Optimizing Power Distribution Network Using Dijkstra's Algorithm And Graph Clustering

Author Dery Rahman Ahaddienata, 13515097[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13515097@std.stei.itb.ac.id

***Abstract*— Optimisasi merupakan suatu proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dapat dicapai). Dalam ilmu matematika, optimisasi digunakan untuk mengatasi permasalahan dengan cara mencoba mencari nilai minimal atau maksimal dari suatu fungsi riil. Ada banyak metode yang dapat digunakan untuk mencari nilai ekstreem tersebut, salah satunya adalah dengan metode *graph*. *Graph* dapat digunakan untuk mencari *shortest path* maupun *longest path* antara satu *node* ke *node* lainnya. Saat kita membahas tentang optimisasi *cost-reduction*, tentunya kita berharap untuk mendapatnya *path* terpendek dari metode *graph* ini.**

Makalah ini akan berfokus tentang penggunaan teori *graph* untuk mengoptimalkan distribusi listrik agar merata. Fokus permasalahan yang akan dibahas adalah *cost-reduction* pada distribusi listrik dengan cara mencari *shortest path* yang dapat dicapai dari sebuah generator ke beberapa daerah dengan menggunakan *Dijkstra's Algorithm*. Kemudian jika ada beberapa daerah terlalu jauh dari generator maka akan dilakukan *clustering* yang akan memisahkan beberapa daerah tersebut dari generator dan kemudian akan di bangun generator baru yang akan memasok beberapa daerah tersebut.

***Keywords*—clustering, dijkstra algorithm, optimization, power distribution.**

I. INTRODUCTION

Listrik merupakan kebutuhan paling penting dalam kehidupan modern saat ini. Semakin tinggi penduduk dunia, kebutuhan akan listrik semakin tinggi pula. Ada banyak sumber yang dapat digunakan untuk menghasilkan listrik, mulai dari batu bara, turbin, tenaga matahari, hingga tenaga nuklir. Saat ini sudah banyak *power plan* yang dibangun, namun masih banyak menemui kendala dalam aplikasinya menyebabkan kerugian bagi perusahaan listrik tersebut. Salah satu contohnya adalah *distribution network* yang tidak optimal. Beberapa dari jaringan distribusi *power plan* tersebut mempunyai jarak yang cukup jauh dari sumber, sehingga listrik yang disalurkan tidak optimal dikarenakan ada sebagian besar energi listrik yang terbuang percuma. Hal ini disebut susut daya listik. Semakin jauh lintasannya, semakin besar hambatan pada jaringan distribusi, sehingga menyebabkan energi yang terbuang semakin besar juga.

Electricity is often generated a long way from where it is used, and is transmitted long distances through power lines. Although the resistance of a short length of power line is relatively low, over a long distance the resistance can become substantial. A power line of resistance R causes a power loss of I^2R ; this is wasted as heat. By reducing the current, therefore, the I^2R losses can be minimized.[4]

Susut daya listrik merupakan persoalan krusial yang menyebabkan kerugian besar bagi perusahaan pemasok energi. Contohnya seperti yang dihadapi oleh PLN (Perusahaan Listrik Negara) di Indonesia dan hingga saat ini permasalahan susut daya tersebut belum dapat sepenuhnya terpecahkan. Pemadaman bergilir terpaksa dilakukan untuk menanggulangi permasalahan ini. [5]

Oleh karena itu, makalah ini ditulis untuk menjawab permasalahan tersebut dengan menggunakan model *graph*. Untuk jaringan distribusi, penulis merekomendasikan untuk mengaplikasikan metode pencarian *shortest path* ke semua daerah dari suatu generator menggunakan *Dijkstra's Algorithm*.

Sebelumnya, pemerintah Indonesia telah menargetkan pasokan listrik sebesar 35000MW[6]. Oleh karena itu penempatan pembangunan generator listrik baru juga perlu diperhatikan secara cermat sehingga tidak akan ada lagi kerugian yang besar akibat pengaruh dari susut daya. Dengan menggunakan pendekatan teori *graph*, penulis akan menggunakan *clustering*. Clustering dilakukan dengan memotong sebuah sambungan listrik jika jarak sambungan dari suatu daerah ke generator terlalu jauh dan melewati *threshold* yang telah ditentukan.

Dalam makalah ini, penulis menggunakan data-data yang bersifat asumsi. Segala data seperti besaran jarak, biaya, dan energi dalam makalah ini bukan merupakan data yang valid. Penulis hanya menggunakan data asumsi tersebut sebagai perhitungan agar diketahui seberapa signifikannya penggunaan *graph* ini untuk mengurangi ongkos dari permasalahan *power distribution network*.

II. BASIC THEORY

2.1 Graph Definition

A graph G = (V, E) consists of V , a nonempty set of vertices (or nodes) and E, a set of edges. Each edge has either one or two vertices associated with it, called its endpoints. An edge is said to connect its endpoints.[1]

Graph terdiri dari V, yang merupakan simpul dan E, yang merupakan sisi. Dari definisi, V tidak boleh kosong, sedangkan E boleh kosong.

Graph dibagi menjadi beberapa kategori. Berdasarkan ada tidaknya sisi ganda atau gelang pada *graph*, graph digolongkan menjadi 2 jenis, *simple graph* dan *unsimple-graph*. Berdasarkan jumlah simpulnya, *graph* dibedakan menjadi *limited graph* dan *unlimited graph*. Berdasarkan dari orientasi arah, *graph* dibedakan menjadi *undirected graph* dan *directed graph* [2]. Jenis *graph* yang digunakan pada *power distribution network* adalah *simple undirected graph*.

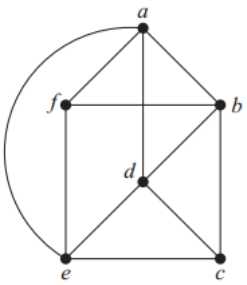

Figure 2.1, simple undirected graph

Pada *power distribution network*, simpul digunakan untuk merepresentasikan daerah-daerah yang akan disuplai sekaligus digunakan untuk merepresentasikan letak generator. Sedangkan sisi digunakan sebagai jarak antara 1 daerah ke daerah lainnya yang menjadi sambungan listrik.

2.2 Weighted Graph

Weighted graph is a graph with numbers assigned to its edges[1].

Graph pada *power distribution network* menggunakan *weighted graph* dengan bobot yang merepresentasikan jarak antara satu tempat ke tempat lainnya.

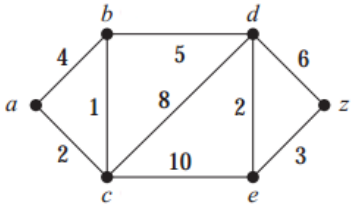

Figure 2.2, weighted graph

2.3 Shortest Path

Persoalan mencari lintasan terpendek di dalam *graph* merupakan salah satu persoalan optimasi. Jenis *graph* yang digunakan dalam untuk mencari lintasan terpendek adalah *weighted graph*[2].

Ada banyak metode yang dapat digunakan untuk mencari *shortest path*, salah satunya adalah dengan menggunakan *Dijkstra's Algorithm*. Dalam kasus *power distribution network*, *shortest path* digunakan untuk menentukan *path* dari sebuah generator ke beberapa daerah yang direpresentasikan sebagai *graph*.

2.4 Dijkstra's Algorithm

Dijkstra's algorithm finds the length of a shortest path between two vertices in a connected simple undirected weighted graph [1].

Dijkstra's Algorithm menggunakan prinsip *greedy*. Pada setiap langkah, kita memilih sisi yang berbobot minimum kemudian memasukannya ke dalam himpunan solusi.

Pseudo-code Dijkstra's Algorithm

```
Procedure Dijkstra (input m: matriks, integer a: simpul awal)

d0,s1,…,sn-1 : integer
d0,d1,…,dn-1 : integer

{langkah 0, inisiasi}
for i ← 0 to n - 1 do
   si ← 0
   di ← mai
endfor

{langkah 1}
sa ← 1 {karena simpul a adalah simpul asal lintasan terpendek, jadi simpul a sudah pasti terpilih dalam lintasan terpendek}
da ← ∞ {tidak ada lintasan terpendek dari simpul a ke a}

{langkah 2, 3, … n - 1}
for i ← 2 to n - 1 do
   {cari j sedemikian sehingga sj = 0 dan dj = min(d0,d1,…,dn-1)}
   sj ← 1 {simpul j sudah terpilih ke dalam lintasan terpendek, perbarui di yang belum dikunjungi dengan di(baru) = min(di(lama),dj + mji}
endfor
```

Pseudo-code Dijkstra's Algorithm[2].

Diberikan sebuah *graph* seperti gambar 2.2. Kemudian, cari *shortest path* minimum dari simpul a ke semua simpul. Matriks yang merepresentasikan *graph* pada gambar 2.2 :

	j	0	1	2	3	4	5
i		a	b	c	d	e	z
0	a	0	4	2	∞	∞	∞
1	b	4	0	1	5	∞	∞
2	c	2	1	0	8	10	∞
3	d	∞	5	8	0	2	6
4	e	∞	∞	10	2	0	3
5	z	∞	∞	∞	6	3	0

Tabel 2.1, representasi graph gambar 2.2

Langkah-langkah menentukan *shortest path*, menggunakan *Dijkstra's Algorithm*.

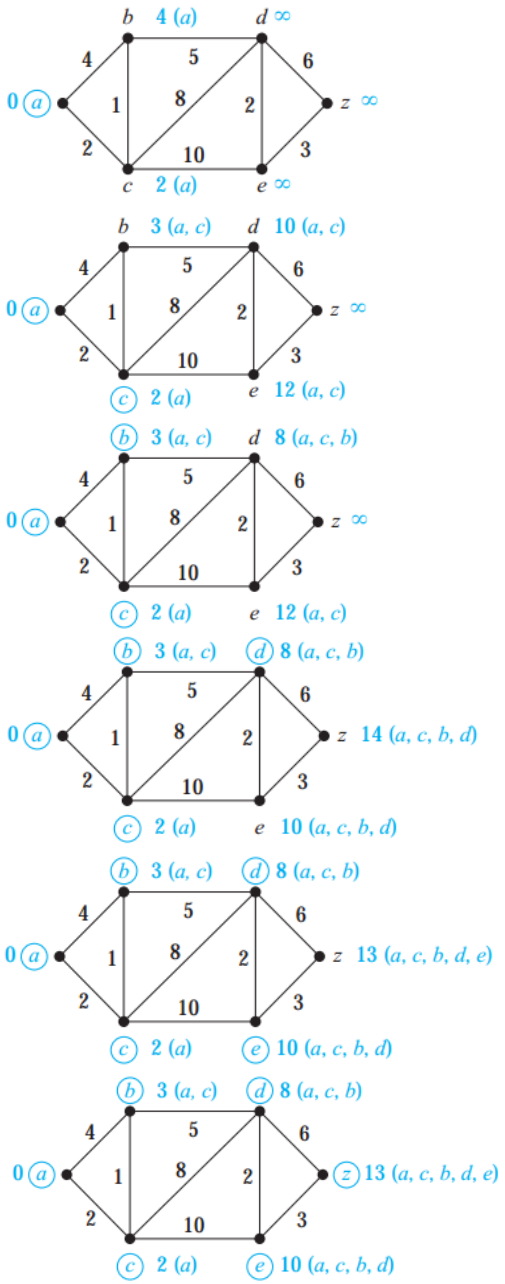

Figure 2.3, langkah pencarian *shortest path*

Gambar 2.3 menunjukan bagaimana *Dijkstra's Algorithm* bekerja pada *graph* gambar 2.2. Kita lihat *pseudo-code Dijkstra's Algorithm*. Diberikan m sebagai matriks, a sebagai indeks simpul. Kita menginginkan *node* a sebagai *starting point*. *Step* inisiasi, definisikan s_0, s_1, s_2, s_3, s_4, dan s_5 sebagai simpul. Jika simpul bernilai 1, berarti sudah dikunjungi. Sebagai inisiai, s_0 hingga s_5 kita beri nilai 0 karena belum dikunjungi dan d_0, d_1, d_2, d_3, d_4, dan d_5 diisi dengan jarak dari simpul a, menuju simpul s_0, s_1, .. s_5.

Step pertama, s_0 di*set* 1 karena telah dikunjungi dan d_0 diisi bilangan yang besar ∞ karena simpul a tidak akan pernah sampai ke simpul a lagi.

Step kedua, cari s_j yang bernilai 0 dan cari d_j yang bernilai paling minimum, kemudian isi s_j dengan 1, karena telah dikunjungi dan update semua d yang belum dikunjungi dengan d_i(baru) = min(d_i(lama), d_j + m_{ij}). Lakukan iterasi seperti ini sebagai langkah 2 hingga langkah ke n-1.

Selanjutnya kita dapatkan d_i dimana i=0,1,2,3,4,5 yang merupakan jarak terpendek dari simpul 0 ke simpul 1,2,3,4,5. Sehingga kita dapatkan :

a→c = 2
a→c→b = 3
a→c→b→d = 8
a→c→b→d→e = 10
a→c→b→d→e→z = 13

2.5 Clustering

Clustering yang dilakukan dengan mencari jarak antara generator dan sebuah *node* yang melebihi *threshold* kemudian memotong sambungan tersebut, sehingga menjadi *graph* yang baru. Dalam *power distribution network*, hal ini dilakukan jika jarak dari suatu daerah ke generator sudah terlalu jauh.

Kita ambil contoh dari hasil Dijkstra's Algorithm sebelumnya. Seumpama, *threshold* yang kita kehendaki adalah 9. Maka jarak node yang pertama kali terdeteksi dari a jika nilainya ≥ 9 sisi tersebut akan dihapus. Sehingga menjadi :

a→c = 2
a→c→b = 3
a→c→b→d = 8
e→z = 3

III. Power Distribution Problem Assumption (Study Case : Paiton Power Station In East Java, Indonesia)

3.1. Asumsi *Graph* Persebaran Kota di Jawa Timur

Berikut disajikan tabel yang berisi asumsi jarak dari satu daerah ke daerah lainnya yang ada di Jawa Timur. Jarak tersebut merupakan jarak dari jalur yang *possible* untuk didirikan sambungan listrik.

Daerah / Kota	Indeks Simpul	Daerah / Kota	Indeks Simpul
Paiton Power Station	0	Lumajang	7
Bondowoso	1	Kediri	8
Besuki	2	Pare	9
Jember	3	Ponorogo	10
Malang	4	Surabaya	11
Pasuruan	5	Sragen	12
Blitar	6		

Tabel 3.1, representasi simpul sebagai daerah/kota

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	∞	27	∞	∞	74	∞	77	∞	∞	∞	∞	∞
1	∞	0	50	40	∞	∞	∞	113	∞	∞	∞	∞	∞
2	27	50	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	∞	40	∞	0	∞	∞	∞	74	∞	∞	∞	∞	∞
4	∞	∞	∞	∞	0	51	78	121	∞	76	∞	∞	∞
5	74	∞	∞	∞	51	0	∞	∞	136	∞	∞	66	∞
6	∞	∞	∞	∞	78	∞	0	167	41	∞	119	∞	∞
7	77	113	∞	74	121	∞	167	0	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	136	41	∞	0	23	∞	133	151
9	∞	∞	∞	∞	76	∞	∞	∞	23	0	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	119	∞	∞	∞	0	∞	∞
11	∞	∞	∞	∞	∞	66	∞	∞	133	∞	∞	0	234
12	∞	∞	∞	∞	∞	∞	∞	∞	151	∞	∞	234	0

Tabel 3.2, matriks asumsi jarak antar daerah

Dari tabel tersebut dapat dibuat representasi *graph*nya

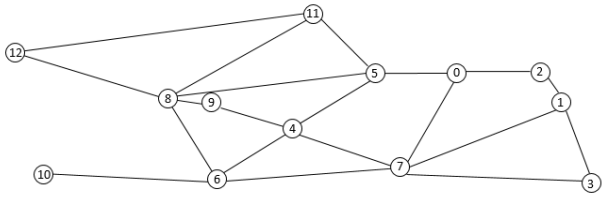

Figure 3.1, *graph* jalur distribusi listrik yang *possible* di Jawa Timur

3.2. Asumsi Perhitungan

Paiton mempunyai kapasitas total 4,040 MW[7]. Seandainya asumsi untuk masing-masing daerah membutuhkan pasokan P_f listrik sebanyak 100-300MW, dan asumsi *heat-losses* adalah sekitar 0.1% setiap kilometer jarak d, maka setiap distribusi ke daerah yang membutuhkan pasokan P_f listrik, *Paiton Power Station* perlu membangkitkan P_i listrik dari generator yang dirumuskan :

$$\mathbf{P_i=P_f*(1+(0.1\%*d))} \ldots \text{(i)}$$

Berikut merupakan asumsi kebutuhan listrik setiap daerah :

Indeks Simpul	Daerah / Kota	Kebutuhan
0	*Paiton Power Station*	N/A
1	Bondowoso	100
2	Besuki	100
3	Jember	120
4	Malang	200
5	Pasuruan	120
6	Blitar	120
7	Lumajang	100
8	Kediri	150
9	Pare	100
10	Ponorogo	200
11	Surabaya	300
12	Sragen	200

Tabel 3.3, asumsi kebutuhan listrik setiap daerah

Dari tabel 3.3, kita dapat mengetahui asumsi total kebutuhan listrik di Jawa Timur adalah sebesar 1810MW. Berikut adalah graph yang merepresentasikan jaringan distribusi listrik yang digunakan (sebelum metode *Dijkstra's Algorithm* diaplikasikan).

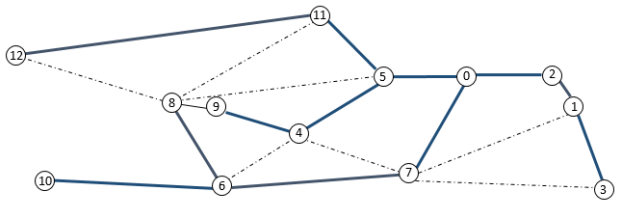

Figure 3.2, representasi *Graph* sebelum perhitungan *Dijkstra's Algorithm*

Dengan menggunakan persamaan (i) dapat diperoleh besar daya yang harus dibangkitkan dari sumber untuk dapat sampai ke suatu daerah tertentu. Berikut adalah tabel perhitungan *power supply* yang diperlukan *Paiton Power Station* untuk setiap daerah (sebelum metode *Dijkstra's Algorithm* diaplikasikan)

Simpul	Daerah / Kota	Jarak (km)	Supply
0	*Paiton Power Station*	0	N/A
1	Bondowoso	77	107.7
2	Besuki	27	102.7
3	Jember	117	134.04
4	Malang	125	225
5	Pasuruan	74	128.88
6	Blitar	244	149.28
7	Lumajang	77	107.7
8	Kediri	285	192.75
9	Pare	201	120.1
10	Ponorogo	363	272.6
11	Surabaya	140	342
12	Sragen	374	274.8

Tabel 3.4, daya yang dibangkitkan dari sumber

Untuk mens*upply* daya sebesar 1810MW, Paiton perlu menyediakan daya sebesar 2157.55MW pada generatornya. Daya susut atau energi yang terbuang selama perjalanan adalah sebesar 347.55MW, atau 16.11%. Nilai tarif listrik dalam rupiah adalah Rp 1.350,00/ kWh[9], sehingga asumsi kerugian pertahun mencapai Rp 4.110.126.300.000,00.

Penulis menggunakan 300km sebagai asumsi *threshold* yang akan digunakan untuk *clustering*.

IV. Implementation Dijkstra's Algorithm on Power Distribution Network

4.1. Implementasi Dengan Bahasa C

Penulis menggunakan bahasa C untuk implementasi *Dijkstra's Algorithm* pada kasus Paiton *Power Distribution* di Jawa Timur.

dijkstra.c

```c
/* A C / C++ program for Dijkstra's single source shortest path algorithm. The program is for adjacency matrix representation of the graph
*/

#include <stdio.h>
#include <limits.h>

// Number of vertices in the graph
#define V 13

/* A utility function to find the vertex with minimum distance value, from the set of vertices not yet included in shortest path tree */
int minDistance(int dist[], bool sptSet[]){
   // Initialize min value
   int min = INT_MAX, min_index;

   for (int v = 0; v < V; v++)
     if (sptSet[v] == false && dist[v] <= min)
         min = dist[v], min_index = v;

   return min_index;
}

// A utility function to print the constructed distance array
int printSolution(int dist[], int n){
   printf("Vertex   Distance from Source\n");
   for (int i = 0; i < V; i++)
      printf("%d \t\t %d\n", i, dist[i]);
}

/* Funtion that implements Dijkstra's single source shortest path algorithm for a graph represented using adjacency matrix representation */
void dijkstra(int graph[V][V], int src){
     int dist[V];
     /* The output array.  dist[i] will hold the shortest distance from src to i */

     bool sptSet[V];
```

```
   /* sptSet[i] will true if vertex i is included in shortest path
tree or shortest distance from src to i is finalized */

     // Initialize all distances as INFINITE and stpSet[] as false
     for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, sptSet[i] = false;

     // Distance of source vertex from itself is always 0
     dist[src] = 0;

     // Find shortest path for all vertices
     for (int count = 0; count < V-1; count++){
       /* Pick the minimum distance vertex from the set of vertices
not yet processed. u is always equal to src in first iteration.*/
         int u = minDistance(dist, sptSet);

         // Mark the picked vertex as processed
         sptSet[u] = true;

         // Update dist value of the adjacent vertices of the picked
vertex.
         for (int v = 0; v < V; v++)
   /* Update dist[v] only if is not in sptSet, there is an edge from
to v, and total weight of path from src to  v through u is smaller
than current value of dist[v] */
         if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX &&
dist[u]+graph[u][v] < dist[v])
            dist[v] = dist[u] + graph[u][v];
     }

     // print the constructed distance array
     printSolution(dist, V);
}

int main(){
   int graph[V][V] = {
   { 0,   0,  27,   0,   0,  74,   0,  77,   0,   0,   0,   0,   0},
   { 0,   0,  50,  40,   0,   0,   0, 113,   0,   0,   0,   0,   0},
   {27,  50,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0},
   { 0,  40,   0,   0,   0,   0,   0,  74,   0,   0,   0,   0,   0},
   { 0,   0,   0,   0,   0,  51,  78, 121,   0,  76,   0,   0,   0},
   {74,   0,   0,   0,  51,   0,   0,   0, 136,   0,   0,  66,   0},
   { 0,   0,   0,   0,  78,   0,   0, 167,  41,   0, 119,   0,   0},
   {77, 113,   0,  74, 121,   0, 167,   0,   0,   0,   0,   0,   0},
   { 0,   0,   0,   0,   0, 136,  41,   0,   0,  23,   0, 133, 151},
   { 0,   0,   0,   0,  76,   0,   0,   0,  23,   0,   0,   0,   0},
   { 0,   0,   0,   0,   0,   0,   0, 119,   0,   0,   0,   0,   0},
   { 0,   0,   0,   0,   0,  66,   0,   0, 133,   0,   0,   0, 234},
   { 0,   0,   0,   0,   0,   0,   0,   0, 151,   0,   0, 234,   0}
   };

   dijkstra(graph, 0);

 return 0;
}
```

Implementasi *Dijsktra's Algorithm*[8].

4.2. Hasil dan Perhitungan

Berikut adalah hasil yang didapat setelah program tersebut dieksekusi

```
Hasil
   Vertex   Distance from Source
   0                0
   1                77
   2                27
   3                117
   4                125
   5                74
   6                203
   7                77
   8                210
   9                201
   10               322
   11               140
   12               361
```

Setelah menjalankan *Dijkstra's Algorithm*, hita dapat mengetahui jarak minimum yang diperlukan dari sumber menuju suatu daerah. Berikut adalah *graph* yang merepresentasikan jaringan distribusi listrik setelah perhitungan *Dijkstra's Algorithm*.

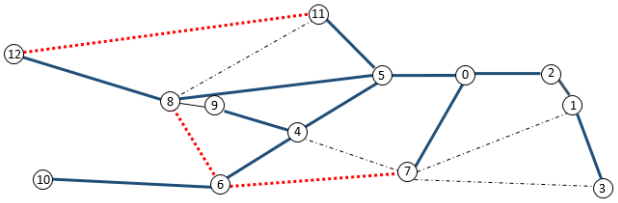

Figure 4.1, representasi *graph* setelah perhitungan *Dijkstra's Algorithm*

Pada gambar 4.1, warna merah merupakan sambungan jaringan distribusi sebelum perhitungan menggunakan *Dijkstra* yang diyakini jika melalui jalur tersebut maka akan memakan jarak yang terlalu jauh dari sumber. Jaringan distribusi yang digunakan setelah perhitungan adalah jalur berwarna biru.

Sambungan distribusi listrik yang diputus antara lain dari Lumajang ke Blitar, Blitar ke Kediri, dan Surabaya ke Sragen. Kemudian dibangun sambungan distribusi listrik yang baru antara lain dari Malang ke Blitar, Pasuruan ke Kediri, dan Kediri ke Sragen.

Berikut adalah table perhitungan *power supply* yang diperlukan *Paiton Power Station* untuk setiap daerah setelah perhitungan *Dijkstra's Algorithm*.

Simpul	Daerah / Kota	Jarak (km)	Supply
0	*Paiton Power Station*	0	N/A
1	Bondowoso	77	107.7
2	Besuki	27	102.7
3	Jember	117	134.04
4	Malang	125	225
5	Pasuruan	74	128.88
6	Blitar	203	144.36
7	Lumajang	77	107.7
8	Kediri	210	181.5
9	Pare	201	120.1
10	Ponorogo	322	264.4
11	Surabaya	140	342
12	Sragen	361	272.2

Tabel 4.1, daya yang dibangkitkan dari sumber setelah optimasi

Total untuk mensupply 1810MW adalah 2130.58MW. Daya susut atau energi yang terbuang selama perjalanan adalah sebesar 320.58MW, atau 15.05%. Sebelum dioptimasi menggunakan *Dijkstra*, daya susut yang terbuang adalah sebesar 347.55MW. Dalam kasus Paiton *Power Distribution Network*, penggunaan *Dijkstra's Algorithm* memberikan efek pengurangan daya susut sebesar 7.76%. Perusahaan listrik dapat mengurangi kerugian sebesar Rp 318.947.220.000,00 per tahun.

V. CLUSTERING POWER DISTRIBUTION NETWORK

Asumsi *Threshold* yang digunakan dalam kasus ini adalah 300km. Oleh karena itu, sambungan distribusi listrik ke Sragen dan Ponorogo akan di*cut*, dan selanjutnya, Sragen dan Ponorogo akan mendapatkan *supply* daya bukan dari Paiton lagi, melainkan dari pembangkit listrik yang lain.

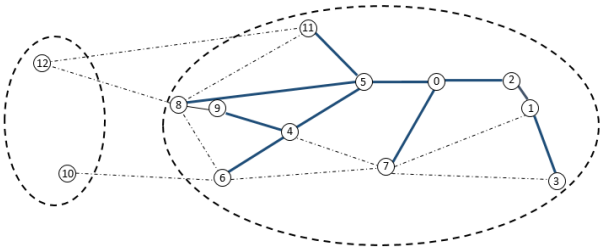

Figure 5.1, hasil *clustering power distribution network*

VI. CONCLUSION

Ada banyak cara untuk mengoptimalkan jaringan distribusi listrik. Salah satunya adalah dengan meminimalisirkan jarak dari sumber listrik ke suatu daerah tertentu. Hal tersebut dapat dilakukan dengan menggunakan pendekatan teori *graph*.

Model *graph* yang digunakan adalah *weighted graph*, dimana simpul merepresentasikan daerah yang membutuhkan *supply* listrik sedangkan sisi merepresentasikan jalur distribusi yang *possible*. Bobot dalam *graph* merepresentasikan jarak antara 1 simpul ke simpul lainnya dalam satuan km.

Dengan menggunakan metode pencarian *shortest path Dijkstra's Algorithm*, dapat diketahui jalur distribusi yang paling optimal, sehingga kerugian akibat daya susut yang terbuang dapat diminimalisirkan.

Kemudian dengan menggunakan *clustering*, diharapkan daerah yang terlalu jauh dapat mendapatkan *supply* listrik dari generator yang baru sehingga pemerataan pembangunan generator dapat terlaksana dan daya yang terbuang selama perjalanan dapat diminimalisirkan juga.

VII. ACKNOWLEDGMENT

Saya ingin mengucapkan terima kasih kepada Allah SWT atas segala yang diberikan-Nya sehingga makalah ini dapat selesai. Ucapan terima kasih juga ditujukan kepada kedua orang tua saya yang selalu mensupport sehingga saya dapat menjalankan perkuliahan di Institut Teknologi Bandung dengan baik. Tak lupa saya ingin mengucapkan terima kasih kepada dosen-dosen mata kuliah matematika diskrit, yaitu Bapak Dr. Ir. Rinaldi Munir dan Ibu Dra. Harlili, M.Sc. atas segala ilmu yang telah dibagikan untuk menyelesaikan makalah ini.

Dalam pembuatan makalah ini, sebagian besar data yang menjadi perhitungan merupakan data asumsi dari penulis. Sehingga penyempurnaan makalah perlu dilakukan.

REFERENCES

[1] K. H. Rosen, *Discrete Mathematics and Its Applications 7th*. New York: McGraw-Hill, 2012.
[2] Rinaldi Munir, *Matematika Diskrit*, Bandung: Informatika, 2009.
[3] J. K. Truss, *Discrete Mathematics for Computer Scientists*. New York: Addison-Wesley Publishing Company, 1991
[4] Andrew Duffy, 2000, *Transmitting Electricity*, Boston University, diakses 8 Desember 2016, http://physics.bu.edu/~duffy/sc545_notes04/transmission.html
[5] Bielisme, 2016. *Rugi/Susut Teknis Pada Sistem Distribusi Tenaga Listrik*,. diakses 7 Desember 2016, https://bielisme.wordpress.com/2016/06/10/rugisusut-teknis-pada-sistem-distribusi-tenaga-listrik/
[6] 2016, *Program 35.000 MW, Pemerintah akan Bangun 8.800 MW Dari EBT*, diakses 8 Desember 2016, http://www3.esdm.go.id/berita/energi-baru-dan-terbarukan/323-energi-baru-dan-terbarukan/8323-program-35000-mw-pemerintah-akan-bangun-8800-mw-dari-ebt.html
[7] *Paiton Unit Capacity*, diakses 8 Desember 2016, http://www.jawapower.co.id/
[8] *Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm)*, diakses 8 Desember 2016, http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/
[9] 2016, *Daftar tarif dasar listrik PLN 2016 dan Cek Tagihan Listrik Online* , diakses 9 Desember 2016, http://obengplus.com/articles/4518/1/Daftar-tarif-dasar-listrik-PLN-2016-dan-Cek-Tagihan-Listrik-Online.html

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 2016

Dery Rahman Ahaddienata, 13515097