak terdapat *leaf* lain dan dapat dilihat bahwa $v_1v_2v_3v_4$, $v_4v_3v_5$, dan $v_5v_3v_2v_1$ merupakan lintasan.

Selanjutnya, berdasarkan *Lemma* 1, batas atas rasio D_H/D_{MIN} sudah dapat ditentukan melalui teorema berikut ini:

Teorema 2:

Misalkan $G = (V, E)$ graf dengan fungsi bobot $f: E \rightarrow \mathbb{R}^+$ pada setiap sisinya dan $W \subseteq V(G)$ sebagai titik *customer*. Misalkan pula T_H adalah *Steiner tree* yang dihasilkan algoritma *heuristik* dengan bobot D_H, sedangkan T_{MIN} adalah *Steiner tree* yang dihasilkan oleh algoritma eksak dengan bobot D_{MIN} dan memiliki *l leaf* [4]. Maka,

$$\frac{D_H}{D_{MIN}} \leq 2\left(1 - \frac{1}{|W|}\right) \qquad (1)$$

3.3. Pengembangan Algoritma Steiner Tree

Algoritma *heuristic Steiner tree* yang telah dijelaskan sebelumnya merupakan garis besar dalam penentuan *Steiner tree*. Dalam bagian ini akan dijelaskan salah satu bentuk pengembangan dari algoritma *heuristic* tersebut. Perlu diperhatikan bahwa beberapa algoritma yang dijelaskan dalam bagian ini hanya merupakan suatu kemungkinan, jadi tidak menutup kemungkinan terdapat cara pengembangan lain dari algoritma *Steiner tree heuristic*.

3.3.1. Algoritma Penentuan Matriks Bobot Minimum

Langkah pertama dalam algoritma *heuristic* adalah menentukan matriks bobot minimum dan matriks lintasan dari graf *G*. Berikut ini merupakan algoritma yang dapat digunakan dalam menentukan matriks bobot minimum dan matriks lintasan terpendek dari suatu graf:

Algoritma Floyd – Warshall:
Masukan: Graf $G = (V,E)$, $V = \{1,2,...,n\}$
Keluaran: M (matriks bobot minimum) dan L (matriks lintasan terpendek).
Langkah-langkah:

1. Tentukan A = (a_{uv}) sebagai matriks bobot minimum berukuran *n* x *n* dan P = (p_{uv}) sebagai matriks lintasan terpendek berukuran *n* x *n* dengan $p_{uv} = v$.
2. Tentukan D = (d_{uv}) dengan d_{uv} adalah bobot sisi yang menghubungkan titik *u* dan *v*.
3. Tetapkan A(0) = D dan P(0) = P.
4. Definisikan $a_{uv}(j)$ menyatakan elemen (u,v) pada A(j) dan $p_{uv}(j)$ menyatakan elemen (u,v) pada P(j).
5. Untuk j:=1,...,n
   Untuk u:=1,...,n
   Untuk v:=1,...,n
   Jika j ganjil maka:
   Jika $a_{uv}(0) \leq a_{uj}(0) + a_{jv}(0)$ maka:
   $a_{uv}(1) = a_{uv}(0)$
   $p_{uv}(1) = p_{uv}(0)$
   Jika tidak:
   $a_{uv}(1) = a_{uj}(0) + a_{jv}(0)$.
   $p_{uv}(1) = p_{uj}(0)$.
   Jika tidak:

Jika $a_{uv}(1) \le a_{uj}(1) + a_{jv}(1)$ maka:
$a_{uv}(0) = a_{uv}(1)$
$p_{uv}(0) = p_{uv}(1)$
Jika tidak:
$a_{uv}(0) = a_{uj}(1) + a_{jv}(1)$.
$p_{uv}(0) = p_{uj}(1)$.

6. Jika n ganjil maka:
M:=A(1) dan L:=P(1)
Jika tidak:
M:=A(0) dan L:=P(0)

Dengan demikian, banyaknya matriks yang diperlukan dalam algoritma di atas hanyalah 4 buah untuk sebarang n dan total diperlukan $4n^2$ satuan tipe data. Hal ini akan sangat menghemat penggunaan memori.[5]

3.3.2. Algoritma Pengkonstruksian Graf Komplit

Langkah kedua dalam algoritma *heuristic Steiner tree* adalah pengkonstruksian graf komplit $G_1 = (W, E_1)$ berdasarkan matriks bobot minimum M bagi graf G. Berikut ini merupakan algoritma yang dapat digunakan dalam pengkonstruksian graf tersebut:

Algoritma Pengkonstruksian Graf Komplit:
Masukan: M, matriks bobot minimum dari graf $G = (V,E)$, $V = \{1,2,...,n\}$ dan $W \subseteq V(G)$.
Keluaran: $G1 = (W, E1)$, sebagai graf komplit dengan $f(\{i,j\}) = (m_{ij})$ untuk $\{i, j\} \in E_1$
Langkah – langkah:

1. Tetapkan 1 $E = \{\ \}$
2. Untuk i:=1,...,n
Untuk j:=1,...,n
Jika $i > j$ dan $i \in W$ dan $j \in W$ maka:
$E_1 := E_1 \cup \{i, j\}$
$f(\{i, j\}) := (m_{ij})$
3. $G_1 = (W, E_1)$ dan $f(i, j)$ untuk $\{i, j\} \in E1$ merupakan keluaran algoritma ini.[2]

3.3.3. Algoritma Pengkonstruksian Minimal Spanning Tree

Untuk menjalankan langkah ketiga dan kelima dalam algoritma *heuristic* diperlukan suatu metode pengkonstruksian minimal *spanning tree*. Beberapa metode yang dapat digunakan untuk menyelesaiakan hal ini diantaranya adalah algoritma *Kruskal* dan algoritma *Prim*. Dalam kasus ini yang akan digunakan adalah algoritma *Kruskal* [1].

Algoritma Kruskal:
Masukan: $G = (V, E)$ dengan fungsi bobot f: $E \rightarrow \mathbb{R}^+$ pada setiap sisinya.
Keluaran: $T = (V, F)$, sebagai minimal *spanning tree*.
Langkah – langkah:

1. Tetapkan $F = \{\ \}$
2. Urutkan $e_i = \{u, v\} \in E$, untuk i =1,...,$|\ E\ |$ secara menaik berdasarkan bobotnya:
Untuk i:=1,...,$|\ E\ |$
Untuk j:=i,...,$|\ E\ |$
Jika $f(e_i) < f(e_j)$ maka:
tmp := e_i
$e_i := e_j$
e_j := tmp
3. Definiskan $e_1, e_2, ..., e_{|H|}$ sebagai barisan sisi dari E yang sudah diurutkan melalui langkah 2.
4. Tetapkan i:=1
Selama $|\ F\ | < |V\ | - 1$
Jika $F \cup e_i$, $T = (V, F)$ tidak membentuk lingkaran maka:
$F: = F \cup e_i$
$i := i + 1$
5. $T = (V, F)$ merupakan *spanning tree* yang dimaksud.[2]

3.3.4. Algoritma Substitusi Lintasan

Algoritma substitusi lintasan dapat digunakan dalam langkah keempat algoritma *heuristic*. Tujuan dari algoritma ini adalah mengganti setiap sisi dalam graf T_1 pada algoritma *heuristic* dengan lintasan terpendek yang bersesuaian pada graf G. Berikut ini merupakan algoritma substitusi *lintasan*:

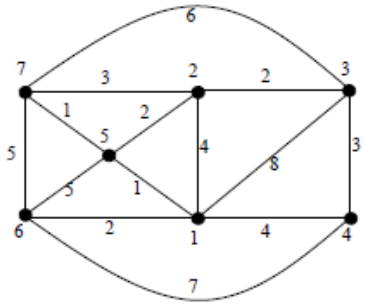
Algoritma Substitusi Lintasan:
Masukan: L, matriks lintasan terpendek dari graf $G = (V,E)$, graf $T_1 = (W, E_1)$ dengan $W \subseteq V(G)$.
Keluaran: $G_2 = (U, E2) \subseteq G$
Langkah – langkah:

1. Definiskan $e_1, e_2, ..., e_{|H|}$ sebagai barisan sisi dari E_1 dan misalkan $e_i(1)$ adalah titik pertama pada sisi e_i dan $e_i(2)$ adalah titik kedua pada sisi e_i, i = 1,...,$|\ E_1\ |$
2. Tetapkan $E_2 := E_1$ dan $U := \{\ \}$
Untuk i:=1,..., $|\ E_1\ |$
$e'(1) := e_i(1)$
Jika $e'_i(1) \notin U$ maka $U := U \cup e'_i(1)$
$e'_i(1) := L(e_i(1), e_i(2))$
Jika $e'_i(2) \notin U$ maka $U := U \cup e'_i(2)$
$E_2 := E_2 \cup e'$
Selama $e'_i(2) \neq e_i(2)$
$e'(1) := e'(2)$
Jika $e'_i(1) \notin U$ maka $U := U \cup e'_i(1)$
$e'(2) := L(e'(1), e'(2))$
Jika $e'_i(2) \notin U$ maka $U := U \cup e'_i(2)$
$E_2 := E_2 \cup e'$
Hapus e_i dari $E2$
3. Hapus sisi yang berulang pada E_2 sehingga setiap sisi muncul tepat sekali.
4. $G_2 = (U, E_2)$ merupakan graf yang dimaksud.

Langkah keenam dalam algoritma *heuristic* sudah cukup jelas sehingga tidak perlu dikembangkan lagi.[2]

3.4. Proses Pembentukan Steiner Tree dengan Algoritma Heuristik

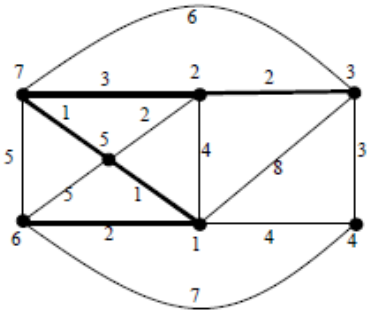
Bagian ini akan memberikan suatu ilustrasi tentang pembentukan *Steiner tree* dengan algoritma *heuristic*. Perhatikan graf $G = (V, E)$ berikut:



Gambar 3. Representasi Graf G

Misalkan himpunan titik *customer* $W = \{3, 6, 7\}$ maka dapat ditentukan *Steiner tree* di G atas W.

Hal pertama yang harus dilakukan pada algoritma *heuristic* adalah menentukan matriks bobot minimum dan matriks lintasan terpendek dengan algoritma 3.3.1, hasil yang didapat serupa dengan hasil pada proses eksak. Selanjutnya, konstruksi graf komplit $G_1=(W, E_1)$ dengan $f(\{i, j\}) = (m_{ij})$ untuk $\{i, j\} \in E_1$, hal ini dapat dilakukan dengan algoritma 3.3.2. Setelah itu, konstruksi minimal *spanning tree* T_1 pada G_1 dengan algoritma 3.3.3 sehingga didapatkan sisi-sisi pada T_1 adalah {1, 2}, {2, 3} dan {3,4}. Kemudian lakukan substitusi untuk setiap sisi pada T_1 dengan lintasan terpendek yang bersesuaian pada G, yakni {1, 2} dengan 1 – 2 – 5, {2, 3} dengan 2 – 3, dan {3,4} dengan 3 – 4 sehingga diperoleh graf G_2, hal ini dapat dilakukan dengan algoritma 3.3.4. Selanjutnya, konstruksi minimal *spanning tree* T_2 pada G_2 dengan algoritma 3.3.3 sehingga didapatkan sisi-sisi pada T_2 adalah {1, 5}, {5, 2}, {3, 4} dan {2, 3}. Terakhir, hapus v dari T_2 sehingga apabila $v \notin W$, $v \in V(T_2)$, dan $d(v) = 1$ sehingga dihasilkan *Steiner tree* hasil aproksimasi berikut:



Gambar 4. Steiner tree hasil algoritma heuristik

Steiner hasil aproksimasi di atas memiliki bobot 9, hal ini sesuai dengan *teorema 2*, yaitu:

$$\frac{D_H}{D_{MIN}} = \frac{9}{8} \leq \frac{4}{3} = 2\left(1 - \frac{1}{|W|}\right)$$

Titik 1, 2, dan 5 juga berperan sebagai titik *Steiner* dalam *Steiner tree* hasil aproksimasi ini.

4. PENERAPAN STEINER TREE DALAM KONSTRUKSI PIPA GAS

Masalah pengkonstruksian jaringan pipa gas dalam satu pulau yang menghubungkan semua kota dalam satu pulau tersebut yang memiliki depo gas/BBM dengan panjang pipa yang minimum dapat dimodelkan sebagai pengkonstruksian *Steiner tree* pada suatu graf. Suatu pulau dapat dipandang sebagai graf dengan setiap kota pada pulau tersebut dimodelkan sebagai titik dan setiap dua kota yang berbatasan langsung diasumsikan memiliki sebuah jalur yang dimodelkan sebagai sisi pada graf, sedangkan bobot dari setiap sisi tersebut adalah panjang jalur yang diwakilinya. Lebih lanjut, setiap kota di pulau tersebut yang memiliki depo gas/bbm dimodelkan sebagai titik customer.

Tujuan dalam pemodelan di atas adalah mencari jaringan berbentuk *tree* yang menghubungkan setiap kota yang termasuk dalam titik *customer* dengan bobot minimum, jaringan inilah yang disebut sebagai *Steiner tree*. Jika dalam *Steiner tree* yang diperoleh terdapat kota - kota lain selain dari kota yang termasuk dalam titik customer maka kota lain tersebut merupakan titik *Steiner*. Secara ringkas, pemodelan matematika dalam masalah ini dapat dijelaskan melalui penjelasan berikut.

Pulau direpresentasikan dalam bentuk graf $G = (V, E)$ yang memiliki fungsi bobot $f : E \rightarrow \mathbb{R}^+$ pada setiap sisinya, dengan V menyatakan himpunan kota-kota di pulau tersebut, E menyatakan himpunan jalur penghubung antar kota yang berbatasan langsung, $f(\{i, j\})$ untuk $\{i,j\} \in E$ menyatakan panjang jalur yang dihitung dengan perhitungan topografi, dan $W \subseteq V$ menyatakan himpunan kota – kota di suatu pulau yang memiliki depo gas/bbm [2].

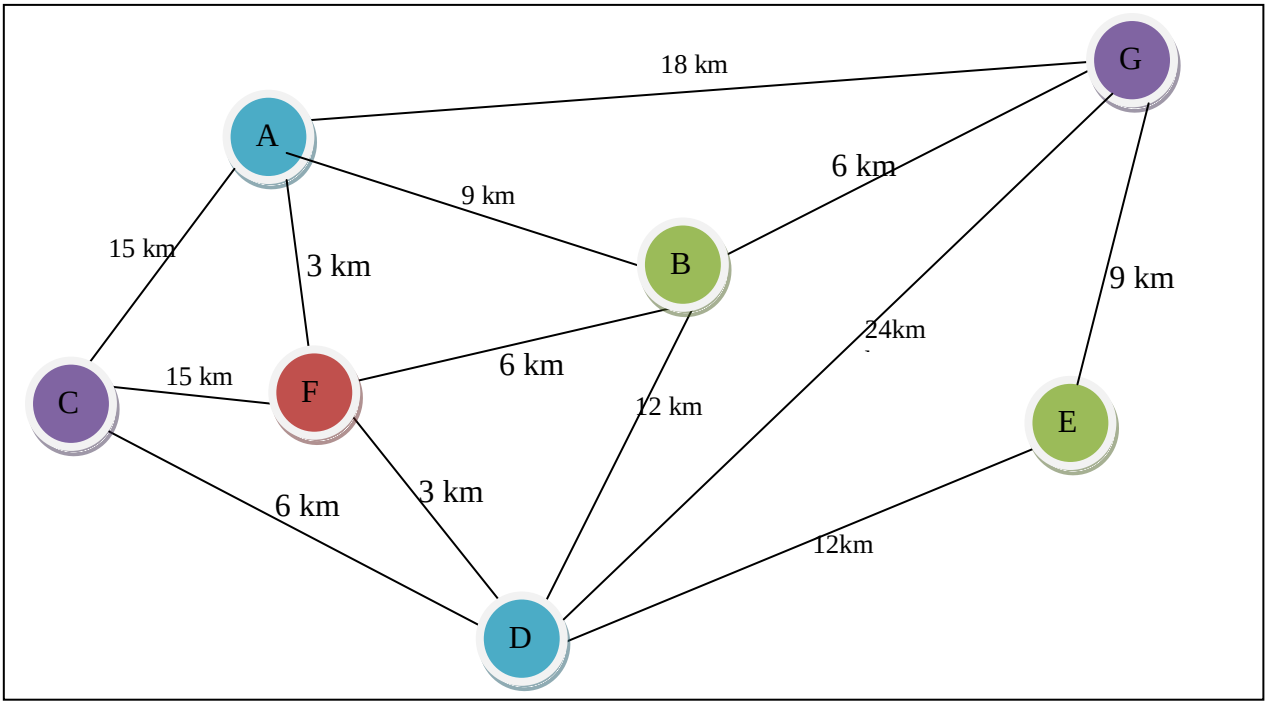
Tujuan pemodelan ini adalah mencari *Steiner tree* $T = (W^*,F)$ di G dengan bobot T minimum, $W \subseteq W^*$, dan kota-kota dalam W^*/W merupakan titik *Steiner*.

4.1. Penentuan Titik Customer

Kriteria penentuan kota yang akan dijadikan sebagai titik *customer* didasarkan atas: kota tersebut merupakan kotamadya atau kabupaten dan kota tersebut memiliki depo gas/bbm darat, laut, atau udara [2].

4.2. Penentuan Sisi

Kriteria yang digunakan untuk menentukan sisi adalah:



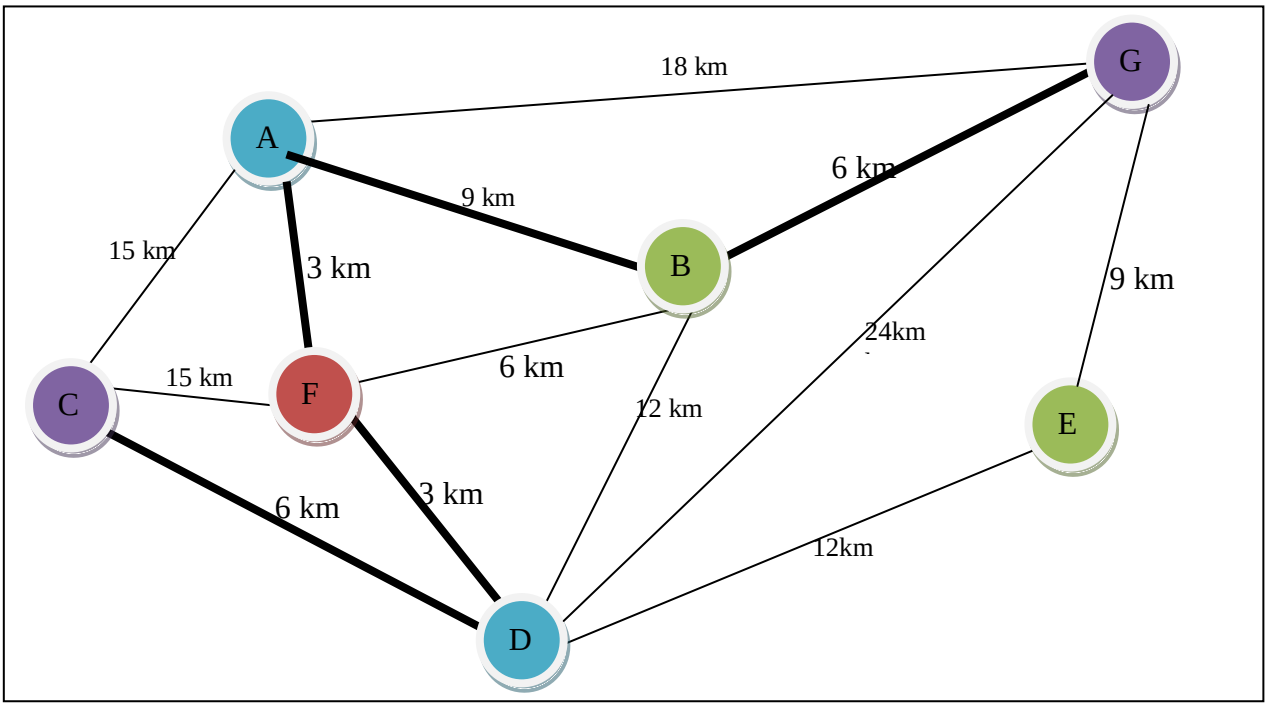
Gambar 5. Contoh peta jaringan pipa gas dalam suatu pulau

1. Sisi hanya mewakili jalur yang menghubungkan dua buah kota yang berbatasan secara langsung [2].
2. Jika terdapat bentang alam seperti waduk, danau, atau gunung (yang memiliki ketinggian > 1500 meter) yang membatasi dua buah kota maka diasumsikan tidak ada sisi yang menghubungkan kedua kota tersebut [2].

4.3. Pengkonstruksian Steiner Tree pada Jaringan Pipa Gas dalam Suatu Pulau

Pada bagian ini akan ditampilkan proses pengkonstruksian *Steiner tree* yang merepresentasikan jaringan pipa gas dalam suatu pulau dengan panjang yang minimum.

Jaringan pipa gas pada suatu pulau misalkan digambarkan seperti gambar 5.



Gambar 6. Hasil konstruksi steiner tree pada jaringan pipa gas

Apabila dipilih kota yang menjadi *customer* yaitu kota A, C, G. Maka hasil konstruksi *steiner tree* dengan algoritma *heuristik*nya seperti gambar 6.

Steiner Tree pada gambar 6 memiliki bobot total 27 km dan menghubungkan 6 kota dalam satu pulau, dengan 3 kota sebagai titik *steiner tree*.

Berikut ini perinciannya:

Tabel 1 Hasil Perincian Kontruksi Steiner Tree pada Jaringan Pipa Gas

Kota 1	Kota 2	Bobot
C	D	6 km
A	F	3 km
F	D	3 km
A	B	9 km
B	G	6 km

Kota yang berperan sebagai titik *Steiner* pada *Steiner tree* tersebut adalah:

D, F, dan B.

5. KESIMPULAN

Kesimpulan yang dapat di dapat dalam makalah ini diantaranya:

1. *Steiner Tree* merupakan salah satu aplikasi teori graf yang dapat diaplikasikan untuk membangun konstruksi jalur pipa gas pada jaringan pipa gas dalam suatu pulau.
2. Untuk membuat *Steiner Tree* dari suatu graf dapat digunakan algoritma *heuristik* dengan perincian algoritma *heuristik* menggunakan algoritma-algoritma lain yaitu algoritma *Floyd-Warshall*, algoritma *Kruskal*, algoritma pengkonstruksian graf komplit, dan algoritma substitusi lintasan.
3. Algoritma *heuristik* untuk membentuk *steiner tree* memiliki waktu yang paling optimal dibanding algoritma lain, meskipun begitu algoritma *heuristik* ini tidak menghasilkan *steiner tree* yang paling optimal, akan tetapi bobot *steiner tree* yang dihasilkan tidak melebihi batas atas.
4. Di Indonesia khususnya Pulau Jawa dapat diterapkan pendistribusian gas dengan menggunakan pipa gas, dengan mengacu pada *steiner tree*, sehingga pendistribusian gas di Pulau Jawa akan lebih optimal dalam hal waktu dan biaya.

REFERENSI

[1] Rinaldi Munir, "Matematika Diskrit", Informatika Bandung, 2003

[2] Adie Pradipto, "Desain Jaringan Pipa Gas Di Pulau Jawa dengan Steiner Tree", FMIPA ITB, 2008

[3] Steiner Tree Problem, http://en.wikipedia.org/wiki/Steiner_tree_problem, 19 Desember 2009, 20:00

[4] A Tabu Search Heuristic for the Steiner Tree Problem, http://www.gerad.ca/fichiers/cahiers/G9801.ps, 19 Desember 2009, 20:00

[5] Algoritma Floyd-Warshall, http://id.wikipedia.org/wiki/Algoritma_Floyd-Warshall, 19 Desember 2009, 20:00

[6] Permen ESDM Tentang Kegiatan Usaha Gas Bumi Melalui Pipa , http://www.tambangnews.com/regulasi/peraturan-menteri/338-permen-esdm-tentang-kegiatan-usaha-gas-bumi-melalui-pipa.html , 20 Desember 2009, 14:37